Proyecto

Objetos distribuidos y persistencia con CORBA, Java y C++*

Alumno

Javier Carlos Ros Sánchez
frs@larural.es
Facultad de Informática
Universidad de Murcia

Directores

Jesús J. García Molina
jmolina@fcu.um.es
Facultad de Informática
Universidad de Murcia

Diego Sevilla Ruiz
dsevilla@fcu.um.es
Facultad de Informática
Universidad de Murcia

1 de septiembre de 1998

* Este proyecto tiene una asignación de **once créditos y medio**.

ÍNDICE

1. Introducción

En la actualidad el desarrollo de software está centrado en tres modelos de computación: orientado a objetos, componentes y distribución sobre Internet. En los últimos años hemos visto cómo se abandonaban las aplicaciones monolíticas basadas en *mainframes* por aplicaciones distribuidas, de modo que la heterogeneidad se ha convertido en uno de los aspectos fundamentales de los sistemas informáticos actuales: aplicaciones desarrolladas en distintos lugares geográficos, programadas en diferentes lenguajes y que funcionan sobre diferentes máquinas y sistemas operativos [OH97].

Las aplicaciones distribuidas han evolucionado desde el tradicional modelo de computación cliente-servidor de dos niveles hasta el modelo de objetos distribuidos resultado de aplicar una perspectiva orientada a objetos a la computación cliente-servidor. Las aplicaciones basadas en objetos distribuidos ofrecen un conjunto de servicios. De esta forma, se hace la distinción entre objetos clientes y objetos servidores. Un objeto cliente pide la realización de cierto servicio a un objeto servidor, a través de la invocación de alguna de sus operaciones y la posterior ejecución de ésta. Dicha petición constará de la operación en cuestión, el objeto sobre el cual se va a ejecutar y de los parámetros que acompañan a la operación. Ahora, los roles cliente y servidor no están predeterminados, sino que cualquier aplicación puede actuar como cliente o servidor. Además, los clientes y servidores podrán estar situados en cualquier parte del mundo, conectados a través de redes universales, como lo es Internet.

Por tanto, la filosofía cliente-servidor amplía su radio de acción, con el concepto de objetos distribuidos. Ahora existe una red de ordenadores, donde los objetos pueden existir en cualquier parte de ella. La computación cliente-servidor pasa a tener un ámbito más extenso, ya que la petición de servicios puede ser local (objetos cliente y servidor dentro del mismo programa, o en la misma máquina), o remota (ambos objetos están en distintas máquinas dentro de la misma red de comunicaciones), sin que por ello haya que cambiar el código.

Tras el éxito de la tecnología orientada a objetos ha surgido la tecnología de componentes como el paradigma que permitirá alcanzar el viejo sueño de construir el software como el hardware, mediante la integración de componentes ya existentes. Los objetos distribuidos serán los componentes software con una interfaz bien definida que podrán combinarse para crear nuevas aplicaciones.

El consorcio OMG (Object Management Group) creado para promover el uso de la tecnología orientada a objetos en un contexto distribuido, ha definido una arquitectura estándar para el desarrollo de aplicaciones basadas en objetos distribuidos. Esta arquitectura se denomina CORBA (*Common Object Request Broker Architecture*) y define los elementos para soportar ese tipo de aplicaciones [OMG98]:

i) El bus software que gestiona las peticiones y respuestas entre objetos y que se denomina ORB (*Object Request Broker*)
ii) Un conjunto de servicios disponibles para los objetos
iii) Un conjunto de facilidades comunes a todos los objetos
iv) *Frameworks* específicos para dominios de aplicación

Por propia definición, un objeto está definido como una entidad encapsulada, cuyo interior no es visible para los demás objetos. Por tanto, los objetos servidores muestran la funcionalidad disponible a los clientes a través de una interfaz pública. Esta interfaz vendrá definida por una lista con las signaturas de las operaciones y los tipos de los atributos que un objeto cliente puede requerir de un objeto servidor. CORBA define un lenguaje independiente del lenguaje de implementación denominado IDL para definir las interfaces de los objetos.

CORBA se ha visto influenciada por la aparición de Java y con ello la disponibilidad de ORB's implementados en Java o la posibilidad de trasladar un objeto servidor de un host a otro con diferente arquitectura (hardware + sistema operativo). Además, los clientes CORBA se pueden desarrollar como applets que se pueden integrar en los browsers de la Web.

1.1 Objetivos de este proyecto

En la actualidad existen varias plataformas que conforman con el estándar CORBA, tales como Orbix, Visibroker, SOM o JDK 1.2. El proyecto trata de cubrir dos objetivos.

Por un lado se elabora una documentación que permita a futuros desarrolladores e investigadores adentrarse en la tecnología CORBA de una forma sencilla y práctica. Esto se conseguirá mediante: i) una introducción de una forma práctica al estándar CORBA, al lenguaje IDL definido en dicho estándar y su transformación a los lenguajes de programación Java y C++, y ii) con un tutorial basado en un ejemplo práctico, que se irá desarrollando gradualmente, comenzando con una implementación básica que posteriormente se complete para abordar los distintos aspectos que conforman CORBA. En el tutorial se hará especial hincapié en la separación entre la funcionalidad y la interfaz gráfica, implementando clientes en modo texto, gráfico y *applets*, aplicando de forma adecuada patrones de diseño [GHJ+94].

Por otro lado se analizará el desarrollo de sistemas de información de gestión utilizando las ventajas que ofrece CORBA. Para ello, se desarrollará una aplicación para la gestión de un centro comercial virtual, discutiendo de forma razonada todas las decisiones de diseño que se han tomado. Una de las principales motivaciones de este desarrollo es el diseño de una infraestructura que permita integrar, de forma automática, bases de datos (ya sea relacional, objeto-relacional u orientada a objetos) con el paradigma de objetos distribuidos de CORBA, capturando de una forma genérica las necesidades de las aplicaciones de gestión con un uso intensivo de bases de datos.

Finalmente se presentarán unas conclusiones basadas en el trabajo realizado sobre la factibilidad de utilizar CORBA en las empresas, como herramienta para crear aplicaciones distribuidas.

1.2 Que NO es el objetivo de este proyecto

No se tratará de explicar en el proyecto el lenguaje Java, ni la utilización de este lenguaje para crear páginas Web.

Tampoco trataremos de realizar transmisiones seguras, dejando esto para un nuevo proyecto en el que se intente ampliar la aplicación realizada para incluir SSL en la transmisión de información.

El aspecto del programa no se cuidará especialmente, ya que hemos preferido centrarnos en las cuestiones relacionadas con CORBA.

No se trata de crear una aplicación que abarque toda la problemática de un centro de comercio virtual, sino de aplicar CORBA a un problema real.

1.3 Organización del proyecto

El proyecto se divide en seis capítulos. En este primer capítulo se describen los objetivos y estructura general del proyecto de una forma resumida.

El segundo capítulo introduce al lector en CORBA explicando las características principales de este estándar. Se explicará brevemente el lenguaje IDL, propuesto por el OMG para la definición de interfaces y el código en Java y C++ que se genera automáticamente a partir de una definición IDL.

El tercer capítulo cubre el primero de los objetivos definido en el apartado 1.2. Se desarrolla el juego *TicTacToe* paso a paso. Inicialmente se crea un esqueleto con el que se desarrolla un servidor y un cliente. Este último se implementa como una aplicación independiente y como un applet. A continuación, se mejorará el programa con la utilización del servicio de nombres que ofrece CORBA, y con el uso de *callback's*, explicando en cada caso para que sirven y como se utilizan, para a continuación mostrar el ejemplo aplicado en el juego. Por último se desarrollará la misma aplicación en C++, mostrando la interacción entre ambos lenguajes.

En el cuarto capítulo se aborda el desarrollo de una aplicación distribuida, como es un centro comercial virtual, de tal forma que se aplique CORBA a un caso práctico, obteniendo un programa que se puede instalar y poner en funcionamiento.

En el quinto capítulo se desarrollará una herramienta para la generación automática de aplicaciones con un alto grado de acceso a bases de datos. Con esto se cubrirá el segundo de los objetivos del proyecto.

Finalmente en el sexto capítulo se aportarán unas conclusiones basadas en la experiencia personal obtenida con el desarrollo con CORBA y en que medida puede éste utilizarse en aplicaciones de gestión.

Se añaden al final del documento varios apéndices con el código fuente de las aplicaciones realizadas.

1.4 Lenguajes, Herramientas y Tecnologías

Para la realización del proyecto se han utilizado una serie de herramientas que se describen a continuación.

En primer lugar el lenguaje elegido para desarrollar las aplicaciones es *Java* ya que aporta grandes ventajas al desarrollo con CORBA [OH97]. Para dar un punto de vista más plural se ha decidido utilizar también *C++* para realizar las mismas aplicaciones.

La herramienta que aportará una implementación de un ORB y un servicio de nombres serán las *JDK 1.2* en el caso de *Java*, por seguir el estándar definido por el OMG sin añadir ninguna solución propietaria, a la vez que su obtención es gratuita, lo cual se buscará para todas las herramientas utilizadas durante el proyecto. Para el caso de *C++* se utilizará la implementación realizada por *Object-Oriented Concepts Inc.* (*ORBacus*).

También se utilizará *CGI*, para implementar los programas *CGI* se utilizará el lenguaje de script *Perl*.

Como herramientas adicionales se tienen un servidor de bases de datos (*Postgresql*), *JDBC* para el acceso a la base de datos desde *Java*, *pq++* para el acceso a la base de datos desde *C++* y un servidor de Web (*Apache*) ambos ejecutándose en un sistema *Linux*.

En el lado de los clientes se ha utilizado el *Internet Explorer 5.0*, junto con el *Java Runtime Enviroment 1.2*, para permitir la ejecución de *applets* implementados con las *JDK 1.2*.

Las librerías gráficas utilizadas para la realización del proyecto son las *Swing* en el caso de Java y las librerías *qt* de *Troll Tech* para los clientes gráficos en *C++*.

Finalmente para la realización del compilador del capítulo 5 se utilizarán las herramientas *lex* y *yacc* junto con el lenguaje de programación *C++*.

2. Sistemas distribuidos con Java/CORBA

Este capítulo discutirá las partes esenciales de un sistema CORBA, y será referenciado a través del resto del proyecto. No se trata en este capítulo de abordar toda la teoría sobre CORBA, sino de dar una visión general suficiente para comprender el resto del documento. Para más información se puede acudir a la bibliografía, por ejemplo a [Sev98] y [OH97]

2.1 Objetos distribuidos

En muchos sentidos CORBA se puede considerar como la evolución natural del paradigma Cliente/Servidor. Este paradigma surge en los años 80 en respuesta a la necesidad de compartir información centralizada con un gran número de usuarios. Normalmente una arquitectura Cliente/Servidor de dos niveles divide el software funcionalmente de tal forma que las aplicaciones de los clientes realizan tanto el proceso de negocio como la interfaz de usuario. El servidor es usado como un almacén de datos. Los problemas de esta arquitectura son evidentes: el mantenimiento del software se hace muy complicado ya que las aplicaciones clientes son muy grandes mezclando la interfaz con la lógica de negocio, los accesos directos a las bases de datos por parte de los clientes pueden provocar inconsistencias en éstas al no haber ninguna capa que compruebe la consistencia de la base de datos antes de realizar la operación, etc.

Como una evolución natural al paradigma Cliente/Servidor se ha producido una división de responsabilidades en tres niveles:

- **Presentación**: Software relacionado con la interfaz de usuario.
- **Lógica de negocio**: Software que implementa los procesos de la organización.
- **Almacenamiento y recuperación de datos**: Software relacionado con las bases de datos.

La separación lógica y pérdida de acoplamiento entre niveles los aíslan del cambio en otros niveles. El segundo nivel que encapsula la lógica de negocio y de aplicación se conoce como el nivel medio. Los servicios que ayudan a soportar la implementación del nivel medio son conocidos como *middleware*.

Las características de un buen *middleware* serían:

- Permitir una separación clara de la lógica de negocio.
- Ayudar en el soporte de la reusabilidad al separar la lógica de negocio de la presentación.
- Debe estar basado en un estándar, para permitir diferentes implementaciones del *middleware* por diferentes fabricantes, compatibles entre sí.

- Debe exhibir una gran fiabilidad y disponibilidad, para poder ser aprovechado en todo tipo de problemas.
- Debe ser altamente escalable, permitiendo el crecimiento de las aplicaciones que utilicen los servicios de dicho *middleware*.
- Debe soportar comunicación entre objetos, implementados en diferentes lenguajes y plataformas, conectadas por una red.

Por otro lado surge, gracias al aumento de potencia y disminución de precios en los ordenadores y a los avances en redes, la computación distribuida como sustitución a los grandes *mainframes*. La computación distribuida es el uso de múltiples equipos en red cooperando para completar un proceso. Si se toma este concepto y se une a un paradigma de tres niveles obtenemos un sistema donde cada uno de los niveles se ejecuta en un nodo distinto (pudiendo cada nodo estar compuesto por varios ordenadores).

Existen múltiples maneras de implementar sistemas distribuidos, como por ejemplo utilizando programación a nivel de socket, los RPC (*Remote procedure calls*), etc. Para más información sobre las distintas técnicas ver [Sev98].

Todos estos sistemas distribuidos están basados en la programación estructurada, pero con el auge de la programación orientada a objetos en los últimos tiempos (con lenguajes como C++, Eiffel, Smalltalk o Java) ha aparecido un nuevo tipo de sistemas distribuidos, donde lo que se distribuyen son objetos en vez de ser procedimientos.

Un objeto puede tener una interfaz pública, que puede ser accedida desde cualquier otro objeto, tanto si se encuentra en la misma máquina como si se encuentra en otra comunicadas ambas mediante una red. De esta forma obtenemos un conjunto de objetos que ofrecen una serie de servicios y que se encuentran distribuidos en una red, pudiendo utilizar cada uno el resto de los servicios de los demás.

En este nuevo modelo de objetos distribuidos, cada objeto tiene referencias a los demás objetos, pero se le oculta la localización de éstos, de tal forma que al invocar un método, la invocación puede ser local, dentro de la misma máquina o puede ser remota, a través de una red. Lo cual quiere decir que un sistema distribuido orientado a objetos, debe permitir el paso de mensajes a objetos remotos de una forma transparente.

2.2 Introducción a CORBA

La unión del paradigma de programación orientado a objetos con la topología Cliente/Servidor, con la intención de facilitar la interacción de objetos, ha provocado la aparición de CORBA. La arquitectura CORBA (Common Object Request Broker Architecture) ha sido definida por el OMG con el fin de crear un estándar que especifique los elementos básicos para dar soporte al desarrollo de aplicaciones distribuidas.

CORBA es superior a otros productos *middleware* por muchas razones, entre las que destacan:

- Es un estándar soportado por la industria, no propietario. Esta siendo desarrollado por la mayoría de las grandes empresas de construcción de software.
- Fuerza la separación de la interfaz de un objeto de su implementación.
- Es escalable.
- El soporte para la reusabilidad es inherente.
- Es transparente del lenguaje y la plataforma.
- Provee independencia del vendedor.
- Abstrae al desarrollador de las comunicaciones.

A continuación se describirá el funcionamiento de CORBA.

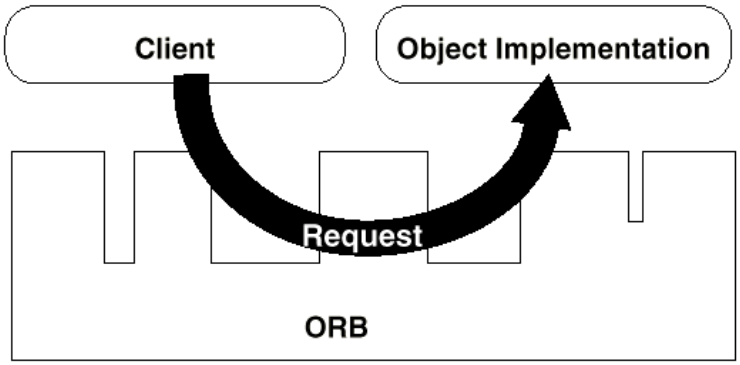

Figura 1: Una petición lanzada a traves del ORB (Tomada de [OMG98])

La Figura 1 muestra un paso de mensaje entre un cliente y un objeto servidor. El *client* representa al objeto cliente que quiere realizar una operación (paso de mensaje) sobre otro objeto, y la *object implementation* es el código y datos que implementan a ese otro objeto, esto es el objeto servidor.

***Nota*:** El OMG llama "*object implementation*" a lo que nosotros llamaremos durante el resto del proyecto "*objeto servidor*".

El ORB es el responsable de todos los mecanismos requeridos para encontrar el objeto servidor de la petición, de preparar el objeto servidor para recibir la petición y de transmitir los datos realizando la petición. La interfaz que el cliente ve es totalmente independiente de donde se encuentre el objeto servidor, del lenguaje de programación utilizado, o de cualquier otro aspecto que no esté reflejado en la interfaz del objeto.

La Figura 2 muestra la estructura de un ORB individual. Las interfaces al ORB son mostradas con cuadros sombreados, y las flechas indican si el ORB es llamado o realiza una llamada a través de la interfaz.

Véase nota a continuación para una descripción de cada uno de los elementos.

El objeto servidor puede elegir qué tipo de adaptador de objetos (OA) quiere utilizar, esta decisión está basada en que tipos de servicios requiere, también puede decidir no utilizar ningún OA, y acceder directamente al ORB.

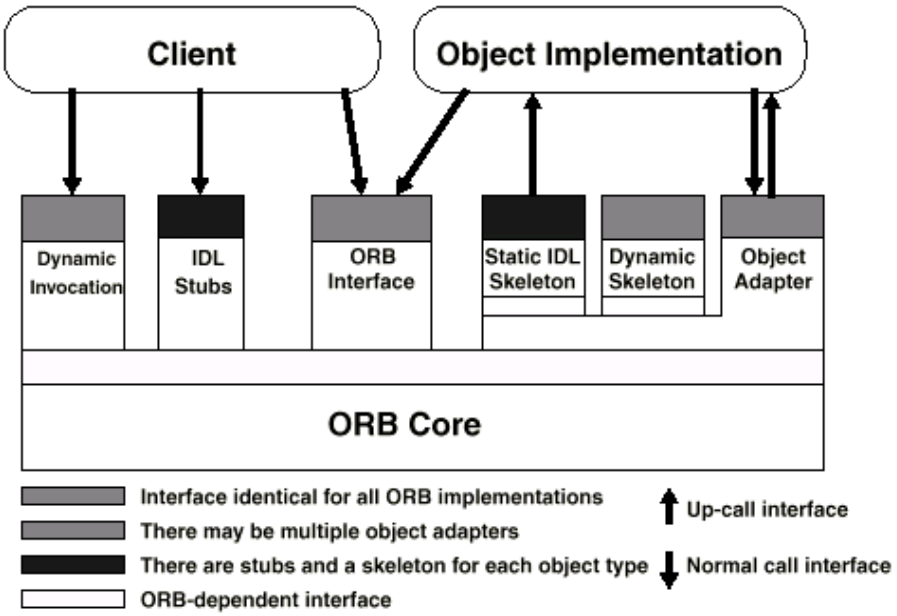

Figura 2: La estructura de un ORB. (Tomada de [OMG98])

***Nota*:** El significado de los diferentes componentes de la Figura 2 es:

Client: Un cliente es un objeto que requiere servicios a través de las llamadas a métodos de un objeto servidor. Destacar que un objeto toma el rol de cliente en un mensaje y con respecto a un servidor, aunque en otro momento puede tomar el rol servidor.

Object Implementation: Es el rol que toma un objeto cuando se produce la invocación de uno de sus métodos de forma remota.

IDL Stubs: Provee la interfaz estática a los servicios del objeto servidor. Define cómo el cliente invocará a los correspondientes servicios del servidor, es como un proxy local al servidor remoto.

Dynamic Invocation: Le permite descubrir al cliente los métodos invocables en el servidor en tiempo de ejecución. Obtiene metainfomación del servidor.

ORB Interface: Es una pequeña API de servicios locales que pueden ser interesantes para una aplicación. Por ejemplo un servicio para convertir de "referencias a objetos" a "cadenas de caracteres", y viceversa.

Static IDL Skeleton: Provee una interfaz estática a cada servicio exportado por el servidor. Es la forma en la que el ORB invoca los métodos del servidor.

Dynamic Skeleton: Le permite al ORB realizar invocaciones de métodos en un objeto que no ha definido qué interfaz implementa en tiempo de compilación.

Object Adapter: Es la principal forma por la que un servidor accede a los servicios de un ORB. Hay múltiples tipos según los servicios implementados, cabe destacar el *Basic Object Adapter (BOA)* y el *Portable Object Adapter (POA)*. También se puede prescindir de un adaptador de objetos y acceder directamente al ORB.

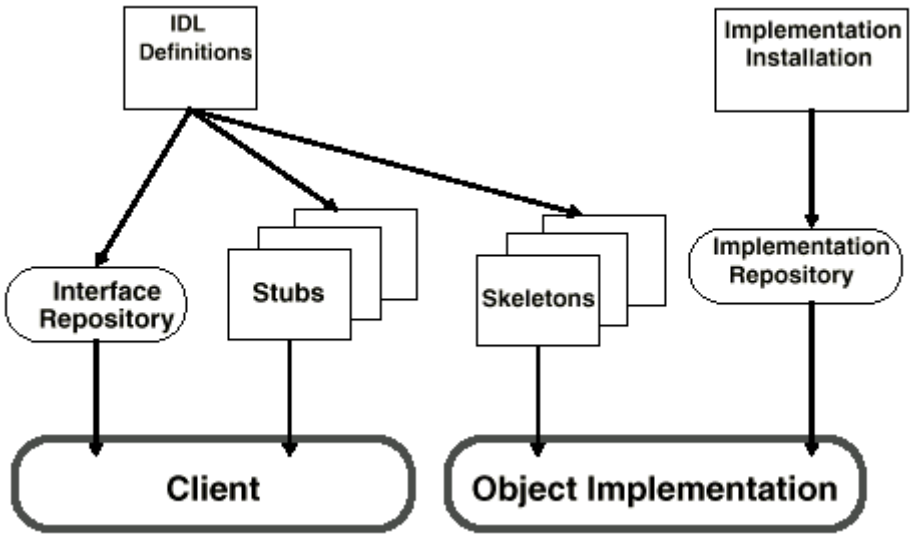

Figura 3: Almacenes de interfaces e implementaciones (Tomada de [OMG98])

La Figura 3 muestra cómo una interfaz se hace disponible a los clientes y a los servidores. La interfaz es definida en IDL, la definición es usada para generar los stubs para accesos estáticos desde un cliente o para dar de alta el servidor en el *Interface Repository* para accesos dinámicos de un cliente. De la interfaz también se generan los skeletons del objeto servidor para permitir a éste recibir peticiones remotas.

La interfaz del objeto servidor es almacenada en el *Implementation Repository* al ser instalado (al darse de alta en el ORB), para su uso durante las peticiones, es decir para que el ORB sepa que peticiones puede realizar a que objetos.

Un cliente es un objeto que utiliza una referencia a otro objeto servidor, para invocar operaciones sobre dicho objeto. El cliente conoce sólo la estructura lógica del objeto, de acuerdo con su interfaz. Se debe tener en cuenta que, aunque consideremos un cliente a un objeto que realiza peticiones a otro, es importante recordar que lo será únicamente en relación con dicho servidor y en un momento dado, al realizar la petición. Un objeto cliente puede ser a su vez servidor si recibe una petición en otro momento.

El objeto servidor provee de semántica a la interfaz, normalmente almacenando información para los atributos de la interfaz y definiendo código para los métodos de la interfaz.

2.3 Interface Definition Language, IDL

El IDL es el lenguaje definido por el OMG para la especificación de interfaces. De esta forma se pueden definir las operaciones que implementará un objeto servidor

independientemente del lenguaje con el que se implementen posteriormente. Los clientes únicamente necesitan conocer la interfaz proporcionada en IDL para saber qué operaciones pueden realizar sobre el objeto, y utilizarán el stub generado a partir del código IDL para realizar dichas peticiones. También se pueden realizar peticiones utilizando invocaciones dinámicas, pero este punto no será abordado en el proyecto actual.

El lenguaje IDL está basado en el lenguaje de programación C++, de hecho el preprocesador utilizado en C++ es utilizado también para el nuevo lenguaje. Por lo tanto las siguientes directivas de preprocesador se pueden utilizar con la misma semántica que en C++: *#define, #undef, #include, #if, #ifdef, #elif, #else, #endif, #defined, #error, #pragma*.

El bloque de construcción del lenguaje son los módulos. Los módulos permiten al implementador agrupar las interfaces en bloques. Un módulo puede contener otros módulos a su vez, además de interfaces. Podríamos comparar los módulos con los paquetes del lenguaje Java.

Los módulos modifican el alcance del nombrado en interfaces, de tal modo que para acceder a una interfaz dentro del módulo actual basta con escribir su nombre, pero si queremos acceder a una interfaz en otro módulo hay que especificarlo de la siguiente forma: *Modulo::interface*.

A continuación se muestra un ejemplo de módulo que incluye dos interfaces:

```
module Centro {
       interface Internauta { ... };

       interface Comercio { ... };

       ...
};
```

***Nota*:** Todos los ejemplos incluidos en este capítulo proceden de los capítulos siguientes, de tal forma que la lectura de este capítulo ayude a la de los siguientes.

Una interfaz provee la descripción de la funcionalidad que tendrá un objeto. La definición de la interfaz da toda la información necesaria al desarrollador de un cliente sobre qué puede usar para interactuar con el objeto servidor. Por supuesto, es necesaria una documentación de dicha interfaz para comprender su semántica, es decir cuándo y para qué utilizarla.

El cuerpo de una interfaz puede contener los siguientes tipos de declaraciones:

- Declaraciones de constantes.
- Declaraciones de tipos.
- Declaraciones de excepciones.
- Declaraciones de atributos.

- Declaraciones de operaciones, incluyendo los argumentos y el valor devuelto por ésta.

A continuación podemos ver un ejemplo con algunas de estas declaraciones:

```
1. interface CarroCompra {
2.     typedef sequence<long> ListaCantidad;
3.     typedef sequence<string> ListaComercios;

4.     attribute Internauta refInternauta;
5.     attribute ListaProductosCantidad refProducto;

6.     void anadirProducto(in Producto prod, in long cantidad,
              in string nombreComercio);
7.     void listarProductos(out ListaProductos prod,
              out ListaCantidad cantidad, out ListaComercios comercios);
8.     void cambiarCantidad(in Producto prod, in string comercio,
              in long cantidad);
9. };
```

Los atributos tienen la siguiente sintaxis:

'attribute' <tipo_atributo> <nombre_atributo>;

Puede verse en el ejemplo anterior en la línea 4 como se define el atributo *refInternauta* del tipo *Internauta.*

Y las operaciones tienen la siguiente sintaxis:

<tipo_valor_devuelto> <nombre_metodo> (<direccion_parametro> <tipo_parametro> <nombre_parametro>, ...);

En el ejemplo anterior se puede ver en la línea 7 el método *listarProductos* que no devuelve nada (tipo *void*) y que tiene tres parámetros.

La dirección del parámetro puede ser de uno de los siguientes tipos:

- *in*: El parámetro es pasado del cliente al objeto invocado.
- *out*: El parámetro es pasado del objeto invocado al cliente.
- *inout*: El parámetro es pasado en ambas direcciones.

Normalmente, el cliente debe esperar a que termine la ejecución del método en el objeto para continuar con su ejecución, pero existe un tipo de operaciones *de un sentido* que no bloquean al cliente cuando las invoca. Para definir una operación de este tipo basta con anteponer la palabra clave *oneway* a la definición del método.

Una interfaz puede además ser derivada de otra interfaz. Esta nueva interfaz puede ser extendida añadiendo nuevas definiciones a las ya definidas en la interfaz base. Además se pueden redefinir elementos de la interfaz base, si se desea acceder a los elementos originales se utilizaría el operador de resolución de ámbito de C++ (*::*).

El lenguaje IDL permite la herencia múltiple haciendo que una interfaz extienda más de una interfaz base. Para especificar que una interfaz extiende a otra se utiliza la siguiente sintaxis:

'interface' <nombre_interface> : <nombre_base1>, <nombre_base2>, ... {...};

En el siguiente ejemplo se pueden ver dos interfaces, *Internauta* y *Socio*, donde ésta hereda de la primera, ambas con una serie de atributos.

```
interface Internauta {
       attribute string nombre, apellido1, apellido2,
              telefono, correoElectronico, numeroCuenta, direccion;
};

interface Socio : Internauta {
       attribute string idSocio, clave;
};
```

Tanto para los atributos como para los argumentos de las operaciones se pueden utilizar cualquiera de los tipos básicos que se describen en la tabla 1.

Tabla 1: Tipos IDL básicos

Tipo	*Identificador IDL*	*Descripción*
punto flotante	float	IEEE números en punto flotante de precisión simple
	double	IEEE números de precisión doble
entero	long	32 bit
	short	16 bit
	unsigned long	32 bit
	unsigned short	16 bit
carácter	char	Una cantidad de 8 bits
booleano	boolean	Verdadero o Falso
octeto	octet	Una cantidad de 8 bits que está garantizado que no va a experimentar ninguna modificación durante la transmisión
any	any	Permite la especificación de valores que pueden expresar un tipo IDL arbitrario

A partir de estos tipos se pueden construir otros nuevos: registros, enumerados y uniones discriminadas.

Un registro es un tipo de datos que permite a los elementos ser agrupados juntos de una forma útil para reflejar la relación que existe entre ellos. Por ejemplo, el tipo color está compuesto de un nombre y de la cantidad de rojo, verde y azul.

```
struct color {
              string nombre;
              short rojo, verde, azul;
};
```

Un tipo enumerado consiste en una lista ordenada de identificadores, por ejemplo los meses del año:

```
enum meses {Ene, Feb, Mar, Abr, May, Jun, Jul, Ago, Sep, Oct, Nov, Dic};
```

Finalmente un tipo unión es un tipo registro cuya finalidad es ahorrar espacio, ya únicamente uno de los campos puede tener un valor asignado en un momento dado, de tal forma que la cantidad de espacio necesaria para la unión es la cantidad de espacio necesaria para el elemento mayor. El campo etiqueta es usado para especificar qué miembro de una unión tiene asignado un valor actualmente.

Por ejemplo un identificador que puede ser una cadena de caracteres o un entero:

```
union identificador switch (long) {
      case 1: string nombre;
      case 2: long id;
};
```

A cada tipo se le puede dar otro nombre para hacerlo más representativo, para ello se utiliza la palabra clave *typedef*. Por ejemplo si las edades se van a representar con un entero en vez de definir los atributos que representen edades como entero se puede definir el tipo *edad* de la siguiente forma:

```
typedef long edad;
```

Todos los tipos se pueden agrupar en arrays y secuencias.

Un array es un conjunto de elementos del mismo tipo, de tamaño fijo. Los arrays pueden ser multidimensionales especificando el tamaño de cada dimensión en el momento de su definición.

Por ejemplo, un tablero en un array bidimensional de fichas:

```
ficha tablero[3][3];
```

Las secuencias son arrays unidimensionales con dos características: Un tamaño máximo (que es determinado en tiempo de compilación, y es opcional pudiendo ser una secuencia sin límites) y una longitud máxima (que es determinada en tiempo de ejecución). Una secuencia no tiene un tamaño fijo.

Por ejemplo una lista de productos sin límite sería:

```
typedef sequence<Producto> ListaProductos;
```

Y con un límite de 100 productos:

```
typedef sequence<Producto,100> ListaProductos;
```

Del mismo modo que se definen variables también se pueden definir constantes:

```
const long longitud_maxima = 1000;
```

Por último el IDL define también un mecanismo de excepciones como mecanismo estándar para el tratamiento de errores. Una operación puede lanzar una excepción indicando que se ha producido un error.

Por ejemplo cuando se pone una ficha pueden haber dos errores, si al jugador no le toca, o si la casilla ya está ocupada:

```
tablero ponerFicha(in Jugador jug, in short x, in short y)
      raises(JugadorInvalido,CasillaOcupada);
```

IDL es un lenguaje neutral y es necesaria una transformación al lenguaje de programación en el que se vaya a implementar finalmente el objeto. Esta transformación es llamada *mapping* y está definida por el OMG, de tal forma que se puede realizar de una forma mecánica.

Cabe destacar que ciertas clases pueden ser especificadas utilizando el lenguaje IDL, en este caso el lenguaje se utiliza únicamente como un lenguaje de especificación, aunque para la transformación a un lenguaje de programación específico no se sigue el estándar que define el OMG, esta especificación se dice que está en pseudo-IDL.

2.4 Transformación de IDL a Java

Este capítulo describirá la transformación de IDL a Java que define el OMG.

Los módulos de IDL son transformados en paquetes de Java de la siguiente forma:

module m {...}; → *package m;*

Una interfaz de IDL se transforma en una interfaz pública de Java, con los métodos correspondientes a las operaciones definidas en el código IDL, y para cada atributo uno o dos método de acceso (Uno para obtener el valor y otro para modificarlo) según sea sólo lectura o no.

interface i {...}; → *public interface i {...}*

Los nombres utilizados por los identificadores en el IDL, son los mismos nombres que se utilizan en las clases Java. Cuando un identificador coincide con una palabra clave de Java se le añade el símbolo '_' delante para diferenciarlo de ésta.

En la Tabla 2 se encuentran las transformaciones de los tipos básicos de IDL a Java.

Tabla 2: Transformación de tipos básicos

Tipo IDL	*Tipo Java*
float	float
double	double
long	int
short	short
unsigned long	int
unsigned short	short
char	char

boolean	boolean
octet	byte

Un enumerado de IDL se transforma a una clase final de Java. Cada elemento del enumerado se transforma en dos constantes static, una un int con un '_' delante del nombre del elemento, y la otra una instancia de la propia clase con el mismo nombre del elemento que representa el valor del elemento.

Por ejemplo, la transformación para un enumerado con los meses:

```
enum meses {Ene, Feb, Mar};
```

sería:

```
public final class meses {
     public static final int _Ene = 0,
                             _Feb = 1,
                             _Mar = 2;
     public static final meses Ene = new meses(_Ene);
     public static final meses Feb = new meses(_Feb);
     public static final meses Mar = new meses(_Mar);
     public int value() {
         return _value;
     }
     public   static   final   meses   from_int(int   i)       throws
org.omg.CORBA.BAD_PARAM {
           switch (i) {
             case _Ene:
                 return Ene;
             case _Feb:
                 return Feb;
             case _Mar:
                 return Mar;
             default:
                     throw new org.omg.CORBA.BAD_PARAM();
           }
     }
     private meses(int _value){
         this._value = _value;
     }
     private int _value;
}
```

Por otro lado, los registros se transforman a una clase final que contiene un atributo por cada campo de la estructura. Se provee a la clase de dos constructores, el primero tiene como argumentos los valores iniciales para los campos, y el otro los inicializa a cero o null.

A continuación se muestra un ejemplo con el registro que representa un color:

```
struct color {
              string nombre;
              short rojo, verde, azul;
};
```

Que se transforma en:

```
public final class color {
    // instance variables
    public String nombre;
```

```
        public short rojo;
        public short verde;
        public short azul;
        // constructors
        public color() { }
        public color(String __nombre, short __rojo, short __verde, short
                                                                  __azul) {
           nombre = __nombre;
           rojo = __rojo;
           verde = __verde;
           azul = __azul;
        }
    }
```

Para una unión la transformación genera una clase final que tiene las siguientes características:

- El mismo nombre que la unión en IDL.
- Un constructor por defecto
- Un método de acceso para el discriminante de la unión llamado *discriminator()*
- Un método de acceso para cada variante
- Un método para cambiar el valor a cada variante
- Un método para cambiar el valor a cada variante que tiene más de un *label*
- Un método para cambiar al valor por defecto

Los tipos definidos por el usuario utilizando *typedef* son sustituidos por los tipos originales en el resto de la transformación. De tal forma que si se define

```
typedef long edad;
```

Cuando aparezca un identificador del tipo *edad* se transformará a un identificador del tipo *int* (que corresponde con la transformación de *long*).

Las secuencias y los arrays de IDL son transformados en arrays de Java.

Cabe destacar que una interfaz de IDL no sólo genera la correspondiente interfaz en Java, sino que además se crean dos clases adicionales con los nombres *<identificador>Helper* e *<identificador>Holder*.

La clase *Helper* contiene una serie de métodos estáticos que sirven para manipular el tipo que se está generando, como la escritura del objeto en un stream, o la lectura de éste, o la transformación de un objeto del tipo *Object* al tipo de la clase (*narrowing*).

La clase *Holder* se utiliza para almacenar una referencia a un objeto de la clase, esto es muy útil en Java porque los parámetros son pasados todos por valor, con lo que no se podrían implementar los parámetros *out* o *inout*. Para implementar este tipo de parámetros se sustituye el propio parámetro por un objeto de la clase Holder correspondiente, de tal forma que modificando la referencia que contiene esta clase se modifique el parámetro a la vuelta. Por ejemplo para una clase '*c*' véase la figura 4.

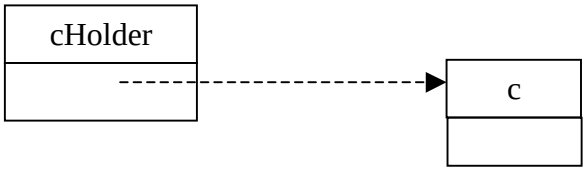

Figura 4 : Diagrama para una clase Holder

Como ejemplo de su utilización, supongamos un método que tiene parámetros del tipo *out*:

```
void listarProductos(out ListaProductos prod, out ListaCantidad
                          cantidad, out ListaComercios comercios);
```

Su transformación sería:

```
void listarProductos(Centro.ListaProductosHolder prod,
             Centro.CarroCompraPackage.ListaCantidadHolder cantidad,
             Centro.CarroCompraPackage.ListaComerciosHolder comercios);
```

Por último la transformación de las excepciones de CORBA a Java es muy natural, generándose una clase final que hereda de *org.omg.CORBA.UserException.*

Por ejemplo, para la excepción que indica que ya hay dos jugadores cuando intenta entrar en la partida un tercero:

```
exception YaHayDosJugadores {};
```

Se transforma en:

```
public final class YaHayDosJugadores
       extends org.omg.CORBA.UserException {
    // constructor
    public YaHayDosJugadores() {
       super();
    }
}
```

Y los métodos que lanzan excepciones simplemente son definidos lanzando dichas excepciones. La operación IDL

```
tablero anadirJugador(in Jugador jug,out ficha usar)
      raises(YaHayDosJugadores);
```

quedaría como:

```
TicTacToe.ficha[][] anadirJugador(TicTacToe.Jugador jug,
                                        TicTacToe.fichaHolder usar)
        throws TicTacToe.JuegoPackage.YaHayDosJugadores;
```

2.5 Transformación de IDL a C++

Este capítulo describirá la transformación de IDL a C++ que define el OMG.

Los módulos de IDL son transformados en espacios de nombre en C++, los cuales se definen con la palabra clave *namespace*, de tal forma que una interfaz se convertirá en una clase que pertenecerá al espacio de nombres correspondiente al módulo en el que fue definida.

Cuando se desee utilizar una clase se deberá calificar completamente con el operador de ámbito. Por ejemplo para la interfaz I en el módulo M tendríamos en C++:

```
module M {                          namespace M {
    interface I {};                 class I { ... };
};                                  }

module M2 {                         namespace M2 {
    interface I2 {                  class I2 {
         M::I op();                        M::I op();
    };                              };
};                                  }
```

También se puede usar la palabra clave "using" de C++ para importar un espacio de nombres pudiendo de esta forma acceder directamente a la clase sin necesidad de utilizar el operador de ámbito.

Si el compilador de C++ no permite el uso de espacios de nombres, el nombre de las clases estará compuesto por el nombre del módulo más el nombre de la interface separados ambos por el carácter de subrayado.

```
class M_I { ... };
```

Una interface es transformada en una clase C++ que contiene definiciones públicas de los tipos, constantes, operaciones y excepciones definidas en la interface.

Para poder utilizar variables del tipo de las clases generadas a partir de una interface en IDL, se deben seguir una serie de reglas, que se describirán a continuación.

En primer lugar un programa no puede crear o mantener una instancia de una clase interfaz, tampoco puede usar un puntero (I*) o una referencia (I&) a una clase interfaz.

El uso de una clase interfaz denota una referencia a un objeto. Para dicho uso se generan a partir de una interfaz I dos tipos llamados I_var e I_ptr. Una variable de cualquiera de estos dos tipos podrá ser utilizada para acceder a las operaciones de la interfaz utilizando el operador '->' "I_xxx obj; obj->op()".

El código cliente usará la mayoría de las veces el tipo I_var porque el objeto referenciado por una variable de este tipo (este objeto contiene los recursos necesarios para poder acceder al objeto remoto, es decir el stub del cliente) será automáticamente liberado cuando la variable sea liberada, o cuando se le asigne un nuevo valor. Por otro lado el tipo I_ptr es más primitivo, siendo más cercano a la semántica de puntero en C++.

A una variable del tipo I_var puede ser asignada otra del tipo I_ptr y viceversa, sin necesidad de ningún casting, pero debe tenerse cuidado porque puede producirse una

liberación de un objeto que aún se esté utilizando. Como por ejemplo en el código siguiente:

```
I_ptr obj_ptr;
obj_ptr = /* Se obtiene la referencia a un objeto */
{
       I_var obj_var;
       obj_var = obj_ptr;
              /* Al salir del ámbito de I_var se libera el objeto
               * automáticamente */
}
obj_ptr -> op(); /* Error el objeto ha sido liberado */
```

Cuando una interfaz hereda de otra (por ejemplo B hereda de A) deben poder realizarse las siguientes operaciones:

```
A_ptr=B_ptr
Object_ptr=B_ptr
A_ptr=B_var
Object_ptr=B_var
```

Pero debe dar error de compilación la asignación *A_var=B_var*, en su lugar debe utilizarse el método *_duplicate* que se explica posteriormente:

```
A_var=B::_duplicate(B_var)
```

Aunque si es legal asignar dos variables del tipo B_var entre sí:

```
B_var1 = B_var2
```

La clase Object es la clase base para cualquier clase CORBA que se defina. En esta clase se definen dos operaciones:

```
void release(Object_ptr obj); /* Libera el objeto */
Boolean is_nil(Object_ptr obj); /* Comprueba si el objeto es nil */
```

En toda clase generada a partir de una interface se define la operación *_duplicate* cuya función es devolver una nueva **referencia** al objeto que se le pasa. Se define en cada clase de la siguiente forma:

```
       static I_ptr _duplicate(I_ptr obj);
```

También se define la siguiente operación:

```
       static I_ptr _narrow(Object_ptr obj);
```

cuya función es la misma, generar una nueva referencia, pero que se diferencia de *_duplicate* en que el parámetro que recibe puede ser de cualquier tipo, si el tipo en tiempo de ejecución del parámetro no corresponde con una subclase de I se devolverá *nil*.

También se define la operación:

```
static I_ptr _nil();      /* Devuelve nil */
```

Los strings se convierten a *char** en C++. Pero para facilitar su uso se define el tipo *String_var*, de forma que se libere el espacio automáticamente. Además se definen los métodos siguientes para el manejo del espacio de memoria (no se debe utilizar alloc, dealloc, ...)

```
namespace CORBA {
        char *string_alloc(ULong len);
        char *string_dup(const char*);
        void string_free(char *);
        ...
}
```

El método *string_alloc* reserva espacio para *len+1* caracteres, *string_dup* realiza una copia de una cadena y *string_free* libera el espacio reservado por cualquiera de las dos anteriores.

Se debe tener especial cuidado al asignar una cadena de caracteres a una variable de tipo *String_var*, ya que ésta espera que se haya almacenado espacio para la cadena utilizando los métodos anteriores, por tanto la siguiente sentencia sería incorrecta:

```
String_var s = "cadena de texto";  /* ERROR */
```

Debería hacerse:

```
String_var s = string_dup("cadena de texto");
```

Las estructuras, uniones y secuencias se transforman en clases C++.

Los enumerados y los registros de IDL son transformados a sus correspondientes estructuras en C++.

Cuando se define un tipo secuencia se genera un tipo para dicha secuencia y un tipo *_var* para manejar las secuencias de forma automática:

```
typedef sequence <string> l;
```

Para la sentencia anterior en IDL se generaría el tipo *l* de una clase especial parametrizada (si el compilador lo permite) que maneja los elementos de la secuencia. Además también se genera el tipo *l_var* para el manejo automático del espacio de los elementos de la lista.

En el caso de los arrays se generan dos tipos:

```
typedef string a[10];
```

Para esta sentencia se genera el tipo *a* que es un array de elementos del tipo definido, es decir hay una correspondencia directa con C++. También se genera el tipo *a_slice* que corresponde con el mismo tipo que *a* pero quitándole una dimensión al array, es decir en este caso con cadenas de caracteres.

Para cada tipo array se define además una función *a_dup* para generar duplicados de un array.

Para cualquier otro *typedef* se genera una sentencia similar en C++.

En cuanto a las operaciones de las interfaces son transformadas en métodos en las clases generadas. Para los parámetros de las operaciones se pueden distinguir cuatro casos, cuando el tipo del parámetro corresponde a otra interfaz, cuando corresponde con un tipo básico, cuando es un tipo secuencia o cuando es un tipo array:

```
//IDL
interface Param {};

typedef sequence <string> lista;
typedef string array[10];

interface I {
       Param op(in Param ent, out Param sal, inout Param entsal);
       long  op2(in long ent, out long sal, inout long entsal);
       lista op3(in lista ent, out lista sal, inout lista entsal);
       array op4(in array ent, out array sal, inout array entsal);
};
```

Para este ejemplo en IDL que tiene cuatro operaciones, una con cada uno de las posibilidades, se generaría el siguiente código en C++:

```
Param_ptr op(Param_ptr ent, Param_ptr& sal, Param_ptr& entsal);
CORBA_Long op2(CORBA_Long ent, CORBA_Long& sal, CORBA_Long& entsal);
lista* op3(const lista& ent, lista*& sal, lista& entsal);
array_slice* op4(const array ent, array_slice*& sal, array entsal);
```

3. Tutorial

A continuación se realizará un tutorial, en el que se describirá paso a paso el desarrollo de una aplicación para un juego distribuido como es el TicTacToe. La finalidad de este tutorial es realizar un documento con el que se pueda aprender la utilización de CORBA de una forma práctica, para facilitar la realización de otros proyectos basados en esta tecnología.

El problema seleccionado es abordado en el libro [PW98], pero desde un punto de vista distinto considerando una única solución al problema, y realizando la implementación con el uso de métodos propietarios de Visibroker, es decir que no se podría migrar esta aplicación a otro ORB. En nuestro caso se ha decidido utilizar sólo la interfaz definida por el OMG, de tal forma que puede ser portada a otro ORB.

La herramienta elegida para el desarrollo del tutorial ha sido la proporcionada por Sun, *Java Development Kit* en su última versión disponible en la actualidad, es decir la *1.2*. Además, también se utiliza la herramienta *idltojava* proporcionada también por Sun de una forma gratuita. La explicación de cómo se usan las diferentes herramientas se realizará sobre la marcha en el tutorial según sean requeridas.

3.1 Juego TicTacToe

El juego que se va a desarrollar es el TicTacToe, este es un juego de tablero muy popular en el que participan dos jugadores, que se enfrentan por conseguir la victoria. El tablero es un cuadrado formado por nueve casillas, organizadas en tres filas y tres columnas (Figura 5).

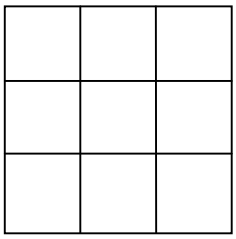

Figura 5 : Tablero del TicTacToe

Inicialmente el tablero está vacío, y los jugadores tienen asignada un tipo de ficha, uno fichas circulares (O) y el otro fichas en forma de cruz (X). Cada jugador tiene un número ilimitado de fichas, y uno de los dos jugadores tendrá el turno inicial.

Durante su turno un jugador debe colocar su ficha en una de las casillas vacías del tablero, tras esto el turno pasa al otro jugador que realizará la misma operación, repitiéndose hasta que uno de los dos jugadores obtenga la victoria o no se puedan colocar más fichas por estar el tablero lleno.

Un jugador vence cuando consigue tener tres de sus fichas formando una línea recta, bien porque consiga uno de los cuatro bordes del cuadrado, o bien porque consiga una de las dos diagonales. En ese momento se detiene el juego dándose por finalizada la partida.

Si al llenarse el tablero ninguno de los dos jugadores ha conseguido colocar tres fichas en línea recta, la partida finaliza en tablas.

3.2 Toma de decisiones iniciales

Se tomarán a continuación una serie de decisiones a partir de la especificación anterior, con estas decisiones se desarrollará una primera aproximación al juego que posteriormente se irá mejorando, para mostrar al lector las diferentes posibilidades de CORBA.

El juego va a ser una aplicación distribuida, dónde encontraremos un servidor que será el encargado de llevar el estado del juego, y dos clientes, uno por jugador.

Los jugadores se conectarán al servidor al comienzo, de tal forma que cuando un jugador desee realizar un movimiento se le informará al servidor para que modifique el estado.

Las aplicaciones de los clientes deberán comprobar el estado del servidor cada cierto tiempo de tal forma que puedan avisar al usuario cuando se produzca algún cambio, como el cambio de turno.

El servidor admitirá la conexión de dos jugadores, momento a partir del cual comenzará la partida, cuando la partida haya finalizado el servidor quedará inutilizable.

3.3 Creación del IDL

A partir de la especificación del juego y de las primeras decisiones tomadas podemos ver que tendremos un único servidor que será el *árbitro* del juego.

Este servidor debe proporcionar atributos para que los clientes puedan conocer el estado del juego, es decir las posiciones de las fichas en el tablero, si ha finalizado el juego (en cuyo caso quién ha vencido, o si hay tablas), y en caso de estar jugando de quién es el turno.

Además de lo anterior, el servidor debe disponer de operaciones que permitan a los clientes poder darse de alta en la partida, y colocar una ficha cuando sea su turno.

Todo lo anterior se incluirá en un módulo (*TicTacToe*). En dicho módulo encontraremos las definiciones de los tipos necesarios (*ficha*, *estadoJuego*, *tipoTablero*), de las excepciones que se pueden producir (*CasillaOcupada*,

YaHayDosJugadores, *JuegoParado*) y de la interface *Juego* con los atributos y métodos comentados anteriormente:

TicTacToe.idl

```
1. module TicTacToe {
2.     enum ficha {circulo,cruz,vacia};
3.     typedef ficha tipoTablero[3][3];
4.     enum estadoJuego {victoriaCirculo, victoriaCruz, tablas, jugando,
                  inicializando};

5.     exception CasillaOcupada {};
6.     exception YaHayDosJugadores {};
7.     exception JuegoParado {};

8.     interface Juego {

9.           readonly attribute tipoTablero tablero;
10.          readonly attribute ficha turno;
11.          readonly attribute estadoJuego estado;

12.          tipoTablero anadirJugador(out ficha usar)
                  raises(YaHayDosJugadores);

13.          tipoTablero ponerFicha(in short x, in short y)
                  raises(JuegoParado, CasillaOcupada);
14.    };
15.};
```

Como se puede ver en el código fuente se han definido dos enumerado en las líneas 2 y 4. Una *ficha* indica lo que puede contener una posición del tablero, es decir puede haberse puesto un circulo, una cruz, o estar todavía vacía. El *estadoJuego* indica en qué posición del juego se encuentra la partida:

- Victoria de las fichas circulo
- Victoria de las fichas cruz
- Fin de la partida por tablas
- La partida se está jugando
- Se espera que se unan los jugadores para comenzar la partida

En la línea 3 se define el tipo tablero de juego como una array de fichas, una por posición del tablero.

Se definen tres excepciones posibles durante el juego en las líneas 5, 6 y 7:

- Se intenta poner una ficha y la casilla está ocupada por otra ficha.
- Un jugador intenta incluirse en la partida pero esta ya está completa.
- Se intenta poner una ficha pero el juego no ha comenzado, o ya ha finalizado.

En la interfaz *Juego* se definen tres atributos en las líneas 9, 10 y 11:

- *tablero:* con el estado actual del tablero.
- *turno:* indica a quién le toca (sólo puede valer *circulo* o *cruz*).
- *estado:* permite saber el estado actual de la partida.

Además de los atributos se definen dos operaciones que se pondrán a disposición de los clientes, éstas vienen especificadas en las líneas 12 y 13:

- *anadirJugador:* Le permite a un jugador intentar unirse a la partida, si esta aún no ha comenzado. Devuelve el estado inicial del tablero y la ficha que usará dicho jugador.
- *ponerFicha:* le permite a un jugador poner una ficha en su turno. Se debe indicar la posición del tablero donde se pondrá la ficha, la ficha que se pondrá dependerá del turno actual.

Una vez escrito el fichero *TicTacToe.idl* debe grabarse en un directorio llamado *TicTacToe* que se encuentre en un directorio referenciado por la variable de entorno CLASSPATH. Por ejemplo: *"CLASSPATH=C:\Java"* y el fichero: *"C:\Java\TicTacToe\TicTacToe.idl"*.

Utilizando la herramienta *idltojava* (que puede ser obtenida en el sitio Web de Sun de forma gratuita "*http://java.sun.com/*") obtenemos automáticamente un conjunto de clases de acuerdo con la transformación IDL a Java descrita por el OMG.

El comando que genera las clases Java sería:

> idltojava –fno-cpp –j .. TicTacToe.idl

El parámetro "–fno-cpp" evita que el programa busque un precompilador de C, en caso de no poner este flag sería necesario tener instalado el MS Visual C++.

El parámetro "–j .." obliga a la herramienta a generar las clases en el mismo directorio, en otro caso crearía un nuevo directorio TicTacToe.

Las clases generadas son:

CasillaOcupada.java
CasillaOcupadaHelper.java
CasillaOcupadaHolder.java
estadoJuego.java
estadoJuegoHelper.java
estadoJuegoHolder.java
ficha.java
fichaHelper.java
fichaHolder.java
Juego.java
JuegoHelper.java
JuegoHolder.java
JuegoParado.java
JuegoParadoHelper.java
JuegoParadoHolder.java
TicTacToe.idl
tipoTableroHelper.java
tipoTableroHolder.java
YaHayDosJugadores.java
YaHayDosJugadoresHelper.java
YaHayDosJugadoresHolder.java
_JuegoImplBase.java
_JuegoStub.java

Por cada excepción, y enumerado se han generado tres clases, aquella cuyo nombre coincide con lo definido, y una clase *Helper* y otra *Holder* como ya se explicó en el apartado **2.4**.

Para el tipo *tipoTablero* no se genera una clase, sino que se sustituye por un array de 3x3 fichas en cada aparición del tipo. Sin embargo sí se generan las clases *Helper* (siempre generada para cualquier tipo) y la clase *Holder* que se genera sólo si el tipo es un array, ya que si no lo es ya hay un *Holder* (hay uno por cada tipo simple, y se genera uno para los demás).

Finalmente para la interfaz *Juego* se generan cinco clases:

- *Juego.java:* Es una interfaz Java que implementa la interfaz IDL.
- *JuegoHelper.java* y *JuegoHolder.java*
- *_JuegoImplBase.java:* Esta clase abstracta implementa la interfaz *Juego*, y es el *skeleton* del servidor, que le permite la recepción de invocaciones remotas. Esta clase debe ser extendida para implementar el servidor.
- *_JuegoStub:* Esta clase es el *stub* del cliente, que permitirá las invocaciones de los clientes.

3.4 Implementación del servidor

A partir de las clases obtenidas de la ejecución de la herramienta *idltojava* se desarrolla una clase que implemente al servidor del juego. A esta clase se le llamará *JuegoImpl*.

Nota: Una regla para el nombrado de las implementaciones de las clases servidoras será <*nombre_clase*>*Impl*.

La clase *JuegoImpl* debe heredar de *_JuegoImplBase*, y debe implementar las operaciones definidas en la interfaz *Juego*.

El código de la clase puede verse a continuación:

JuegoImpl.java

```
package TicTacToe;

public class JuegoImpl extends _JuegoImplBase {
              // Atributos de la interfaz IDL
       protected TicTacToe.ficha[][] tablero;
       protected TicTacToe.ficha turno;
       protected TicTacToe.estadoJuego estado;

    // Atributo interno para saber el número de jugadores en la partida
       protected int numeroJugadores = 0;

              // Constructor que inicializa el estado del juego
       public JuegoImpl() {
              super();

              inicializaTablero();
              estado=estadoJuego.inicializando;
              turno=ficha.circulo;
       }

              // Métodos para el acceso al estado del juego
       public TicTacToe.ficha[][] tablero() {
```

```
        return tablero;
    }

    public TicTacToe.ficha turno() {
        return turno;
    }

    public TicTacToe.estadoJuego estado() {
        return estado;
    }

        // Implementación del primer método definido en la interfaz
    public TicTacToe.ficha[][] anadirJugador
                            (TicTacToe.fichaHolder usar)
        throws TicTacToe.YaHayDosJugadores {
        if (numeroJugadores==2) {
            System.out.println("ERROR :Jugador no Añadido,
                                        juego completo");
            throw new TicTacToe.YaHayDosJugadores();
        }

        if (numeroJugadores==0) {
            usar.value=ficha.circulo;
            System.out.println("Jugador 1 Añadido");
        }
        else {
            System.out.println("Jugador 2 Añadido");
            usar.value=ficha.cruz;
        }

        numeroJugadores++;

        if (numeroJugadores==2) {
            estado = estadoJuego.jugando;
            turno = ficha.circulo;
        }

        return tablero;
    }
        // Implementación del segundo método de la interfaz
    public TicTacToe.ficha[][] ponerFicha(short x, short y)
        throws TicTacToe.JuegoParado, TicTacToe.CasillaOcupada {

        if (x<0 || x>2 || y<0 || y>2)
            throw new CasillaOcupada();

        if (!tablero[x][y].equals(ficha.vacia))
            throw new CasillaOcupada();

        if (!estado.equals(estadoJuego.jugando))
            throw new JuegoParado();

        tablero[x][y]=turno;

        ficha f=vencedor();

        if (f.equals(ficha.cruz)) {
            estado=estadoJuego.victoriaCruz;
        }
        else if (f.equals(ficha.circulo)) {
            estado=estadoJuego.victoriaCirculo;
        }
        else {
            if (tablerolleno())
                estado=estadoJuego.tablas;
            else {
                if (turno.equals(ficha.cruz))
                    turno=ficha.circulo;
```

```
                        else
                                turno=ficha.cruz;
                        }
                }

                return tablero;
        }

        // Método auxiliar que inicializa el tablero dejandolo vacío
        private void inicializaTablero() {
                tablero = new ficha[3][3];
                for (int i=0; i<3; i++)
                        for (int j=0; j<3; j++)
                                tablero[i][j]=ficha.vacia;
        }

        // Método privado para ver si uno de los dos jugadores ha vencido
        private ficha vencedor() {
                ...
        }

                // Comprueba si se ha llenado el tablero de fichas
        private boolean tablerolleno() {
                for (int i=0;i<3;i++)
                        for(int j=0;j<3;j++)
                                if (tablero[i][j].equals(ficha.vacia))
                                        return false;
                return true;
        }
}
```

Como puede verse se introducirá dentro de esta clase un atributo por cada atributo que encontráramos en el IDL. Según el modificador que acompañara al atributo encontraremos:

- **Ninguno:** Aparecerán dos métodos con el mismo nombre del atributo, uno sin parámetros que devolverá el atributo de la clase, y otro con un parámetro que contendrá en nuevo valor del atributo de la clase.

- **Readonly:** Únicamente tendremos el método que devuelve el atributo de la clase.

En el caso del atributo *turno* de la interfaz *Juego*, sólo se genera un método para su acceso, ya que tiene el modificador *readonly*.

Esto no debe implementarse necesariamente así, puede calcularse el valor devuelto por los métodos a partir de otros atributos, o como una constante, pero es recomendable utilizar un atributo por cada atributo en la interfaz, ya que hace más legible el código.

A continuación encontramos las operaciones definidas en el IDL a las que debe darse una implementación de acuerdo a la especificación realizada.

Como se observa en la interfaz Java denominada *Juego* generada a partir de la interfaz IDL, para implementar las operaciones hay que tener en cuenta si los parámetros son de entrada o de salida:

- **in:** El tipo del parámetro es idéntico al especificado en el IDL.

- **out, inout:** El tipo del parámetro es *<tipobase>Holder* donde *tipobase* es el especificado en el IDL.

La clase *JuegoImpl* implementa además una serie de métodos privados para la utilización por el resto de los métodos:

- *inicializaTablero:* El método para la inicialización del tablero.
- *vencedor:* Devuelve la ficha del jugador que ha vencido, si alguno lo ha hecho.
- *tableroLleno:* Método para la comprobación de la terminación de la partida.

Finalmente se debe crear una clase aplicación que sea la encargada de crear una nueva partida y situarla a la escucha de posibles clientes, dándola de alta en el ORB. A esta clase se le llamará *JuegoServer*, de tal forma que para lanzar el servidor el comando será:

> java TicTacToe.JuegoServer

Para esta operación se utilizan las clases que se proporcionan con las JDK 1.2, en el paquete *org.omg.CORBA.**.

El código de la clase *JuegoServer* es el siguiente:

JuegoServer.java

```
package TicTacToe;

1. import java.io.*;
2. import org.omg.CORBA.*;

3. public class JuegoServer {

4.     public static void main(String[] args) {
5.   try {
6.           ORB orb = ORB.init(args,null);
7.           Juego objJuego = new JuegoImpl();
8.           orb.connect(objJuego);

9.           PrintWriter salida = new PrintWriter (new FileWriter (
                                                  new File ("IOR.txt")));
10.          salida.print(orb.object_to_string(objJuego));
11.          salida.flush();
12.          salida.close();

13.          System.out.println("Juego listo");

// Espero a que se pulse enter para terminar la ejecución del servidor
14.          try {
15.                BufferedReader teclado;
16.                teclado=new BufferedReader
                            (new InputStreamReader(System.in));
17.                String linea=teclado.readLine();
18.          } catch (IOException ex) {}

19.          System.out.println("Finalizado el juego");
20.      } catch (IOException ex) {
21.          System.err.println("No se puede escribir el IOR
                                           en el fichero");
22.          ex.printStackTrace();
```

```
23.       } catch (Exception ex) {
24.           System.err.println("No se puede crear el servidor");
25.           ex.printStackTrace();
26.       }
27.     }
28. }
```

En primer lugar se crea un nuevo ORB (Línea 6) pasándole los parámetros de la aplicación y sin propiedades, ésta será la inicialización estándar, a no ser que se quiera cambiar alguna propiedad del ORB, en cuyo caso la inicialización debería ser:

```
Properties p = new Properties();
p.put("Nombre Propiedad", "Valor Propiedad");
ORB orb = ORB.init(args, p);
```

Con las siguientes posibilidades:

Propiedades estándar, existentes en todo ORB:

- *org.omg.CORBA.ORBClass:* Indica cual es la clase que implementa el ORB. En el caso de las JDK 1.2 la clase por defecto es *org.omg.CORBA.iiop.ORB*.
- *org.omg.CORBA.ORBSingletonClass:* Igual que el anterior pero para el caso especial de que se usase el método *init* sin parámetros.

Propiedades específicas del ORB de las JDK 1.2:

- *org.omg.CORBA.ORBInitialHost:* El nombre de la máquina que ejecuta los servicios iniciales, como el servicio de nombres. Por defecto la máquina local para aplicaciones, y el servidor Web para applets.
- *org.omg.CORBA.ORBInitialPort:* El puerto en el que escucha el servicio de nombres. Por defecto el 900

A continuación se crea un nuevo objeto de la clase *JuegoImpl* (Línea 7) que será la partida a crear, y se conecta este objeto con el ORB (Línea 8), de tal forma que el último conozca la existencia del primero, para pasarle las peticiones que reciba.

En este momento aparece un problema, ¿Cómo se conectarán los clientes al objeto servidor?. Para que un cliente pueda acceder al objeto servidor debe obtener una referencia a éste, y uno de los grandes problemas de CORBA es cómo obtener dicha referencia.

Posibles soluciones son las aportadas en este apartado con el IOR y en el apartado **3.8** con el servicio de nombres, otras posibles soluciones son publicar el IOR en un servidor Web, y que el cliente obtenga este a través de una petición a dicho servidor.

Las dos soluciones que se implementarán será el paso del IOR y la utilización del servicio de nombres. Ahora se explicará la primera técnica, y en el apartado **3.8** se explicará e implementará la segunda solución.

Cada objeto que se conecta al ORB (es decir se da de alta en el ORB, Línea 8) recibe un identificador único llamado IOR, este identificador es único dentro del ORB y

entre cualquier par de ORB's, de tal forma que si un cliente obtiene el IOR del servidor ya podrá acceder a cualquiera de los servicios que este provea.

El IOR de un objeto se puede obtener con un método de la interfaz del ORB: *object_to_string* (Ver línea 10 del código), de esta forma podemos guardar el IOR en un fichero, llevar el fichero hasta el cliente (Con un disco, por ftp, web, ...) para que lo lea y obtenga de nuevo la referencia al servidor utilizando el método *string_to_object* del ORB.

Como ejemplo podemos ver el IOR del objeto *juego*, instancia de la clase *JuegoImpl* que representa a la interfaz *Juego*, en una máquina cuyo nombre de DNS es *servidor*:

IOR:000000000000001849444c3a546963546163546f652f4a7565676f3a312e30
000000001000000000000003000010000000000097365727669646f720000040d00000
018afabcafe00000002928220ab000000080000000000000000

Como puede verse es un conjunto de valores hexadecimales. Si pasamos los valores hexadecimales a caracteres obtenemos:

```
IDL:TicTacToe/Juego:1.0
servidor
```

Esto identifica la interfaz del objeto servidor que se está ejecutando (*TicTacToe/Juego*) y la máquina de Internet en la que lo está haciendo (*servidor*). Además de esta información, la parte final del IOR no se traduce en caracteres, su función es la de identificar el objeto dentro de la máquina.

Lo que hace el servidor es almacenar el IOR en el fichero *IOR.txt* (Líneas 9, 10, 11 y 12), tras lo cual se queda esperando una pulsación de teclado para finalizar. Esto último lo hace porque si el programa termina la máquina virtual java se descargará de memoria eliminando así al servidor.

3.5 Implementación del cliente

La interfaz del cliente se ha realizado utilizando las librerías *Swing* que acompañan a las JDK 1.2.

El código de la clase *VentanaAplic*, que constituye la aplicación cliente, es el siguiente:

VentanaAplic.java

```
1. package TicTacToe;

2. import java.awt.*;
3. import java.awt.event.*;
4. import javax.swing.*;
5. import org.omg.CORBA.*;
6. import java.io.*;

7. public class VentanaAplic extends JFrame {
```

```
8.   Label estado = new Label();
9.   Label error = new Label();
10.  Label casillas[][] = {{new Label(), new Label(), new Label()},
11.                {new Label(), new Label(), new Label()},
12.                {new Label(), new Label(), new Label()}};

13.    Juego juego;
14.    Timer tiempo;
15.    ficha turnoAntes;
16.    ficha suFicha;
17.    estadoJuego estadoAntes;

18.    public VentanaAplic(String args[]) {
                  ...
19. try {
20.          ORB orb = ORB.init(args,null);
21.          File ficheroEntrada = new File("IOR.txt");
22.          FileReader entrada = new FileReader( ficheroEntrada);
23.          char IOR[] = new char[(int)ficheroEntrada.length()];
24.          if (entrada.read(IOR,0, IOR.length)
                                    <ficheroEntrada.length()) {
25.                System.err.println("No se ha podido leer el IOR");
26.                estado.setText("ERROR");
27.                return;
28.          }
29.          org.omg.CORBA.Object objJuego =
                          orb.string_to_object(new String(IOR));
30.          juego = JuegoHelper.narrow(objJuego);
31.          fichaHolder aux = new fichaHolder();
32.          turnoAntes = juego.turno();
33.          estadoAntes = juego.estado();
34.          ficha tabl[][] = juego.anadirJugador(aux);
35.          suFicha=aux.value;
36.          dibujarTablero(tabl);
37.          tiempo=new Timer(1000,new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
            tiempoTranscurrido(e);
            }
            });
38.           tiempo.start();
39.    } catch (YaHayDosJugadores ex) {
40.          System.err.println("Ya hay dos jugadores");
41.          estado.setText("ERROR");
42.    } catch (IOException ex) {
43.          System.err.println("No se ha podido leer el IOR");
44.          estado.setText("ERROR");
45.    } catch (Exception ex) {
46.          System.err.println("No se ha podido conectar
                                    con el servidor");
47.          estado.setText("ERROR");
48.          ex.printStackTrace();
49.    }
50.  }

51.    private void jbInit() throws Exception  {
            ...
52.    }

53.    private void dibujarTablero(ficha[][] tablero) {
54.          for (int i=0;i<3;i++)
55.                for (int j=0;j<3;j++) {
56.                      if (tablero[i][j]==ficha.cruz)
57.                            casillas[i][j].setText(" X");
58.                      else if (tablero[i][j]==ficha.circulo)
59.                            casillas[i][j].setText(" O");
60.                      else
61.                            casillas[i][j].setText("");
62.                      casillas[i][j].invalidate();
```

```
63.                 }
64.           if (estadoAntes.equals(estadoJuego.victoriaCirculo)) {
65.                 if (suFicha.equals(ficha.circulo))
66.                       estado.setText("Ha vencido");
67.                 else
68.                       estado.setText("Ha perdido");
69.           }
70.           else if (estadoAntes.equals(estadoJuego.victoriaCruz)) {
71.                 if (suFicha.equals(ficha.cruz))
72.                       estado.setText("Ha vencido");
73.                 else
74.                       estado.setText("Ha perdido");
75.           }
76.           else if (estadoAntes.equals(estadoJuego.tablas)) {
77.                 estado.setText("Tablas");
78.           }
79.           else if (estadoAntes.equals(estadoJuego.inicializando)) {
80.                 estado.setText("Espere jugador");
81.           }
82.           else {
83.                 if (turnoAntes.equals(suFicha))
84.                       estado.setText("Su turno");
85.                 else
86.                       estado.setText("Espere su turno");
87.           }
88.    }

89.    public void tiempoTranscurrido(ActionEvent e) {
90.           if (!estadoAntes.equals(juego.estado())) {
91.                 estadoAntes=juego.estado();
92.                 dibujarTablero(juego.tablero());
93.                 if (!estadoAntes.equals(estadoJuego.jugando))
94.                       tiempo.stop();
95.           }
96.           else if (!turnoAntes.equals(juego.turno())) {
97.                 turnoAntes=juego.turno();
98.                 dibujarTablero(juego.tablero());
99.           }
100.  }

101.   protected void processWindowEvent(WindowEvent e) {
       ...
102.   }

103.   void label4_mouseClicked(MouseEvent e,int i,int j) {
104.         if (!estadoAntes.equals(estadoJuego.jugando))
105.               return;

106.         try {
107.               error.setText("");
108.               dibujarTablero(juego.ponerFicha
                                          ((short)i, (short)j));
109.         } catch (TicTacToe.JuegoParado ex) {
110.               error.setText("No puede colocar fichas");
111.         } catch (CasillaOcupada ex) {
112.               error.setText("Mala posición");
113.         }
114.   }
115. }
```

Lo primero que tiene que hacer el cliente es obtener una referencia al objeto servidor para intentar darse de alta en el juego. Para esto crea un ORB (Línea 20), y lee del fichero que ha creado el servidor el valor del IOR (Líneas 21, 22, 23 y 24), para después convertirlo en una referencia al objeto servidor, utilizando el método *string_to_object* de la clase ORB (Línea 29).

El problema que se plantea es que el método *string_to_object* devuelve un objeto de la clase *org.omg.CORBA.Object*, que necesitamos convertir a un objeto de tipo *Juego*. Para ello se utiliza la clase *JuegoHelper* que tiene el método *narrow*, al cual se le pasa un objeto de la clase CORBA.Object y devuelve un objeto *Juego* (Línea 30).

Una vez obtenida la referencia al objeto *Juego* se debe dar de alta el jugador (Línea 34), siempre comprobando los posibles errores producidos en el servidor, indicados estos con excepciones. Para dar de alta a un jugador se utiliza el método *anadirJugador* de la interfaz *Juego*, este método tiene como parámetro la ficha que el servidor ha asignado al jugador, pero este parámetro fue definido en el IDL como *out* por lo que debe ser del tipo *fichaHolder*, la construcción de un objeto de este tipo puede verse en la Línea 31, en la que se usa el constructor sin parámetros (se definió como sólo de salida por lo que no hay que pasarle nada), si se hubiese definido en el IDL como *inout* se utilizaría un constructor con un parámetro que sería el objeto a enviar al servidor.

Finalmente se inicializa un cronómetro (Líneas 37 y 38) para comprobar posibles cambios en el estado del juego. El método *tiempoTranscurrido* (Línea 89) cada *x* tiempo comprueba el estado del objeto servidor buscando cambios en éste, si se produce algún cambio se lo indica al usuario a través de la interfaz gráfica. Ésta está compuesta por un tablero de tres por tres, y dos etiquetas, una para indicar el estado de la partida y otra para los errores que se produzcan durante la partida, como el intento de poner una ficha en una casilla ocupada.

Cuando se pulsa una casilla con el ratón se avisa al servidor para que modifique el estado de la partida.

La interfaz de la aplicación resultante tendría el aspecto de la figura 6.

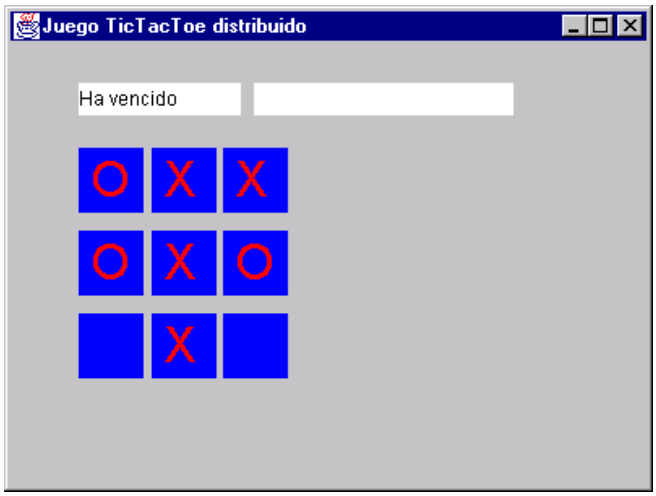

Figura 6 : Ventana del juego TicTacToe

3.6 Implementación de un cliente como un applet

A continuación se implementará el mismo cliente anterior, pero dicha implementación se realizará como un applet, que se llamará *JuegoApplet*.

El problema con el que se encuentran los applets es la implementación de la Máquina Virtual Java 1.2 por parte del explorador, ya que una versión antigua del explorador puede no permitir la ejecución de nuestro applet. Para solucionar este problema se utiliza un programa ofrecido por Sun que modifica una página Web de tal forma que el explorador en vez de utilizar su máquina virtual utilizará la que se tenga instalada en el host. Las máquinas virtuales que se pueden instalar son las llamadas *JRE (Java Runtime Enviroment),* pero como se están utilizando las JDK 1.2 también se deben utilizar las JRE 1.2 (éstas pueden encontrarse en el Web de Sun *java.sun.com*, tanto en un fichero independiente o junto con las JDK1.2).

En primer lugar, se desarrolla la página Web normalmente, llamando al applet con una etiqueta normal del tipo *<APPLET ...>*, una vez que se tiene se lanza el programa *htmlconv12 (java HTMLConverter) ,* ofrecido gratuitamente por Sun, y se le indica al programa cual es la página Web a convertir.

Por ejemplo para el juego del TicTacToe tendríamos una página inicial como sigue:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
<TITLE>
Pagina Web para jugar al Tic Tac Toe
</TITLE>
</HEAD>
<BODY>
Juego TicTacToe distribuido.<BR>
<APPLET
  CODE     = "TicTacToe.JugadorApplet"
  NAME     = "TicTacToe"
  WIDTH    = 400
  HEIGHT   = 300
  HSPACE   = 0
  VSPACE   = 0
  ALIGN    = middle
>
</APPLET>
</BODY>
</HTML>
```

Tras pasarle la página anterior al programa *HTMLConverter* se obtiene la siguiente página:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
<TITLE>
Pagina Web para jugar al Tic Tac Toe
</TITLE>
</HEAD>
<BODY>
Juego TicTacToe distribuido.<BR>
<!--"CONVERTED_APPLET"-->
```

```
      <!-- CONVERTER VERSION 1.0 -->
      <OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
      WIDTH = 400 HEIGHT = 300 NAME = "TicTacToe" ALIGN = middle VSPACE = 0
HSPACE = 0  codebase="http://java.sun.com/products/plugin/1.2/jinstall-12-
win32.cab#Version=1,2,0,0">
      <PARAM NAME = CODE VALUE = "TicTacToe.JugadorApplet" >
      <PARAM NAME = NAME VALUE = "TicTacToe" >

      <PARAM NAME="type" VALUE="application/x-java-applet;version=1.2">
      <COMMENT>
      <EMBED type="application/x-java-applet;version=1.2" java_CODE =
"TicTacToe.JugadorApplet" NAME = "TicTacToe" WIDTH = 400 HEIGHT = 300 ALIGN =
middle VSPACE = 0 HSPACE = 0
pluginspage="http://java.sun.com/products/plugin/1.2/plugin-
install.html"><NOEMBED></COMMENT>

      </NOEMBED></EMBED>
      </OBJECT>

      <!--
      <APPLET  CODE = "TicTacToe.JugadorApplet" WIDTH = 400 HEIGHT = 300 NAME
= "TicTacToe" ALIGN = middle VSPACE = 0 HSPACE = 0 >


      </APPLET>
      -->
      <!--"END_CONVERTED_APPLET"-->

      </BODY>
      </HTML>
```

Por otro lado se desarrolla el applet siguiendo la misma estructura que se indica en el apartado 3.5 para la clase *VentanaAplic*. La única diferencia se encuentra en la creación del ORB, en la que se utiliza como primer parámetro del método *init* el propio applet en vez de los parámetros de la aplicación como en el caso anterior. Quedaría pues de la siguiente forma:

```
ORB orb = ORB.init(this,null);
```

En la máquina en la que tengamos el servidor Web (que puede coincidir o no con la máquina donde se esté ejecutando el servidor del TicTacToe) se debe *publicar* la página convertida, y las clases generadas en el paquete *TicTacToe*, y aquellas que utilice nuestro applet y que no se encuentren en las JDK 1.2 (En este caso ninguna). Las clases se colocarán en el servidor en un directorio correspondiendo con el paquete de la clase, dentro del *CODEBASE* que se especifique en la página Web, por ejemplo, si el *CODEBASE* es igual a "*http://www.servidor.es/~jros/*" y tenemos la clase *TicTacToe.Juego*, el archivo en el servidor Apache con una configuración por defecto sería: "*/home/jros/public_html/TicTacToe/Juego.class*".

En la máquina del cliente se deben instalar las JRE 1.2 y un browser (en las pruebas se utilizó Netscape 4.5).

Por otro lado, en otra máquina se debe lanzar el servidor de TicTacToe, y copiar el fichero "IOR.txt" generado por el servidor en el directorio "D:\Java" del cliente, ya que es en este directorio donde lo buscará el applet (Si se desea se puede modificar dicho applet para que lo busque en otro directorio).

Pero aún con todo esto el programa sigue sin funcionar y la causa es el SandBox en el que se ejecutan los applets Java (Modelo de seguridad de Java), donde por defecto no deja ni acceder a un fichero del disco duro local, ni conectar con otra máquina que no sea el servidor Web desde donde se obtuvo el applet. En este caso se desea hacer las dos cosas, por tanto se deben modificar los permisos del SandBox de java.

Para modificar los permisos existe un fichero llamado “java.policy”, que en una máquina con Windows, y una instalación típica de las JRE se encuentra en el directorio “Archivos de programa\JavaSoft\JRE\1.2\lib\security”, al cual se le deben añadir las líneas:

```
grant {
      permission java.security.AllPermission;
};
```

Con lo anterior tendremos la menor seguridad posible, es decir le daremos acceso a todos los applets para acceder a todos los recursos. Se puede (y se debe) tomar una política más restrictiva en la que únicamente se le den acceso a los applets procedentes del servidor Web (incluso se deberían firmar digitalmente para mayor seguridad), y sólo para acceder al fichero “IOR.txt” y para conectarse con el servidor del *TicTacToe*. Para más información sobre seguridad en los applets ver [Jaw98]. Un ejemplo de este tipo de políticas sería añadir en las siguientes líneas sustituyendo a las anteriores:

```
grant CodeBase “http://www.servidor.es/” {
      permission java.io.FilePermission “d:\java\ior.txt”, “read”;
      permission java.net.SocketPermission “servidorConTicTacToe.es”,
      “accept, connect, resolve”;
};
```

Una vez concedidos los permisos necesarios bastaría con lanzar el browser y acceder a la página Web deseada, con lo que aparecerá el cliente de TicTacToe y se añadirá el jugador a la partida.

La arquitectura para la aplicación resultante se puede ver en la figura 7.

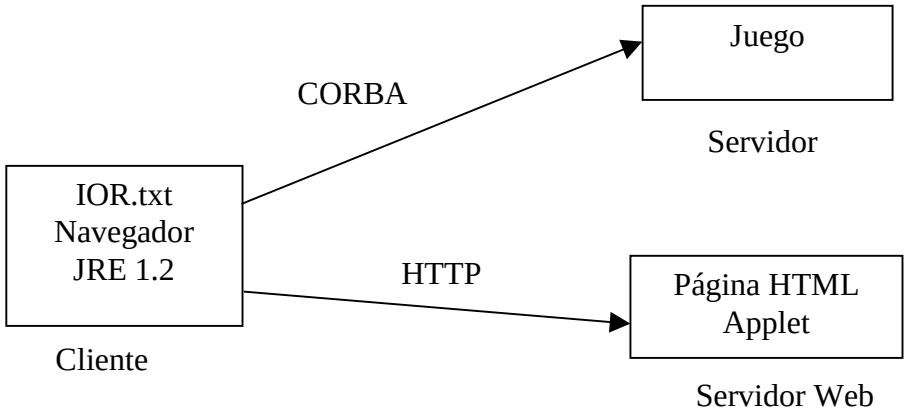

Figura 7 : Arquitectura del TicTacToe

Finalmente el navegador mostraría una ventana como se ve en la figura 8.

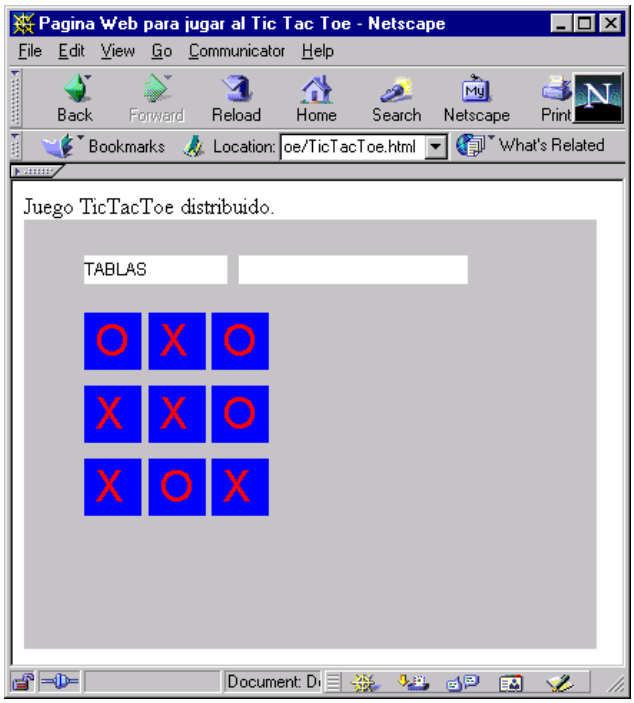

Figura 8: Applet del juego TicTacToe

3.7 Uso de los callback

A partir de este momento se intentará mejorar la implementación del juego, aprovechando algunas de las posibilidades que nos ofrece CORBA.

El polling al servidor para detectar cambios en el estado del juego, provoca un gran tráfico en la red, además de imposibilitar al cliente para realizar otras tareas correctamente, y todo esto sin ser necesario ya que los cambios en el estado del juego no se producen con tanta frecuencia como para tener que preguntar constantemente.

Para solucionar este problema el cliente se convertirá a su vez en un servidor de tal forma que cuando se produzca un cambio en el juego el objeto *Juego* pueda avisar al cliente de este cambio.

Para poder realizar esto hay que modificar el fichero de definiciones de interfaces (IDL) para añadir la interfaz del cliente.

> ***Nota:*** Se llamará cliente al que desempeña la función de jugador en la partida y servidor al que lleva el manejo del juego, pero a partir de este momento todos serán servidores desde el punto de vista de que todos pueden recibir peticiones.

Se va a intentar separar la implementación del cliente de la implementación de la interfaz con el usuario, de tal forma que sea sencillo desarrollar diferentes interfaces gráficas para la misma lógica de aplicación (gráfica, de texto, applet, ...).

En el nuevo fichero IDL se define una interfaz para el cliente, llamada *Jugador*, está interfaz será utilizada tanto por el servidor (para avisar de cambios en el estado de la partida), como por la interfaz gráfica para conocer el estado del juego y para interactuar con dicho estado.

TicTacToe.idl

```
1. module TicTacToe {
2.     enum ficha {circulo,cruz,vacia};
3.     enum estadoJugador {jugando, victoria, tablas, derrota};
4.     typedef ficha tablero[3][3];

5.     exception CasillaOcupada {};
6.     exception JugadorInvalido {};
7.     exception ErrorEnServidor {};
8.     exception YaHayDosJugadores {};

9.     interface Jugador {
10.           readonly attribute boolean meToca;
11.           readonly attribute tablero tab;
12.           readonly attribute estadoJugador estado;
13.           readonly attribute ficha usada;

14.           oneway void turno(in boolean es,in tablero tab);
15.    oneway void fin(in boolean victoria, in boolean tablas,
                       in tablero tab);
16.           void ponerFicha(in short x, in short y)
                     raises(ErrorEnServidor,CasillaOcupada);
17.    };

18.    interface Juego {
19.           readonly attribute tablero tab;

20.           tablero anadirJugador(in Jugador jug,out ficha usar)
                     raises(YaHayDosJugadores);
21.           tablero ponerFicha(in Jugador jug, in short x, in short y)
                     raises(JugadorInvalido,CasillaOcupada);
22.    };
23. };
```

En la interfaz *Jugador* se definen los atributos necesarios para que la interfaz gráfica pueda conocer el estado de la partida (Líneas 10 a 13). Estos atributos son el estado del tablero, la ficha que usa, el estado en el que se encuentra la partida y si le toca jugar. Además se define un método para colocar una nueva ficha en el tablero (Línea 16).

Se definen los atributos anteriores en el *Jugador*, para simplificar la implementación de la interfaz gráfica, que sólo accederá al objeto *Jugador* (que se encuentra en la misma máquina que la interfaz gráfica), y no necesitará acceder al *Juego*.

Las operaciones que utilizará el servidor para avisar de un cambio de estado, *turno* y *fin*, (Líneas 14 y 15) son del tipo *oneway*, de tal forma que una vez que el servidor ha realizado la petición no tiene que esperar a que el cliente le responda.

Por otro lado la interfaz *Juego* modifica las operaciones para que reciban como parámetro adicional el jugador. Con la operación que añade jugadores queda registrado quienes son los jugadores admitidos en la partida y de esta forma se les puede avisar cuando se modifique el estado de la partida, y cuando se coloque una ficha para comprobar que intenta jugar el cliente adecuado, comprobación que antes no se podía realizar, ya que los clientes no estaban identificados.

La implementación del servidor es la siguiente:

JuegoImpl.java

```
1. package TicTacToe;

2. import TicTacToe.*;

3. public class JuegoImpl extends _JuegoImplBase {
4.   private ficha[][] tablero;
5.   private Jugador jug1,jug2;
6.   private estadoJugador estado;
7.   private int numeroServidor;
8.   private ORB orb;
9.   private String IORJug1;
10.  private String IORJug2;
11.  private String IORturno;


12.   public JuegoImpl(ORB orb) {
13.          super();

14.          this.orb = orb;
15.    inicializaTablero();
16.    estado=estadoJugador.jugando;
17.  }

  //readonly attribute tablero tab;
18.  public ficha[][] tab() {
19.    return tablero;
20.  }

21. public ficha[][] anadirJugador(Jugador jug, fichaHolder usar)
    throws TicTacToe.YaHayDosJugadores {

22.    if (jug1==null) {
23.      jug1=jug;
24.      usar.value=ficha.cruz;
25.      IORJug1=orb.object_to_string(jug1);
26.      System.out.println("Jugador 1 Añadido: " + IORJug1);
27.    }
28.    else if (jug2==null) {
29.      jug2=jug;
30.      usar.value=ficha.circulo;
31.      IORJug2=orb.object_to_string(jug2);
32.      System.out.println("Jugador 2 Añadido: " + IORJug2);
33.      jug1.turno(true,tablero);
34.      jug2.turno(false,tablero);
35.     IORturno=IORJug1;
36.    }
37.    else {
38.      System.out.println("ERROR :Jugador no Añadido");
39.      throw new TicTacToe.YaHayDosJugadores();
40.    }

41.    return tablero;
42.  }

43.  private ficha vencedor() {
```

```
            // Sin cambios
44.    }

45.  private boolean tablerolleno() {
            // Sin cambios
46.    }

47.    public ficha[][] ponerFicha(Jugador jug, short x,short y)
48.    throws TicTacToe.JugadorInvalido, CasillaOcupada {

49.    if (!estado.equals(estadoJugador.jugando))
50.      throw new TicTacToe.JugadorInvalido();

51.    if (x<0 || x>2 || y<0 || y>2)
52.      throw new CasillaOcupada();

53.    if (tablero[x][y]!=ficha.vacia)
54.      throw new CasillaOcupada();

55.    if (jug1==null || jug2==null)
56.      throw new TicTacToe.JugadorInvalido();

57.          String IORJug = orb.object_to_string(jug);

58.    if (!IORJug.equals(IORturno))
59.      throw new TicTacToe.JugadorInvalido();

60.    if (jug.usada().equals(jug1.usada()))
61.      tablero[x][y]=ficha.cruz;
62.    else
63.      tablero[x][y]=ficha.circulo;

64.    ficha f=vencedor();

65.    if (f.equals(ficha.cruz)) {
66.      jug1.fin(true,false,tablero);
67.      jug2.fin(false,false,tablero);
68.      estado=estadoJugador.victoria;
69.    }
70.    else if (f.equals(ficha.circulo)) {
71.      jug1.fin(false,false,tablero);
72.      jug2.fin(true,false,tablero);
73.      estado=estadoJugador.derrota;
74.    }
75.    else {
76.      if (tablerolleno()) {
77.        jug1.fin(false,true,tablero);
78.        jug2.fin(false,true,tablero);
79.        estado=estadoJugador.tablas;
80.      }
81.      else {
82.        if (jug.usada().equals(jug1.usada())) {
83.          jug2.turno(true,tablero);
84.          jug1.turno(false,tablero);
85.                  IORturno=IORJug2;
86.        }
87.        else {
88.          jug1.turno(true,tablero);
89.          jug2.turno(false,tablero);
90.                  IORturno=IORJug1;
91.        }
92.      }
93.    }

94.    return tablero;
95.  }

96. }
```

En este caso el servidor avisa a los clientes cuando se produce cualquier modificación en el estado de la partida (Líneas 33, 34, 66, 67, 71, 72, 77, 78, 83, 84, 88 y 89), y para ello almacena dos atributos, *jug1* y *jug2*, que son las referencias a dichos jugadores.

Para comprobar que realmente el jugador que intenta poner la ficha es a quien le toca, se almacena de cada jugador el IOR (ya que este identificador es único) (Líneas 25 y 31), además se guarda el IOR del jugador al que le toca en el atributo *IORturno* (Líneas 35, 85 y 90). Cuando se hace una llamada al método *ponerFicha* se comprueba que el jugador (Pasado como parámetro) corresponde al que tiene el turno (Línea 58) provocando una excepción en caso contrario (Línea 59).

La clase *JuegoServer* no sufre ninguna modificación.

En el lado del cliente se produce la ya comentada división entre el servidor que maneja al jugador, y la interfaz gráfica que interactua con este servidor realizando las peticiones y reflejando los cambios de estado.

Igual que no se deseaba que el cliente preguntara cada cierto tiempo al servidor por su estado para comprobar si éste había cambiado, tampoco se desea que la interfaz gráfica pregunte cada cierto tiempo al jugador para ver si cambia el estado. Ambos se encuentran ejecutándose en la misma máquina virtual, con lo que no se necesitan llamadas remotas para su comunicación, por este motivo se le puede pasar al cliente el objeto que implementa la interfaz gráfica para que le avise de cambios de estado sin necesidad de implementar la interfaz gráfica como un servidor CORBA.

El problema está en que como no se desea hacer dependiente de ningún entorno al cliente no se le puede pasar un objeto de ninguna interfaz gráfica específica, por lo que se crea una interfaz (*JugadorInter*) que debe implementar cualquier entorno. Esta interfaz está formada por un método que será el invocado (llamado desde el objeto *jugador*) cuando se produzca un cambio en el estado.

En la figura 9 podemos ver la arquitectura con la que quedaría la ejecución del programa TicTacToe.

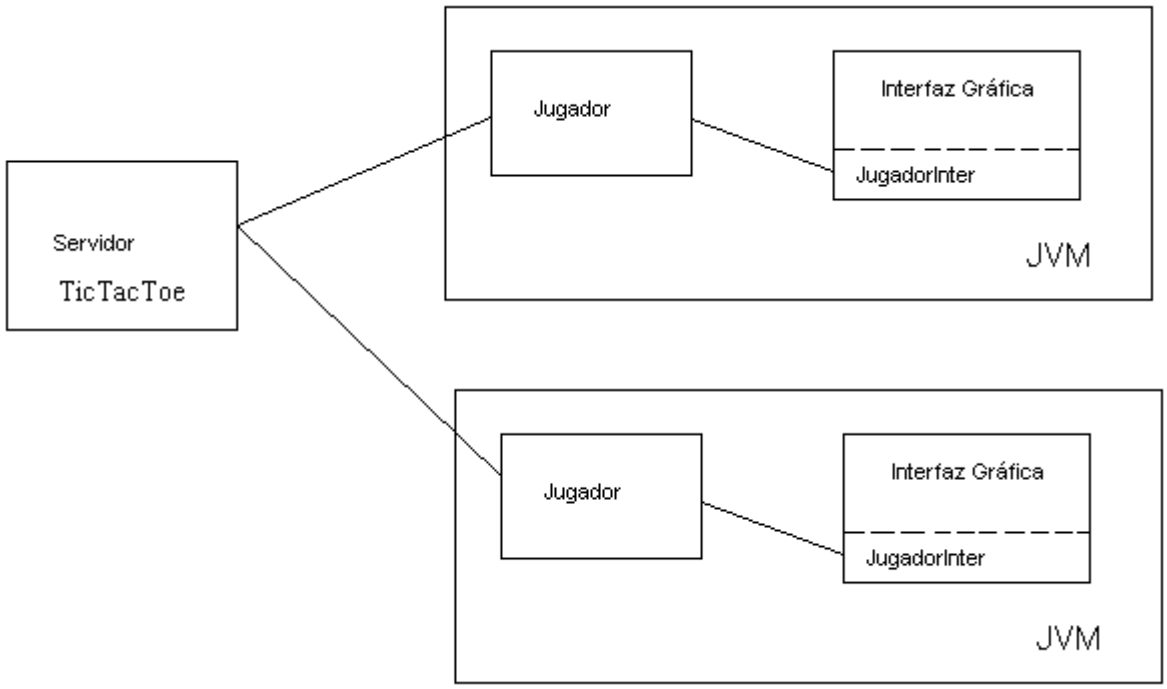

Figura 9 : Estructura del TicTacToe

El código para la interfaz *JugadorInter* es el siguiente:

JugadorInter.java

```
1. package TicTacToe;

2. public interface JugadorInter {
3.   public void estadoCambiado(estadoJugador est,boolean meToca);
4. }
```

El método *estadoCambiado* será llamado por el objeto *jugador* cuando se produzca algún cambio en el estado, de tal forma que la interfaz gráfica refleje el cambio.

El código para la implementación del jugador es:

JugadorImpl.java

```
1. package TicTacToe;

2. import TicTacToe.*;
3. import java.util.*;
4. import java.io.*;

5. public class JugadorImpl extends _JugadorImplBase {
6.     private boolean meToca;
7.     private ficha[][] tablero;
8.     private Juego juego;
9.     private estadoJugador estado;
10.    private ficha usada;
11.    private JugadorInter jugador;

12.    public JugadorImpl(org.omg.CORBA.ORB orb,JugadorInter jug)
                    throws TicTacToe.YaHayDosJugadores {
```

```
13.           super();

14.           fichaHolder fh=new fichaHolder();
15.           meToca=false;
16.       try {
17.           File ficheroEntrada = new File("D:\\Java\\IOR.txt");
18.           FileReader entrada = new FileReader( ficheroEntrada);
19.           char IOR[] = new char[(int)ficheroEntrada.length()];
20.           if                                    (entrada.read(IOR,0,
IOR.length)<ficheroEntrada.length()) {
21.                   System.err.println("No se ha podido leer el IOR");
22.           }
23.     org.omg.CORBA.Object objJuego=
                              orb.string_to_object(new String(IOR));
24.           juego = JuegoHelper.narrow(objJuego);

25.           tablero=juego.anadirJugador(this,fh);
26.       } catch (Exception ex) {
27.                   System.err.println("Error   al   conectar   con   el
servidor");
28.       }

29.           usada=fh.value;
30.           estado=estadoJugador.jugando;
31.           jugador=jug;
32.    }

       // oneway void turno(in tablero tab);
33.    public void turno(boolean es,ficha[][] tab) {
34.           tablero=tab;
35.           meToca=es;
36.           jugador.estadoCambiado(estado,meToca);
37.    }

       //readonly attribute boolean meToca;
38.    public boolean meToca() {
39.           return meToca;
40.    }

       //readonly attribute estadoJugador estado;
41.    public estadoJugador estado() {
42.           return estado;
43.    }

       //readonly attribute tablero tab;
44.    public ficha[][] tab() {
45.           return tablero;
46.    }

      //oneway void fin(in boolean victoria,in boolean tablas,in tablero tab);
47.    public void fin(boolean victoria, boolean tablas,ficha[][] tab) {
48.           if (victoria)
49.                  estado=estadoJugador.victoria;
50.           else if (tablas)
51.                  estado=estadoJugador.tablas;
52.           else
53.                  estado=estadoJugador.derrota;
54.           tablero=tab;
55.           jugador.estadoCambiado(estado,meToca);
56.    }

       //void ponerFicha(in short x, in short y)
              //raises(ErrorEnServidor,CasillaOcupada);
57.    public void ponerFicha(short x,short y)
              throws TicTacToe.ErrorEnServidor, CasillaOcupada {

58.           try {
```

```
59.                 tablero=juego.ponerFicha(this,x,y);
60.           } catch (TicTacToe.JugadorInvalido e) {
61.                 throw new TicTacToe.ErrorEnServidor();
62.           }
63.    }

       //readonly attribute ficha usada;
64.    public ficha usada() {
65.           return usada;
66.    }
67. }
```

Como se puede ver se ha trasladado la funcionalidad que antes se encontraba en la interfaz gráfica, al objeto *jugador*, de tal forma que se simplifica la realización de una nueva interfaz.

En el constructor del objeto (Línea 12) se recibe un parámetro del tipo *JugadorInter* que corresponderá con la interfaz gráfica, a la que hay que avisar de las modificaciones del estado (Líneas 36 y 55).

En este caso no es necesario conectar el objeto con el ORB (*ORB.connect(obj)*) como se hacía en el caso de *JuegoImpl*, ya que al pasar el objeto como parámetro a un servidor remoto se realiza dicha conexión implícitamente.

La implementación de una interfaz gráfica se ve sumamente simplificada. La siguiente implementación se basa en una interfaz de texto muy simple, las implementaciones correspondientes para una interfaz gráfica y un applet pueden verse en el apéndice.

JugadorTextoAplic.java

```
1. package TicTacToe;

2. import org.omg.CORBA.*;
3. import java.util.*;
4. import java.io.*;

5. public class JugadorTextoAplic implements JugadorInter {
6.   private TicTacToe.Jugador jugador;
7.   private BufferedReader teclado;

8.   static public void main(String[] args) {
9.     (new JugadorTextoAplic(args)).jugar();
10.  }

11.  public JugadorTextoAplic(String[] args) {
12.    try {
13.      ORB orb = ORB.init(args,null);
14.      jugador = new JugadorImpl(orb,this);
15.      teclado = new BufferedReader(new InputStreamReader(System.in));
16.    }
17.    catch (Exception e) {
18.      e.printStackTrace();
19.    }
20.  }

21.  private void dibujarTablero() {
22.    ficha f=jugador.usada();
23.    ficha[][] tablero = jugador.tab();

24.    System.out.println();
25.    for (int i=0;i<3;i++) {
```

```
26.      for (int j=0;j<3;j++)
27.        {
28.      if (j!=0)
29.        System.out.print("|");
30.          if (tablero[i][j]==ficha.cruz)
31.            System.out.print(" X ");
32.          else if (tablero[i][j]==ficha.circulo)
33.            System.out.print(" O ");
34.          else
35.            System.out.print("   ");
36.        }
37.      System.out.println();
38.    }
39.  }

40.  public void estadoCambiado(estadoJugador est,boolean meToca) {
41.    dibujarTablero();
42.    if (est.equals(estadoJugador.jugando)) {
43.      if (meToca) {
44.    String entradaX, entradaY;
45.    int valorX, valorY;
46.        System.out.println("Su turno");
47.    while (true) {
48.      try {
49.        System.out.print("Coordenada X (1-3): ");
50.        entradaX = teclado.readLine();
51.        valorX = Integer.parseInt(entradaX);
52.//      if (valorX<1 || valorX>3) continue;
53.        System.out.print("Coordenada Y (1-3): ");
54.        entradaY = teclado.readLine();
55.        valorY = Integer.parseInt(entradaY);
56.//      if (valorY<1 || valorY>3) continue;
57.            jugador.ponerFicha((short)(valorY-1), (short)(valorX-1));
58.            dibujarTablero();
59.        break;
60.      } catch(NumberFormatException ex) {
61.        continue;
62.          } catch (TicTacToe.ErrorEnServidor ex) {
63.            System.out.println("El servidor ha fallado");
64.          } catch (CasillaOcupada ex) {
65.            System.out.println("Casilla Ocupada, o inexistente");
66.          } catch (IOException ex) {
67.        ex.printStackTrace();
68.      }

69.    }
70.      }
71.      else
72.        System.out.println("Espere su turno");
73.    }
74.    else if (est.equals(estadoJugador.victoria))
75.      System.out.println("Ha vencido");
76.    else if (est.equals(estadoJugador.derrota))
77.      System.out.println("Ha perdido");
78.    else if (est.equals(estadoJugador.tablas))
79.      System.out.println("TABLAS");
80.  }

81.  public void jugar() {
82.    dibujarTablero();
83.    System.out.println("Espere ...");
84.    while (true);
85.  }

86. }
```

Se puede ver que la clase *JugadorTextoAplic* implementa la interfaz *JugadorInter* (Línea 5), y el método *estadoCambiado* (Línea 40) provee la funcionalidad para la interacción con el usuario, ya que es invocado por el objeto *jugador* cada vez que se produce un cambio en el estado de la partida.

El constructor de la clase (Línea 11) crea el ORB (Línea 13), y con este crea un Jugador utilizando la clase *JugadorImpl* (Línea 14), con el que interactuará la interfaz para mostrar el desarrollo de la partida.

3.8 Uso del servicio de nombres

Uno de los grandes problemas con los que se han enfrentado los programadores que utilizaban CORBA los últimos años, ha sido la obtención del IOR del servidor por parte de los clientes. Como se ha visto la solución tomada a este problema hasta el momento es muy básica, ya que se basa en la distribución de un fichero de texto que contiene dicho IOR.

Se pueden encontrar otras soluciones mejores, entre las que destacan las soluciones propietarias de los distribuidores de ORB's, como por ejemplo Visibroker de Visigenic que utiliza una implementación del BOA que permite obtener referencias a servidores de forma automática.

Junto con el estándar CORBA, el OMG definió también una serie de servicios para facilitar el desarrollo de aplicaciones con CORBA. Cada uno de los fabricantes implementa los servicios que cree adecuados, siempre siguiendo el estándar, de tal forma que una de las formas de decidirse por un fabricante u otro puede ser el número de servicios que implemente.

Entre estos servicios nos encontramos con el *servicio de nombres*, que es uno de los más utilizados, y que implementan todos los ORB's estudiados. Este servicio sería parecido al DNS o al NIS, es decir una especie de listín telefónico donde podemos encontrar referencias a servidores a partir de nombres. Los servidores se dan de alta en el servidor de nombres con un identificador, de tal forma que cuando un cliente pregunte por dicho identificador el servidor de nombres le devuelva una referencia al objeto servidor.

Los nombres dados a los objetos se almacenan de una forma jerárquica con estructura de árbol. A cada uno de los nodos no terminales del árbol (*contexto* según el OMG) se le asocia un nombre, y los nodos hoja además de un nombre tienen asociado un objeto. De tal forma que la referencia a un objeto se encuentra totalmente calificada por la secuencia de nombres desde la raíz del árbol hasta la hoja donde se encuentra la referencia.

Para la utilización del servicio de nombres hay un paquete llamado "org.omg.CosNaming" donde se encuentra toda la funcionalidad asociada a dicho servicio.

El código para dejar una referencia a un objeto en el servidor de nombres dentro de un determinado contexto como por ejemplo el *contexto* TicTacToe y con el nombre *Servidor* sería:

```
1.      ORB orb = ORB.init(args,null);
2.      org.omg.CORBA.Object obj;
3.      try {
4.          obj = orb.resolve_initial_references("NameService");
5.      } catch (Exception ex) {
6.       System.out.println("Error: Servicio de nombres no encontrado");
7.        return;
8.      }
9.      NamingContext root_context = NamingContextHelper.narrow(obj);
10.     if (root_context==null) {
11.      System.out.println("Error: Servicio de nombres no encontrado");
12.       return;
13.     }
14.     TicTacToe.Juego implObject = new JuegoImpl(orb);
15.     NameComponent nombre[] = new NameComponent[1];
16.     nombre[0]=new NameComponent("TicTacToe","");
17.      NamingContext servidores;
18.      try {
19.         servidores =
              NamingContextHelper.narrow(root_context.resolve(nombre));
20.      } catch (org.omg.CosNaming.NamingContextPackage.NotFound ex) {
21.        servidores = root_context.bind_new_context(nombre);
22.      }
23.     try {
24.          nombre[0]=new NameComponent("Servidor","");
25.          servidores.bind(nombre,implObject);
26.     }catch (org.omg.CosNaming.NamingContextPackage.AlreadyBound ex){
27.           System.out.println("Error: Ya existe el servidor");
28.            return;
29.     }
```

Lo primero que tiene que hacer cualquier objeto que desee acceder al servicio de nombres es obtener una referencia al servidor de nombres, para ello el ORB ofrece el método *resolve_initial_references* para acceder a los diferentes servicios, este método tiene un parámetro del tipo *String* que debe ser "*NameService*" para obtener una referencia al servicio de nombres (Línea 4). Pero este método devuelve un objeto de la clase *org.omg.CORBA.Object* que hay que transformar al tipo *org.omg.CosNaming.NamingContext*, y para ello se utiliza el método *narrow* de la clase *NamingContextHelper* (Línea 9).

El objeto devuelto es una referencia al *contexto raíz* (nodo raíz del árbol de nombres), a partir de aquí se tiene que avanzar en el árbol hasta llegar al nodo intermedio donde deseemos dejar la referencia a nuestro servidor. Para avanzar por el árbol se utiliza el método *resolve* de la interfaz *NamingContext* (Línea 19). Este método tiene un único parámetro que es el nombre del siguiente nodo al que se quiere avanzar, este nombre es del tipo *NameComponent* (Línea 15) y está formado por dos cadenas de texto, la primera es el nombre en sí, y la segunda es un tipo, que puede calificar a la primera cadena, aunque lo que define un nodo es su nombre (Línea 16).

Si durante el acceso al nodo deseado (método *resolve*) se produce la excepción *org.omg.CosNaming.NamingContextPackage.NotFound* (Línea 20), significaría que el nodo deseado no existe, en este caso lo que se hará será crearlo con el método *bind_new_context* que recibe el nombre del *contexto* a crear (Línea 21).

Una vez obtenido el nodo donde se encontrará la referencia al servidor, se utilizará el método *bind* para establecer dicha referencia (Línea 25). Este método tiene dos parámetros, el nombre que se le va a asociar y el objeto servidor al que se le va a asociar.

El cliente debe obtener también la referencia al *contexto raíz* para después recorrer el árbol hasta la hoja con la referencia al servidor deseado, lo cual se realiza utilizando el método *resolve* de la interfaz *NamingContext*, como puede verse a continuación:

```
1.      ORB orb = ORB.init(args,null);
2.      org.omg.CORBA.Object obj;
3.      try {
4.          obj = orb.resolve_initial_references("NameService");
5.      } catch (Exception ex) {
6.       System.out.println("Error: Servicio de nombres no encontrado");
7.        return;
8.      }
9.      NamingContext root_context = NamingContextHelper.narrow(obj);
10.     if (root_context==null) {
11.      System.out.println("Error: Servicio de nombres no encontrado");
12.       return;
13.     }
14.     NameComponent nombre[] = new NameComponent[1];
15.     nombre[0]=new NameComponent("TicTacToe","");
16.      NamingContext servidores;
17.      try {
18.         servidores =
              NamingContextHelper.narrow(root_context.resolve(nombre));
19.      } catch (org.omg.CosNaming.NamingContextPackage.NotFound ex) {
20.        System.out.println("Error: Servidor no encontrado");
21.        return;
22.        }
23.     try {
24.        nombre[0]=new NameComponent("Servidor","");
25.        org.omg.CORBA.Object juegoObj=servidores.resolve(nombre);
26.        Juego juego=JuegoHelper.narrow(juegoObj);
27.     }catch (Exception ex){
28.           System.out.println("Error: Servidor no encontrado");
29.            return;
30.     }
```

El servicio de nombres se va a aprovechar para poder tener más de un servidor de TicTacToe a la espera, de tal forma que los clientes se van uniendo al que esté disponible. Los servidores se encontrarán en el contexto *TicTacToe*, con el nombre *ServidorXX*, siendo XX un número que identificará de forma unívoca a cada servidor.

Para dar de alta el servidor se intentará dar de alta con el nombre Servidor0, y si esto falla se intentará con el siguiente número, hasta un máximo de 100 servidores, como puede verse a continuación:

```
1.      int i=0;
2.      while (true) {
3.        try {
4.          nombre[0]=new NameComponent("Servidor" + i,"");
5.          servidores.bind(nombre,implObject);
          // Si lo consigue asociar termina
6.          break;
7.      }catch (org.omg.CosNaming.NamingContextPackage.AlreadyBound ex){
          // Mientras no lo pueda asociar porque exista otro
```

```
          // servidor continua intentandolo
8.          i++;
9.          if (i==100) {
10.         System.out.println("Error: No más servidores");
11.            return;
12.       }
13.     }
```

A la hora de obtener una referencia al servidor por parte de un objeto *jugador*, se pueden intentar sucesivamente obtener una referencia a *Servidor0, Servidor1*, ... hasta que se encuentre un servidor existente, y que no tenga aún los dos jugadores:

```
1.    int i=0;
2.      while (true) {
3.        try {
4.          nombre[0]=new NameComponent("Servidor" + i,"");
5.          juegoObj=servidores.resolve(nombre);
6.          juego=JuegoHelper.narrow(juegoObj);
7.          tablero=juego.anadirJugador(this,fh);
          // Si lo consigue asociar termina
8.          break;
10.        } catch (Exception ex) {
          // Mientras no lo pueda asociar porque exista otro
          // servidor continua intentandolo
11.          i++;
12.          if (i==100) {
13.            System.out.println("Error: No hay servidor");
14.         }
15.       }
```

También se puede obtener una lista de los servidores que haya en el contexto *TicTacToe* y conectarse al primero que no tenga los dos jugadores:

```
1.    BindingListHolder lista = new BindingListHolder();
2.    BindingIteratorHolder listaiter = new BindingIteratorHolder();

3.    servidores.list(100,lista,listaiter);

4.    int i=0;
5.    while (i<lista.value.length) {
6.      try {
7.        juegoObj=servidores.resolve(lista.value[i].binding_name);
8.        juego=JuegoHelper.narrow(juegoObj);
9.        tablero=juego.anadirJugador(this,fh);
10.       break;
11.     } catch (Exception ex) {
12.       juego=null;
13.       i++;
14.     }
15.   }
16.   if (juego==null)
17.       System.out.println("Error: No hay servidor");
```

Para almacenar la lista de referencias que se encuentran en un nodo se utiliza un objeto de la clase *BindingListHolder* (Línea 1), y para poder recorrerlos se utiliza un iterador del tipo *BindingIteratorHolder*. Para obtener la lista se utiliza el método *list* de la interfaz *NamingContext* al cual se le pasa el número máximo de referencias a obtener, la lista donde almacenarlas y el iterador que se usará sobre dicha lista (Línea 3).

Para recorrer la lista se puede utilizar el iterador, o un array de referencias que se almacenan en el atributo *value* de la lista (Líneas 5 y 7).

Las referencias a los clientes no se almacenan en el servicio de nombres, porque no es necesario, ya que el servidor del juego obtiene una referencia a ellos al añadirse a la partida (como parámetro del método *anadirJugador*).

Para poder ejecutar en este caso el juego, hay que lanzar el servidor de nombres, que en el caso de las JDK 1.2 se llama *tnameserv*. Tras esto se lanza el servidor del TicTacToe. Si este servidor no está en la misma máquina que el servidor de nombres, se le debe indicar donde está éste, lo cual se puede hacer con una propiedad al crear el ORB o con un parámetro de la aplicación:

```
>java TicTacToe.JuegoServer -Dorg.omg.CORBA.ORBInitialHost=192.168.105.2
```

Por otro lado los clientes se lanzan igual que hasta ahora, pero si no se ejecutan en la misma máquina que el servidor de nombres se le debe pasar el mismo parámetro que al *JuegoServer*.

Los applets buscan el servidor de nombres no en la máquina donde se ejecutan sino en el servidor Web de donde han sido obtenidos, si no se encuentra en esta máquina hay que indicarlo o bien con una propiedad al crear el ORB, o bien con un parámetro al applet:

```
<PARAM  NAME="org.omg.CORBA.ORBInitialHost"
        VALUE="192.168.105.2">
```

3.9 Implementación en C++

De la misma forma que ha sido desarrollado el juego para Java, se ha desarrollado también para C++, implementación que se describirá a continuación. De la misma forma se mostrará la interacción entra un servidor y un cliente en lenguajes distintos.

Para el desarrollo en C++ se ha utilizado la implementación de CORBA llamada *ORBacus* en su versión 3.1 desarrollada por *Object-Oriented Concepts Inc.*, que se encuentra tanto para C++ como para Java. Esta implementación puede ser utilizada tanto en entornos Linux/Unix como en Windows, ya que se puede obtener el código fuente para compilarlo en la plataforma deseada, dichas fuentes son de libre distribución y pueden encontrarse en http://www.ooc.com.

ORBacus viene provisto de un traductor de idl a C++ y a Java, de una implementación del servicio de nombres y del servicio de eventos.

Se ha implementado la última versión del juego TicTacToe en C++ por ser la más completa y la que más aspectos cubre de la programación con CORBA. Para dicha implementación se ha utilizado la misma definición de interfaces que en el capítulo **3.7**,

con lo que se podrá mezclar un servidor realizado en uno de los dos lenguajes con los clientes del otro lenguaje (propiedad de independencia del lenguaje de CORBA).

En primer lugar se debe compilar el fichero *TicTacToe.idl* para generar los stub's y los skeleton's en C++, para realizar esta operación se utiliza el traductor *idl* que obtenemos con *ORBacus*.

```
>idl TicTacToe.idl
```

Con dicho comando obtenemos cuatro ficheros:

TicTacToe.cpp TicTacToe.h TicTacToe_skel.cpp TicTacToe_skel.h

Los dos primeros corresponden con la definición e implementación de los stub's y tipos de datos generados a partir de la definición en IDL, y los dos últimos corresponden con la implementación de los skeleton´s necesarios para desarrollar los servidores.

Siguiendo los mismos pasos que para el desarrollo de la aplicación en Java, una vez obtenidos los ficheros anteriores de forma automática se pasará a la implementación de los objetos servidores, a continuación se desarrollará una aplicación encargada de generar un servidor y darlo de alta en el servicio de nombres, y finalmente se desarrollará un cliente que utilice dicho servidor.

Los dos objetos servidores que necesitan implementación son el *Juego* y el *Jugador*, aunque éste último sólo sea necesario para el desarrollo del cliente, por lo que se describirá más abajo.

La implementación para el *Juego* estaría dividida en dos ficheros (como es normal en C++), un fichero *.h* con la definición de la clase y otro *.cpp* con la implementación de los métodos de dicha clase:

JuegoImpl.h

```
1. #ifndef JUEGOIMPL_H
2. #define JUEGOIMPL_H

3. #include <OB/CORBA.h>
4. #include "TicTacToe_skel.h"

5. class TicTacToe_Juego_Impl : public TicTacToe_Juego_skel {

6. TicTacToe_tablero_var tablero;
7. TicTacToe_Jugador_var jug1,jug2;
8. TicTacToe_estadoJugador estado;
9. CORBA_ORB_var orb;
10.CORBA_String_var IORJug1;
11.CORBA_String_var IORJug2;
12.CORBA_String_var IORturno;

13.void inicializaTablero();
14.TicTacToe_ficha vencedor();
15.bool tablerolleno();

16.public:

17.TicTacToe_Juego_Impl(CORBA_ORB_ptr orbArg);
18.TicTacToe_tablero_slice* tab();
```

```
19.TicTacToe_tablero_slice* anadirJugador(TicTacToe_Jugador_ptr jug,
                                          TicTacToe_ficha& usar);

20.TicTacToe_tablero_slice* ponerFicha(TicTacToe_Jugador_ptr jug,
                                          CORBA_Short x, CORBA_Short y);
21.};

22.#endif
```

Es conveniente destacar que en todo fichero de cabeceras deben incluirse la líneas que indiquen al preprocesador que no añada más de una vez este fichero en una compilación (Líneas 1,2 y 22).

A continuación deben incluirse los ficheros de cabeceras con todas las definiciones necesarias de CORBA (<*OB/CORBA.h*>) y el que tiene la definición de los skeleton`s (<*TicTacToe_skel.h*>).

Es entonces cuando comienza la definición propiamente dicha de la clase que implementará la interfaz *Juego*. El nombre de la clase será *modulo_interfaz_impl* por convenio en el resto del proyecto, y debe heredar de forma pública de la clase que contiene el skeleton de esta interfaz: *TicTacToe_Juego_skel* (Línea 5). Como puede verse en *ORBacus* no se utiliza el espacio de nombres que aconseja el OMG para el nombrado de las clases dentro de los módulos, sino que se utiliza la otra alternativa, utilizar el carácter '_' entre nombre de módulo y nombre de interfaz (En futuras versiones se sustituirá esta forma por la aconsejada).

En la línea 6 se define la variable donde será almacenado el valor del atributo *tab* definido en la interfaz. Como puede verse esta variable es del tipo *TicTacToe_tablero_var*, se utilizarán siempre variables del tipo *_var* para evitar tener que liberar las referencias antes de la destrucción de la variable, ya que con este tipo de variables esto se hace de forma automática como se define en el estándar de transformación de IDL a C++.

En las líneas 7 a 12 se definen una serie de variables internas a la clase *Juego* que sirven para manejar el estado de una partida tal y como se hacía en la implementación para Java. Destacar que todas ellas son del tipo *_var* excepto *estado*, ya que es una variable de un tipo enumerado, que es representado mediante un entero en C++, es decir esta variable es de un tipo básico de C++, con lo que no necesita el manejo de referencias.

A continuación se definen tres métodos internos a la clase tal como se hizo en Java (Líneas 13,14 y 15).

En la línea 17 se define un constructor para la clase en el que se le debe pasar un ORB. Cuando se trata del paso de parámetros se utilizarán variables del tipo *_ptr* ya que no se necesita que se manejen automáticamente las referencias.

Las tres operaciones definidas en el IDL para esta interfaz se ven definidas en la clase en las líneas 18, 19 y 20, y verán su implementación en el siguiente fichero:

JuegoImpl.cpp

```
1. #include "JuegoImpl.h"
```

```
2. #include <string.h>

3. TicTacToe_Juego_Impl::TicTacToe_Juego_Impl(CORBA_ORB_ptr orbArg) {
4.     orb=orbArg;
5.     inicializaTablero();
6.     estado=TicTacToe_jugando;
7.     jug1=TicTacToe_Jugador::_nil();
8.     jug2=TicTacToe_Jugador::_nil();
9. }

10.void TicTacToe_Juego_Impl::inicializaTablero() {
11.    tablero = new TicTacToe_ficha[3][3];
12.    for (int i=0; i<3; i++)
13.      for (int j=0; j<3; j++)
14.        tablero[i][j]=TicTacToe_vacia;
15.}

16.TicTacToe_tablero_slice* TicTacToe_Juego_Impl::tab() {
17.   return TicTacToe_tablero_dup(tablero);
18.}

19.TicTacToe_tablero_slice*
       TicTacToe_Juego_Impl::anadirJugador(TicTacToe_Jugador_ptr jug,
             TicTacToe_ficha& usar) {
20.    if (CORBA_is_nil(jug1)) {
21.      jug1=TicTacToe_Jugador::_duplicate(jug);
22.      usar=TicTacToe_cruz;
23.      IORJug1=CORBA_string_dup(orb->object_to_string(jug1));
24.      cout << "Jugador 1 Anadido: " << IORJug1 << endl;
25.    }

26.    else if (CORBA_is_nil(jug2)) {
27.      jug2=TicTacToe_Jugador::_duplicate(jug);
28.      usar=TicTacToe_circulo;
29.      IORJug2=CORBA_string_dup(orb->object_to_string(jug2));
30.      cout << "Jugador 2 Anadido: " << IORJug2 << endl;
31.      jug1->turno(true,tablero);
32.      jug2->turno(false,tablero);
33.      IORturno=CORBA_string_dup(IORJug1);
34.    }
35.    else {
36.       cerr << "ERROR :Jugador no Anadido" << endl;
37.       throw TicTacToe_YaHayDosJugadores();
38.    }

39.    return TicTacToe_tablero_dup(tablero);

40.}

41.TicTacToe_ficha TicTacToe_Juego_Impl::vencedor() {
...
42.}

43.bool TicTacToe_Juego_Impl::tablerolleno() {
...
44.}

45.TicTacToe_tablero_slice*
       TicTacToe_Juego_Impl::ponerFicha(TicTacToe_Jugador_ptr jug,
             CORBA_Short x, CORBA_Short y) {
46.    if (estado!=TicTacToe_jugando)
47.      throw TicTacToe_JugadorInvalido();

48.    if (x<0 || x>2 || y<0 || y>2)
49.      throw TicTacToe_CasillaOcupada();

50.    if (tablero[x][y]!=TicTacToe_vacia)
51.      throw TicTacToe_CasillaOcupada();
```

```
52.    if (CORBA_is_nil(jug1) || CORBA_is_nil(jug2))
53.      throw TicTacToe_JugadorInvalido();

54.    CORBA_String_var IORJug =
                  CORBA_string_dup(orb->object_to_string(jug));

55.    if (strcmp(IORJug,IORturno)) {
56.       cout << " No te toca " << endl;
57.      throw TicTacToe_JugadorInvalido();
58.    }

59.    if (!strcmp(IORJug,IORJug1))
60.      tablero[x][y]=TicTacToe_cruz;
61.    else
62.      tablero[x][y]=TicTacToe_circulo;
63.    TicTacToe_ficha f=vencedor();

64.    if (f==TicTacToe_cruz) {
65.      jug1->fin(true,false,tablero);
66.      jug2->fin(false,false,tablero);
67.      estado=TicTacToe_victoria;
68.    }
69.    else if (f==TicTacToe_circulo) {
70.      jug1->fin(false,false,tablero);
71.      jug2->fin(true,false,tablero);
72.      estado=TicTacToe_derrota;
73.    }
74.    else {
75.      if (tablerolleno()) {
76.        jug1->fin(false,true,tablero);
77.        jug2->fin(false,true,tablero);
78.        estado=TicTacToe_tablas;
79.      }
80.      else {
81.        if (!strcmp(IORJug,IORJug1)) {
82.          jug2->turno(true,tablero);
83.          jug1->turno(false,tablero);
84.      IORturno=IORJug2;
85.        }
86.        else {
87.          jug1->turno(true,tablero);
88.          jug2->turno(false,tablero);
89.      IORturno=IORJug1;
90.        }
91.      }
92.    }

93.    return TicTacToe_tablero_dup(tablero);
94.}
```

Como puede verse la implementación es semejante a la que encontrábamos en Java, salvando las distancias entre ambos lenguajes, y entre las especificaciones realizadas por el OMG para dichos lenguajes. Como diferencias principales cabe destacar las expuestas a continuación.

En la inicialización se les asigna a ambos jugadores el valor de nulo devuelto por el método *_nil* que encontramos en la clase *TicTacToe_Jugador* siendo dicho método estático (Líneas 7 y 8).

En las operaciones en las que se debe devolver el tablero se utiliza el método *TicTacToe_tablero_dup* (Líneas 17, 39 y 93) para obtener una nueva referencia al tablero antes de devolverlo, ya que al devolver un objeto de forma remota, el ORB

elimina la referencia que se devuelve. Dicho método es generado en la transformación de IDL a C++ por ser *tablero* un tipo array, si este fuera una clase existiría un método en dicha clase llamado *_duplicate* para realizar la misma función.

Para comprobar si un objeto es nulo se utiliza el método *CORBA_is_nil* (Líneas 20, 26 y 52).

Cuando se le asigna un valor a una variable interna del objeto servidor se debe obtener una nueva referencia a dicho valor, para ello se utilizará el método *_duplicate* (Líneas 21 y 27).

A la hora de utilizar cadenas de caracteres del tipo *CORBA_String* se debe recordar que hay que utilizar el correspondiente método *CORBA_string_dup* para almacenar la cadena en una zona de memoria válida, como se describe en el capítulo **2.5** (Líneas 23, 25 ...).

A continuación se desarrolla el programa que será el encargado de dejar el objeto Juego a la espera de peticiones remotas, y asociará una referencia a dicho objeto con un nombre del servicio de nombres, posibilitando de esta forma a los clientes la obtención de la referencia al objeto servidor.

El código para dicho programa es el siguiente:

JuegoServer.cpp

```
1. #include <OB/CORBA.h>
2. #include <OB/CosNaming.h>
3. #include "TicTacToe.h"
4. #include "JuegoImpl.h"
5. #include <signal.h>

6. CORBA_BOA_var boa;
7. void finalizar(int a);

8. int main(int narg,char *arg[])
9. {

10.   CORBA_ORB_var orb = CORBA_ORB_init(narg,arg);
11.   boa = orb->BOA_init(narg,arg);

   // Uso el servicio de nombres
12.   CORBA_Object_var obj;

13.   try
14.   {
15.      obj = orb -> resolve_initial_references("NameService");
16.   }
17.   catch(const CORBA_ORB::InvalidName&)
18.   {
19.      cerr << arg[0] << ": no puede resolver 'NameService'" << endl;
20.      return 1;
21.   }

22.   if(CORBA_is_nil(obj))
23.   {
24.      cerr<<arg[0]<<":'NameService' es una referencia a nil" << endl;
25.      return 1;
26.   }

27.   CosNaming_NamingContext_var rootContext =
                          CosNaming_NamingContext::_narrow(obj);
```

```
28.   if(CORBA_is_nil(rootContext))
29.   {
30.      cerr << arg[0]
31.          << ": 'NameService' no es un servidor de nombres"
32.          << endl;
33.      return 1;
34.   }

35.   try
36.   {
37.      TicTacToe_Juego_var implObject = new TicTacToe_Juego_Impl(orb);

38.      orb->connect(implObject);

39.      CosNaming_Name nombre;
40.      nombre.length(1);
41.      nombre[0].id = CORBA_string_dup("TicTacToe");
42.      nombre[0].kind = CORBA_string_dup("");

43.      CosNaming_NamingContext_var servidores;

44.      try {
45.        servidores = CosNaming_NamingContext::_narrow(
                                      rootContext->resolve(nombre));
46.      } catch (CosNaming_NamingContext::NotFound ex) {
47.        servidores = rootContext->bind_new_context(nombre);
48.      }
49.      int i=0;
50.      while (true) {
51.        try {
52.          nombre[0].id = CORBA_string_dup("Servidor");
53.          nombre[0].id += i;
54.          servidores->bind(nombre,implObject);
          // Si lo consigue asociar termina
55.          break;
56.         } catch (CosNaming_NamingContext::AlreadyBound ex) {
          // Mientras no lo pueda asociar porque exista otro
          // servidor continua intentandolo
57.          i++;
58.          if (i==100) {
59.             cerr << "Error: No se pueden tener mas de 100
                                          servidores" << endl;
60.             return 1;
61.          }
62.        }
63.      }

64.      cout << "Esta listo. Como Servidor" << i << endl;
65.      cout << "Pulse CTRL+Z para finalizar" << endl;
66.      signal(SIGTSTP,finalizar);

67.      boa->impl_is_ready(CORBA_ImplementationDef::_nil());

68.      servidores->unbind(nombre);

69.   }
70.   catch (const CosNaming_NamingContext::NotFound& ex)
71.   {
72.      cerr << arg[0] << ": Producida la excepcion 'NotFound' (";
73.      switch(ex.why)
74.      {
75.          case CosNaming_NamingContext::missing_node:
76.                cerr << "nodo no encontrado";
77.                break;

78.          case CosNaming_NamingContext::not_context:
79.                cerr << "no contexto";
```

```
80.                 break;

81.           case CosNaming_NamingContext::not_object:
82.                 cerr << "no objeto";
83.                 break;
84.      }
85.      cerr << ")" << endl;
86.      return 1;
87.   }
88.   catch(const CosNaming_NamingContext::CannotProceed&)
89.   {
90.      cerr<<arg[0]<<":Producida la excepcion 'CannotProceed'"<<endl;
91.      return 1;
92.   }
93.   catch(const CosNaming_NamingContext::InvalidName&)
94.   {
95.      cerr<<arg[0]<< ": Producida la excepcion 'InvalidName'"<<endl;
96.      return 1;
97.   }
98.   catch(const CosNaming_NamingContext::AlreadyBound&)
99.   {
100.      cerr<<arg[0]<<": Producida la excepcion 'AlreadyBound'"<<endl;
101.      return 1;
102.   }
103.   catch(const CosNaming_NamingContext::NotEmpty&)
104.   {
105.      cerr<<arg[0] << ": Producida la excepcion 'NotEmpty'" << endl;
106.      return 1;
107.   }

108.   return 0;
109. }

110. void finalizar(int a) {
111.    boa->deactivate_impl(CORBA_ImplementationDef::_nil());
112. }
```

En primer lugar se crea un objeto que represente al *ORB* (Línea 10) que se va a utilizar y aparece por primera vez un *Adaptador de Objetos*, en este caso el *BOA*, ya que con *ORBacus* es necesaria su utilización para que un objeto servidor pueda recibir invocaciones remotas. El *BOA* se crea a partir del *ORB* utilizando el método *BOA_init* que recibe los mismos parámetros que son necesarios para la inicialización del propio *ORB*, es decir el número de argumentos, y dichos argumentos (Línea 11).

Para dejar un objeto a la espera de invocaciones basta con crear dicho objeto a partir de la implementación que se ha realizado para él (Línea 37) y conectarlo con el *ORB* utilizando el método *connect* (Línea 38). Para que el *ORB* le pase al objeto las peticiones que se le realicen hay que utilizar el método *impl_is_ready* del *BOA* que deja detiene el programa a la espera de invocaciones remotas (Línea 67).

Con todo lo anterior se tiene un objeto servidor a la espera de invocaciones remotas, pero falta poner a disposición de los clientes dicho servidor, y para ello se utilizará el servicio de nombres. La utilización de dicho servicio es semejante a la que se encuentra en Java, ya que la llamada a los métodos encargados de crear un nuevo contexto, o de avanzar por uno, como el resto de métodos para trabajar con el servicio son estándares y están definidos en el documento proporcionado por el OMG.

Se encuentra un problema a la hora de la finalización del programa, ya que el programa se encuentra bloqueado en la llamada a *impl_is_ready*, de tal forma que si se aborta la ejecución no se podrá desconectar el objeto del servicio de nombres, con lo

que clientes remotos pueden pensar que dicho objeto sigue a la escucha. Para solucionar este problema se captura la señal de *STOP* (Línea 66) que se produce al pulsar las teclas *Ctrl+Z* en un sistema Linux, de tal forma que cuando se reciba dicha señal sea invocado el método *deactivate_impl* del *BOA* (Línea 111) para que se salga del bloqueo producido anteriormente, y de esta forma se pueda dar de baja el objeto servidor en el servicio de nombres (Línea 68) antes de finalizar el programa.

Para poner en funcionamiento el programa que crea un objeto servidor para el juego hay que compilar en primer lugar las clases desarrolladas, para ello se puede utilizar un fichero Makefile como el que se presenta a continuación:

Makefile

```
CPPFLAGS = -I.

JuegoServer: TicTacToe.o TicTacToe_skel.o JuegoImpl.o JuegoServer.o
	g++ TicTacToe.o TicTacToe_skel.o JuegoImpl.o JuegoServer.o -o
		JuegoServer -lOB -lCosNaming
```

Lo primero que se debe hacer es lanzar el servicio de nombres, el cual se encuentra en un programa que acompaña al *ORBacus* llamado *nameserv*. En *ORBacus* para que un programa pueda acceder al servicio de nombres debe conocer el *IOR* de dicho servidor, para lo cual se proporciona la opción *–i* al servidor de nombres, de tal forma que saque por pantalla el *IOR*, con lo que habrá que ejecutar:

```
> nameserv –i >ficheroIOR
```

El programa que quiera hacer uso de dicho servicio deberá tener acceso al fichero generado y se ejecutará con la siguiente sentencia:

```
> JuegoServer –ORBservice NameService ‘cat ficheroIOR‘
```

La forma de hacer llegar el fichero al ordenador donde se ejecute dicha sentencia se deja a libre elección (mediante Web, Ftp, Irc, Nfs, ...).

Para el desarrollo de un cliente es necesario implementar la interfaz *Jugador*, y el programa que se encargue de la interacción con el usuario. La implementación de dicha interfaz se realiza de la misma forma que la realizada para la interfaz *Juego*, con dos ficheros, uno para la definición de la clase y otro para la implementación de los métodos de dicha clase, y deben tenerse en cuenta las mismas consideraciones comentadas anteriormente.

Para la implementación del cliente se ha utilizado una clase que se encarga de crear un nuevo objeto *Jugador* encargado de interactuar con el servidor en el que se dé de alta. A continuación se muestra la definición de dicha clase:

JugadorTextoAplic.h

```
#ifndef JUGADORTEXTOAPLIC_H
#define JUGADORTEXTOAPLIC_H

#include <OB/CORBA.h>
#include "TicTacToe.h"
```

```
#include "JugadorInter.h"

class JugadorTextoAplic : public TicTacToe_JugadorInter
{
TicTacToe_Jugador_var jugador;
void dibujarTablero();
CORBA_BOA_var boa;

public:

JugadorTextoAplic(int narg, char *arg[]);
void estadoCambiado(TicTacToe_estadoJugador est,bool meToca);
void jugar();
};

#endif
```

En esta clase hay un constructor, un método para comenzar a jugar, y se hereda de la clase *TicTacToe_JugadorInter* que es la clase abstracta que define el método *estadoCambiado* tal como se hacía en Java.

La implementación para dicha clase se encuentra en el siguiente fichero:

JugadorTextoAplic.cpp

```
1. #include "JugadorTextoAplic.h"
2. #include "JugadorImpl.h"
3. #include <string>

4. JugadorTextoAplic::JugadorTextoAplic(int narg, char *arg[]) {
5.       CORBA_ORB_var orb = CORBA_ORB_init(narg,arg);
6.       boa = orb->BOA_init(narg,arg);
7.       boa->init_servers();
8.       jugador = new TicTacToe_Jugador_Impl(orb,this);
9. }

10. void JugadorTextoAplic::dibujarTablero() {
...
11. }

12. void JugadorTextoAplic::estadoCambiado(
                              TicTacToe_estadoJugador est,bool meToca) {
...
13. }

14. void JugadorTextoAplic::jugar() {
15.   dibujarTablero();
16.   cout << "Espere ..." << endl;
17.   boa->impl_is_ready(CORBA_ImplementationDef::_nil());
18. }

19. void main(int narg, char *arg[]) {
20.   JugadorTextoAplic aplic(narg,arg);
21.   aplic.jugar();
22. }
```

En este fichero se define la función *main* (Línea 19) que es la que se ejecuta en primer lugar, en ésta se crea un objeto de la clase *JugadorTextoAplic* (Línea 20) y se le pasa el mensaje *jugar* (Línea 21).

Al crear el objeto se ejecuta el constructor en el que se crea un *ORB* y un *BOA* (Líneas 5 y 6), para crear a continuación un *Jugador* (Línea 8), el problema se encuentra al ejecutarse el constructor de la implementación del *Jugador*, ya que se intenta pasar

como parámetro el propio objeto *Jugador* al *Juego* al llamar a *anadirJugador*, sin que se haya llamado al método *impl_is_ready* del *BOA*, con lo que se produce un error en tiempo de ejecución. Para solucionar este problema se debe ejecutar el método *init_servers* del *BOA* (Línea 7) antes de crear el objeto *Jugador*. El método *init_servers* es específico del ORB utilizado (*ORBacus*), aunque cualquier otro ORB dispondrá de un método con las mismas características. Estos problemas de incompatibilidad quedan solucionados con la especificación por parte del OMG del *Portable Object Adapter (POA)*, el cual no se ha utilizado porque, debido a su reciente aparición, no existen implementaciones actualmente.

En el método *jugar* se dibuja el tablero y se llama a *impl_is_ready*, lo cual bloquea el programa, aunque con la llegada de invocaciones remotas se ejecutará el método *estadoCambiado* donde se interactua con el usuario final.

Para compilar este programa se puede generar otro fichero *Makefile* (también se pueden unir el del servidor con este):

Makefile

```
CPPFLAGS = -I.

JugadorTextoAplic: JugadorTextoAplic.o JugadorImpl.o TicTacToe.o
                           TicTacToe_skel.o
       g++ JugadorTextoAplic.o JugadorImpl.o TicTacToe.o
              TicTacToe_skel.o -o JugadorTextoAplic -lOB -lCosNaming
```

Una vez compilado el programa se puede ejecutar si se encuentra en funcionamiento el servidor de nombres, y algún servidor del juego, con la siguiente instrucción:

```
> JugadorTextoAplic –ORBservice NameService ‘cat ficheroIOR‘
```

Con esto se añadiría un nuevo jugador a la partida, si esta partida ya tenía un jugador anteriormente, se comenzará pidiéndole la posición de la primera ficha al primer jugador.

Finalmente se describirá como hacer que interactue un programa distribuido si este ha sido realizado en diferentes lenguajes (C++ y Java en este caso), para ello se tratará de lanzar el servidor en C++ y un cliente en Java y otro en C++.

El principal problema se encuentra en la forma de acceder al servicio de nombres que tiene *ORBacus* y las *JDK1.2*, en el primer caso se debe tener el *IOR* del servicio de nombres, mientras que en el otro caso se accede al servicio de nombres mediante un protocolo propietario, conociendo la máquina y el puerto en el que se encuentra dicho servicio. Todo esto es suponiendo que se utilice el método del *ORB* llamado *resove_initial_services* que es el usado durante todo el capítulo, también se pude utilizar el método *string_to_object* utilizando el *IOR* del servidor de nombres, con lo que no habría ningún problema ni en C++ ni en Java.

Se va a suponer que se utilizan las implementaciones realizadas hasta el momento sin ninguna modificación, por esta razón se debe utilizar el servidor de nombres que acompaña a las *JDK1.2* ya que es el único capaz de comunicarse con un

programa realizado con dichas librerías, mientras que uno realizado en *ORBacus* puede comunicarse con cualquiera, mientras se conozca el *IOR* de éste.

En primer lugar debe lanzarse el servidor de nombres de las *JDK1.2* y almacenarse en un fichero el *IOR* de dicho servidor, para ello se utilizarán los siguientes comandos:

```
> tnameserv > auxIOR
> grep IOR auxIOR > ficheroIOR
```

A continuación ya se podrá lanzar el servidor:

```
> JuegoServer –ORBservice NameService ‘cat ficheroIOR‘
```

Y los dos clientes, uno en C++ y otro en Java:

```
> JugadorTextoAplic –ORBservice NameService ‘cat ficheroIOR‘
> java TicTacToe.JugadorAplic
```

Si se desease utilizar el servidor de nombres de *ORBacus* (ya que este permite referencias permanentes, con lo que prevendríamos caídas del servidor de nombres) debería modificarse el código del programa en Java de la siguiente forma:

```
org.omg.CORBA.Object obj=orb.resolve_initial_references("NameService");
                      →
org.omg.CORBA.Object obj=orb.string_to_object(cadenaIOR);
```

Donde *cadenaIOR* es la cadena que almacena el *IOR* del servidor de nombres, esta cadena puede ser obtenida por el programa a través de la línea de comandos de la siguiente forma:

```
> java TicTacToe.JugadorAplic ‘cat ficheroIOR‘
```

Con esto ya se podría utilizar el servidor de nombres de *ORBacus*, y almacenar el *IOR* de dicho servidor en el fichero correspondiente, tal y como se hacía anteriormente.

4. Ejemplo de una aplicación de negocio

En este capítulo se abordará el segundo de los objetivos del proyecto, el desarrollo de una aplicación de negocio utilizando CORBA.

No se trata de implementar completamente la aplicación, creando una interfaz gráfica muy cuidada, es decir no se pretende crear la aplicación final, sino estudiar los problemas que aparecen al desarrollar una aplicación de negocio, y valorar las posibles soluciones para estos problemas, creando un prototipo.

4.1 Análisis

4.1.1 Especificación

La aplicación que se va a desarrollar es un centro comercial virtual, la especificación para dicho centro es la que se muestra a continuación.

"El centro comercial virtual está basado en la idea del comercio electrónico. En el centro se recibirían las solicitudes de los comercios que se quisieran adherir. Cuando un comercio manda la solicitud, debe enviar todos sus datos y un catálogo con los productos que ofrece a través del centro. Para cada producto debe enviar unas especificaciones determinadas, como el precio, si este producto está en oferta, sus características físicas, etc.

En cualquier momento, un comercio puede introducir nuevos productos, eliminarlos de su catálogo o modificar cualquiera de sus características. Generalmente se tendrá el precio de cada producto actualizado.

También se gestiona la posible baja de un comercio o la modificación de cualquiera de sus datos.

Para comprar en el centro virtual, un internauta cualquiera puede introducirse a través de Internet en la página principal. Desde ese momento podrá navegar a través del centro virtual. Cuando encuentre un producto que le interese, lo introducirá en lo que llamamos el "carro de la compra". Este objeto contiene todos los productos que el cliente quiere comprar en ese momento. El cliente puede quitar un producto del carro en cualquier momento.

El cliente debe introducir su número de tarjeta de crédito junto con sus datos personales en la hoja de pedido, que formalizará la compra definitiva de todo lo que tiene en el carro de la compra. Si es socio debe poner su número de socio y de esta forma se ahorrará poner todos sus datos personales. Un cliente puede hacerse socio en cualquier momento de la navegación rellenando una solicitud para ello.

Los socios podrán beneficiarse de las ofertas especiales del centro, no tendrán que rellenar la hoja de pedido cada vez que realicen una compra, recibirán catálogos de los comercios, recibirán regalos por volumen de compra, etc.”

4.1.2 Estudio de los casos de uso

Representa la funcionalidad requerida en el sistema a implementar.

El diagrama obtenido a partir de la especificación anterior puede verse en la figura 10.

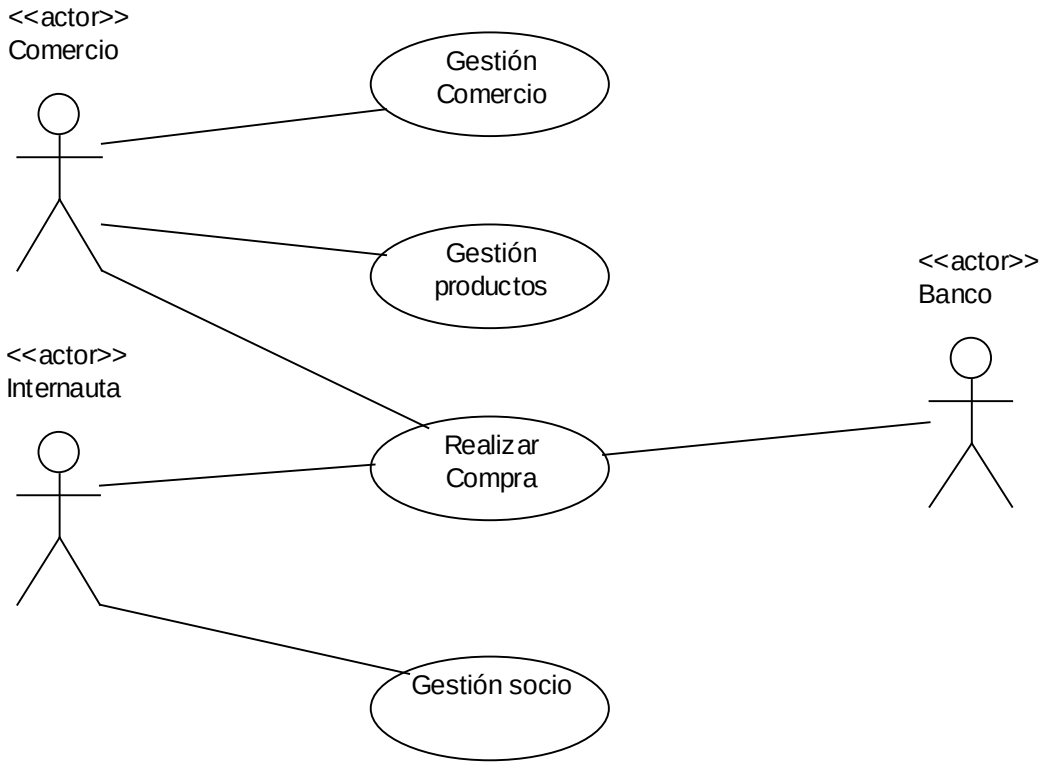

Figura 10 : Casos de uso del centro virtual

Se verá a continuación el significado de cada uno de los casos de uso:

4.1.3.1 Gestión comercio

Los comercios serán los encargados de darse de alta en el centro virtual, también deberán gestionar las modificaciones y las bajas.

4.1.3.2 Gestión productos

También los comercios gestionarán el catálogo de productos que tienen a la venta, características, precio, ...

4.1.3.3 Gestión socio

Los internautas podrán darse de alta en el centro para realizar las compras de una forma sencilla. También pueden modificar sus datos y darse de baja.

4.1.3.4 Realizar compra

Un internauta (Socio o no) entra en el centro comercial y realiza un paseo por los distintos comercios seleccionando productos que introduce en su carro de la compra, finalmente da el visto bueno con lo que se avisa al comercio de la compra realizada, y se realiza el pago con el banco.

Para explicar este caso de uso se muestra un diagrama de secuencia en la figura 11 en el que aparecen todos los objetos involucrados y su interacción.

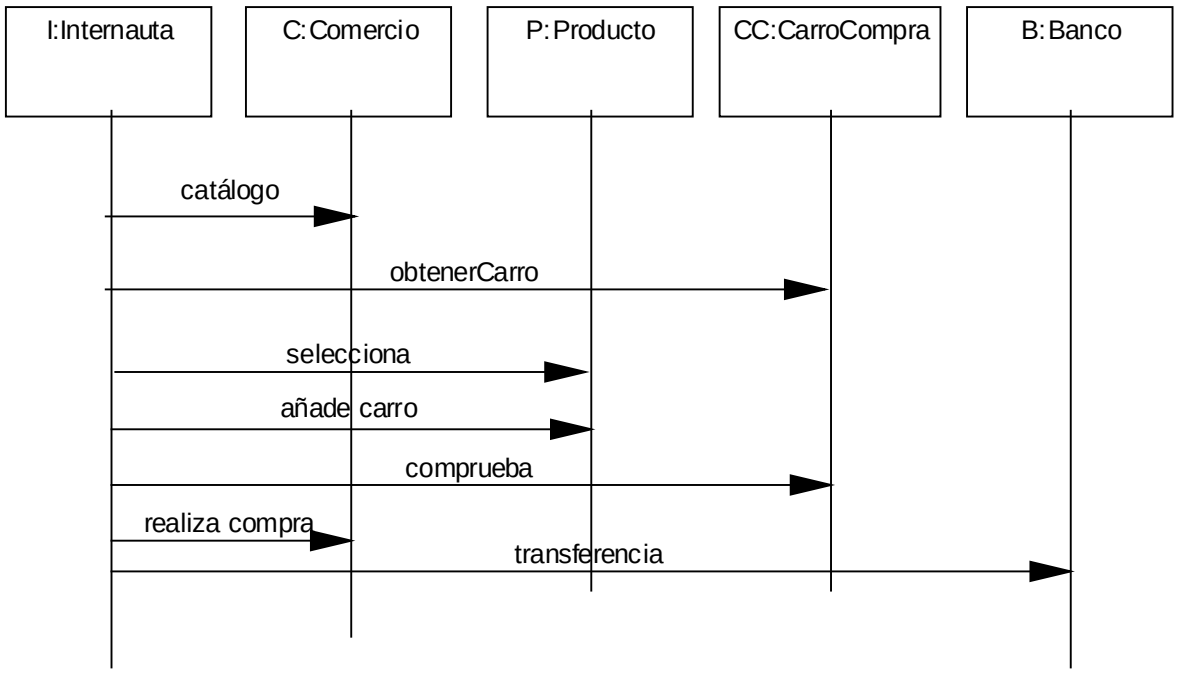

Figura 11 : Realizar compra. Diagrama de secuencia

4.1.3 Diagrama de clases

De un primer análisis se puede obtener el diagrama de clases de la figura 12, donde se encuentran las clases que pertenecen al dominio del problema.

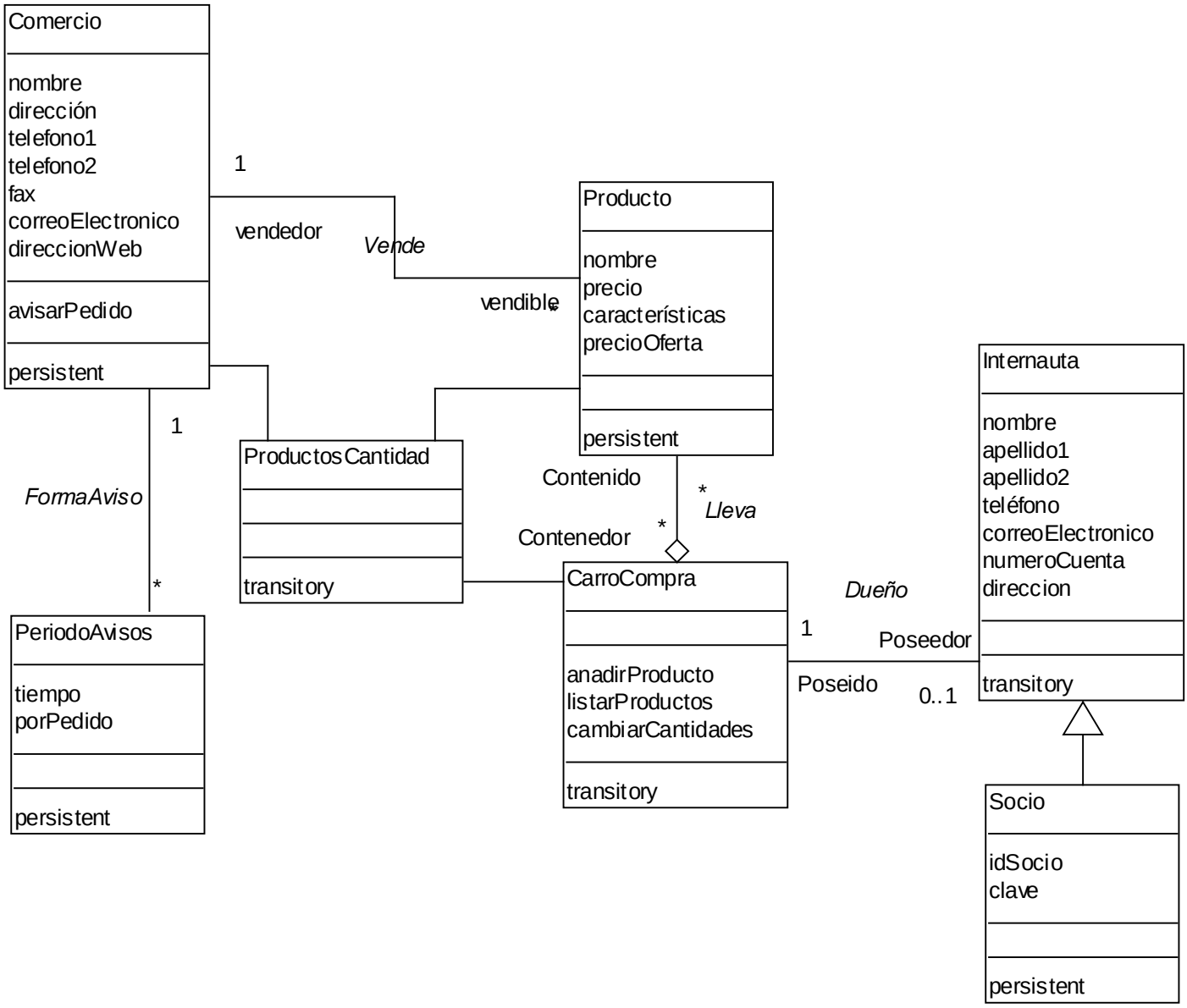

Figura 12: Diagrama de clases del centro

4.1.4 Cuestiones de diseño

Durante el diseño de la aplicación surgieron las siguientes cuestiones a resolver para poder realizar la implementación:

- ¿Dónde almacenar la información persistente acerca de los clientes, comercios, productos, ...?

- ¿Qué tipos de clientes se desarrollarán (Aplicación o Applet)?

- ¿Qué servidor Web se usará?

- ¿Qué formato tendrán las páginas Web de los comercios?

- ¿Quién y desde dónde podrá acceder a la base de datos?

- ¿Cómo accederán los clientes a los datos?

- ¿Cómo se le dará continuidad a la visita de un cliente por las distintas páginas Web, de forma que lo que introduzca en el carro de la compra en una, no desaparezca al pasar a otra?

4.1.5 Decisiones de diseño

Se le dará respuesta en este apartado a cada una de las preguntas del punto anterior, ofreciendo diferentes posibilidades donde sea oportuno, y seleccionando la opción que más se ajuste a nuestras necesidades.

- **¿Dónde almacenar la información persistente acerca de los clientes, comercios, productos, ...?**

El almacenamiento de información se realizará en una base de datos. Se ha tratado de buscar una base de datos comercial cuyo rendimiento sea aceptable y de bajo coste, a ser posible de libre distribución, pero que tenga las herramientas necesarias para realizar la aplicación deseada, en este caso acceso remoto a la base de datos utilizando los lenguajes de programación Java y C++.

De los sistemas existentes se utilizará PostgreSql por ser de libre distribución, con los componentes necesarios para poder realizar las consultas desde Java y C++. Otras posibilidad sería utilizar MySql, o algún sistema gestor en venta como Oracle o Informix. Se ha elegido PostgreSql por su fácil instalación y mantenimiento.

- **¿Qué tipos de clientes se desarrollarán (Aplicación o Applet)?**

La respuesta a esta pregunta depende de qué mercado de clientes se espera. Si sólo se esperan clientes conocidos a los que podamos distribuirles el software para que ellos se lo instalen se optaría por la primera opción, ya que es más sencillo el desarrollo al no tener que depender de distintas implementaciones de los navegadores (Con los posibles errores que éstas conlleven).

El centro comercial en desarrollo no se encontraría dentro de este grupo de aplicaciones, ya que la intención es que cualquier internauta pueda acceder al centro para realizar su compra, con lo que el mercado de clientes es muy amplio, obligando a

utilizar páginas Web, y por tanto applets dentro de dichas páginas para el acceso al centro virtual.

- **¿Qué servidor Web se usará?**

Dentro del gran abanico de posibilidades (Oracle Web Server, Internet Information Server, Netscape, Apache, CERN, ...) se ha elegido Apache por las siguientes razones:

- Gran productividad
- Libre distribución
- Ofrece muchos complementos (CGI, Servlet, Conexiones seguras con SSL, ...)
- Gran popularidad (Solución de problemas y actualizaciones de bugs casi inmediata)

Otra posible solución sería utilizar Oracle Web Server junto a una base de datos Oracle 8. En este caso tendríamos mayor productividad, más robustez, y un gran abanico de complementos. La decisión de utilizar PostgreSql es por ser éste de libre distribución, con lo que cualquier usuario puede instalarlo y probar el centro virtual, o continuar con su desarrollo.

Conviene señalar en cualquier caso que pasar de una opción a otra provocaría un cambio mínimo en el código de la aplicación, y muy localizado. Bastaría con modificar la cadena de conexión a la base de datos en la clase *ConstantesServidor*, siendo la nueva cadena de conexión *"jdbc:oracle:thin:scott/tiger@192.168.105.2:1521:oradb"*.

- **¿Qué formato tendrán las páginas Web de los comercios?**

Ante esta pregunta tenemos dos posibles soluciones:

a) Realizar una página Web estándar para todos los comercios, donde el centro virtual se encargue del mantenimiento de los productos, y de su visualización para todo cliente.

b) Que cada comercio pueda realizar su propia página Web, y mantenga sus productos en una base de datos propia, indicando al comercio únicamente los datos comunes a cualquier producto (nombre, breve descripción, precio, ...) que un internauta adquiera.

La solución adoptada es una mezcla de ambas, por un lado cada comercio tendrá su propia página Web, con el consiguiente mantenimiento de un servidor Web, base de datos, ... Pero el centro virtual ofrecerá como servicio extra a los comercios una página Web estándar, y el almacenamiento en su base de datos del catálogo de productos de dicho comercio.

- **¿Quién y desde dónde podrá acceder a la base de datos?**

El acceso a la base de datos es un tema *peligroso*, ya que cualquiera no puede acceder libremente a los datos almacenados, para solucionar esto se generarán una serie de *Factorías*, que serán objetos que permanecerán en el Servidor de Aplicación. Para acceder a dichos objetos de forma remota se utilizará CORBA, y serán estos objetos *Factoría* los únicos que puedan acceder a la base de datos utilizando JDBC.

De esta forma se puede configurar el sistema gestor de bases de datos para que sólo se pueda acceder a él desde las máquinas donde se encuentran dichas Factorías, facilitando de esta forma la configuración de dicho servidor para mantener la seguridad del centro comercial.

- **¿Cómo accederán los clientes a los datos?**

Como se ha indicado en el punto anterior, a los datos únicamente podrán acceder los objetos Factorías, por lo que el problema queda reducido a ¿Cómo accederán los clientes a los objetos factorías?.

Para acceder a los objetos factorías los clientes utilizarán CORBA, la única dificultad se encuentra en obtener una referencia inicial a dichas factorías, que como se describió en el tutorial, la mejor forma de obtener una referencia a un objeto servidor (IOR), es a través del servicio de nombres.

- **¿Cómo se le dará continuidad a la visita de un cliente por las distintas páginas Web, de forma que lo que introduzca en el carro de la compra en una, no desaparezca al pasar a otra?**

Cuando un cliente entra al centro comercial se genera un nuevo carro de la compra para dicho cliente, este carro de la compra se encuentra en el servidor de aplicaciones, de tal forma que el cliente se comunicará con el carro para introducir o sacar productos durante su visita a los comercios.

El problema se encuentra cuando el cliente pasa de una página Web a otra, ¿cómo sabe el applet de la nueva página Web el IOR del carro de la compra del cliente?

Para solucionar este problema se ha utilizado el protocolo CGI, de tal forma que se le pasa a un programa escrito en Perl el IOR del carro de la compra, y este programa genera una página Web donde el applet tiene un parámetro que es el IOR correspondiente.

Otra solución habría sido utilizar *PHP* junto con *cookies* para almacenar en el explorador Web el valor del IOR del carro de la compra, de tal forma que al acceder al servidor Web el propio navegador enviara de forma automática dicho IOR.

Se ha desarrollado un programa en Perl por cada página Web que debe recibir como parámetro el IOR del carro de la compra. Otra posibilidad sería tener un solo

programa en Perl que tuviera dos parámetros, el IOR y la página Web a mostrar. Ver figura 13.

Para ver una descripción paso a paso de un acceso al centro virtual ver el apartado **4.3**.

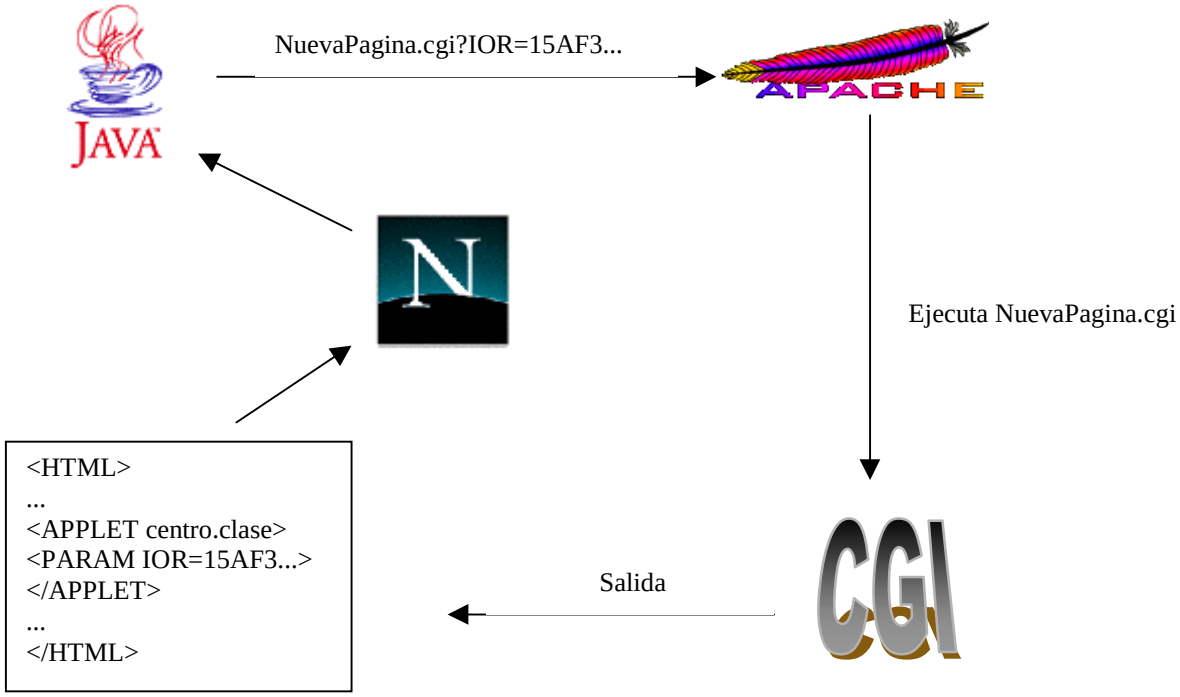

Figura 13 : Uso de CGI

4.1.6 Diagrama de despliegue

De las decisiones tomadas anteriormente puede desprenderse un diagrama de despliegue UML que describa la situación de las máquinas implicadas en el centro, cada uno de lo nodos que se encuentran a continuación es una computadora, o un conjunto de computadoras.

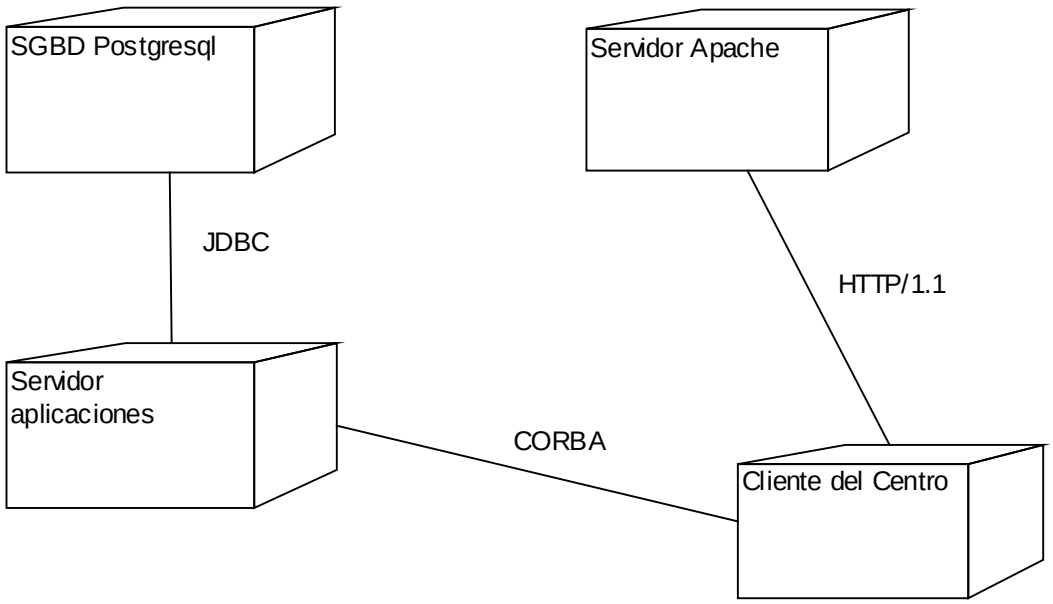

Figura 14 : Diagrama de despliegue del centro

4.2 Implementación

A continuación se describirá la aplicación desarrollada, siempre abordando los problemas relacionados con CORBA y la persistencia de los objetos remotos, en primer lugar se describirá la aplicación en Java.

4.2.1 Creación del IDL

Además de las clases presentadas en el diagrama de clases del apartado 4.1.3 se necesitan clases para representar a los objetos que se encuentren en el servidor de aplicaciones y que permiten a los programas cliente acceder a la base de datos. A estas clases se les llamará *FactoriaX*, donde X será la información que manejará dicha factoría. Se pueden encontrar dos factorías de este tipo, una para los comercios y otra para los socios.

Con toda la información obtenida durante el análisis, se pueden definir las siguientes interfaces para los objetos servidores:

```
module Centro {
      interface Internauta {
            attribute string nombre, apellido1, apellido2, telefono;
            attribute string correoElectronico, numeroCuenta;
            attribute string direccion;
      };
```

```
        interface Socio : Internauta {
              attribute string idSocio, clave;
        };

        interface PeriodoAvisos {
              attribute string tiempo;
              attribute boolean porPedido;
        };

        interface Producto {
              attribute string nombre, caracteristicas;
              attribute long long precio, precioOferta, idProducto;
        };

        typedef sequence<Producto> ListaProductos;

        interface Comercio {
              attribute string nombre, direccion, telefono1, telefono2;
              attribute string fax, correoElectronico, direccionWeb;
              attribute PeriodoAvisos refPeriodoAvisos;
              attribute ListaProductos refProducto;
        };

        struct ProductosCantidad {
              Producto prod;
              long cantidad;
              string nombreComercio;
        };
        typedef sequence<ProductosCantidad> ListaProductosCantidad;

        interface CarroCompra {
              typedef sequence<long> ListaCantidad;
              typedef sequence<string> ListaComercios;

              attribute Internauta refInternauta;
              attribute ListaProductosCantidad refProducto;

              void anadirProducto(in Producto prod, in long cantidad,
                                  in string nombreComercio);
              void listarProductos(out ListaProductos prod,
                                   out ListaCantidad cantidad,
                                   out ListaComercios comercios);
              void cambiarCantidad(in Producto prod, in string comercio,
                                    in long cantidad);
        };

        interface FactoriaSocio {
              CarroCompra validarSocio(in string id, in string clave);
              long altaSocio(in string idSocio, in string clave,
                             in string nombre, in string apellido1,
                             in string apellido2, in string telefono,
                             in string correoElectronico,
                             in string numeroCuenta, in string direccion);
        };

        interface FactoriaComercio {
              typedef sequence<string> ListaComercios;

              ListaComercios listarComercios();
              string direccionWeb(in string nombre);
              ListaProductos listarProductos(in string Comercio);
        };

};
```

Como se puede observar hay varias interfaces sin operaciones debido a que la funcionalidad para el acceso a la base de datos se ha concentrado en las factorías. Dichas interfaces podrían haber sido desarrolladas como *struct*, evitando de esta forma tener que implementar servidores para ellas, pero se han utilizado interfaces para mostrar la implementación de un mayor número de servidores, pudiendo proveer posteriormente a estos de cierta funcionalidad independiente del acceso a bases de datos.

Con este fichero (*CentroComercial.idl*) generamos automáticamente los stub's y los skeleton's utilizando la herramienta *idltojava*:

>idltojava –fno-cpp –j .. CentroComercial.idl

Del problema se ha implementado la parte necesaria y suficiente para mostrar cómo utilizar las distintas tecnologías involucradas, y como interactuan éstas. Se ha omitido la implementación de los siguientes aspectos:

- Acceso por internautas que no sean socios.
- Generación de facturas.
- Altas de empresas.

4.2.2 Implementación de los servidores

Las interfaces que han sido implementadas son las siguientes:

Socio
Producto
Internauta
FactoriaSocio
FactoriaComercio
CarroCompra

En el caso de las tres primeras la implementación es casi inmediata ya que no contienen ningún método. Veremos como ejemplo el caso del *Producto*, para el que la implementación será la siguiente:

ProductoImpl.java

```
1. package Centro;

2. public class ProductoImpl extends _ProductoImplBase {
3.     String nombre, caracteristicas;
4.     long precio, precioOferta, idProducto;

5.     public ProductoImpl(long idProducto, String nombre, long precio,
                    String caracteristicas, long precioOferta) {
6.           this.idProducto = idProducto;
7.           this.nombre = nombre;
8.           this.precio = precio;
9.           this.caracteristicas = caracteristicas;
10.          this.precioOferta = precioOferta;
11.    }
```

```
12.    public String nombre() {
13.           return nombre;
14.    }
15.    public void nombre(String arg) {
16.           nombre = arg;
17.    }
18.    public String caracteristicas() {
19.           return caracteristicas;
20.    }
21.    public void caracteristicas(String arg) {
22.           caracteristicas = arg;
23.    }
24.    public long precio() {
25.           return precio;
26.    }
27.    public void precio(long arg) {
28.           precio = arg;
29.    }
30.    public long precioOferta() {
31.           return precioOferta;
32.    }
33.    public void precioOferta(long arg) {
34.           precioOferta = arg;
35.    }
36.    public long idProducto() {
37.           return idProducto;
38.    }
39.    public void idProducto(long arg) {
40.           idProducto = arg;
41.    }
42. }
```

La interfaz debe heredar de la clase *_ProductoImplBase* (Línea 2) para que de esta forma obtenga la funcionalidad de un servidor que utiliza CORBA (utiliza el *skeleton* para permitir los accesos remotos).

Para cada atributo de la interfaz *IDL Producto* se define un atributo en la clase *ProductoImpl* que será el encargado de almacenar el valor correspondiente (Líneas 3 y 4), además se implementan dos métodos por cada atributo definido en la interfaz. Estos atributos se definen en la interfaz *Java Producto*. Por ejemplo para el atributo *precio* se define el método *long precio()* (Línea 24) que se utiliza para acceder al valor del atributo, y el método *void precio(long arg)* (Línea 27) utilizado para modificar el valor del atributo *precio*.

Las interfaces *Internauta* y *Socio* se implementan de forma similar. Únicamente destacar que para implementar *Socio* que hereda de *Internauta* hay que definir los atributos y métodos tanto de *Socio* como de la interfaz de la que hereda: *Internauta*.

El resto de interfaces son factorías que acceden a la base de datos del centro virtual, y para ello utilizan la clase *ConstantesServidor* que define una serie de constantes para el servidor del centro, útiles al resto de las clases:

ConstantesServidor.java

```
1. package Centro;

2. import java.sql.*;
3. import org.omg.CosNaming.*;

4. public class ConstantesServidor {
5.     static final String servidor = "192.168.105.1";
```

```
6.     static final String nombreBaseDatos = "centrovirtual";
7.     static final String usuario = "jros";
8.     static final String claveUsuario = "";
9.     static Connection baseDatos;
10.    static Statement consultas;
11.    static NamingContext productosNaming;

12.    static {
13.           try {
14.                  Class.forName("postgresql.Driver");
15.                  baseDatos = DriverManager.getConnection(
                            "jdbc:postgresql://" + servidor + "/" +
                            nombreBaseDatos, usuario, claveUsuario);
16.                  consultas = baseDatos.createStatement();
17.           } catch (Exception ex) {
18.                  ex.printStackTrace();
19.           }
20.    }
21. }
```

En esta clase se definen las siguientes constantes:

- *servidor*: Corresponde con la dirección IP del servidor de bases de datos, o el nombre de DNS de dicho servidor.
- *nombreBaseDatos*: Nombre de la base de datos dentro del servidor.
- *usuario*: Nombre del usuario con el que se accederá a la base de datos.
- *claveUsuario*: Clave del usuario en el servidor de bases de datos.
- *baseDatos*: Conexión con el servidor de bases de datos.
- *consultas*: Objeto del tipo *java.sql.Statement*, que se utiliza para realizar peticiones de datos, inserciones, o modificaciones al sistema gestor de bases de datos.
- *productosNaming*: Contexto correspondiente con el centro virtual.

En el constructor de clase (Línea 12) se inicializan las variables para el acceso a la base de datos, para ello se utiliza *JDBC* para acceder a una base de datos *PostgreSql*.

Lo primero que se debe hacer para acceder a la base de datos es cargar la clase que contenga el driver para el acceso a la base de datos, en este caso *postgresql.Driver* (Línea 14), es aconsejable utilizar las clases generadas por *postgresql* para las *JDK1.2* que se pueden obtener para la versión 6.5 del sistema gestor de bases de datos.

A continuación es posible establecer una conexión con la base de datos, para ello se utiliza el método *getConnection* de la clase *DriverManager* (Línea 15). A este método se le deben pasar como parámetros, una cadena de conexión, el nombre del usuario y la contraseña con los que se accederá a la base de datos.

La cadena de conexión está formada por las siguientes partes en el caso de una conexión con un sistema *PostgreSql*:

```
jdbc:postgresql://nombreServidor/nombreBaseDatos
```

En el caso de Oracle la cadena sería de la siguiente forma:

```
jdbc:oracle:thin:usuario/password@servidor:puerto:SIDoracle
```

Una vez establecida la conexión con la base de datos se crea un nuevo objeto de la clase *java.sql.Statement* para enviar las sentencias SQL a la base de datos, para ello se utiliza el método *createStatement* de la clase *Connection.*

La interfaz *FactoriaSocio* se implementa como se muestra a continuación:

FactoriaSocioImpl.java

```
1. package Centro;

2. import java.sql.*;
3. import org.omg.CORBA.*;

4. public class FactoriaSocioImpl extends _FactoriaSocioImplBase {

5.     public CarroCompra validarSocio(String id, String clave) {
6.           try {
                    // Comienza la validación
6.     ResultSet resultado =
             ConstantesServidor.consultas.executeQuery (
             "SELECT clave FROM socio WHERE idSocio='" + id + "';" );

             // Consulta realizada

             // Lo primero se comprueba que hay alguna fila en la tabla
             // correspondiente a este socio, es decir que esté dado de
             // alta

7.                 if (! resultado.next()) {
                          // No existe tal socio
8.                   System.out.println("No ha devuelto ninguna fila");
9.                   return null;
10.                }

             // Si el socio existe se comprueba que la clave es correcta
11.                String claveVerdadera =
                                    resultado.getString("clave");
12.                if (!clave.equals(claveVerdadera)) {
13.                      System.out.println("Clave erronea");
                         // La clave no es correcta
14.                      return null;
15.                }

     // Tras la validación se crea un carro de la compra para este socio
16.                CarroCompraImpl carro = new CarroCompraImpl();
17.                String args[] = {};
18.                ORB orb = ORB.init(args, null);
19.                orb.connect(carro);

                   // Se crea el socio que estará asociado al carro
20.                SocioImpl socio = new SocioImpl();
21.                socio.idSocio(id);
22.                socio.clave(clave);

                   // Se asocia el socio al carro
23.                carro.refInternauta(socio);

                   // Se devuelve el carro de la compra
24.                return carro;
25.          } catch (SQLException ex) {
26.                ex.printStackTrace();
27.          }
28.          return null;
29.    }

30.    public int altaSocio(String idSocio, String clave, String nombre,
             String apellido1, String apellido2, String telefono,
```

```
                String correoElectronico, String numeroCuenta,
                String direccion) {

31.             try {
32.               return ConstantesServidor.consultas.executeUpdate(
                       "INSERT INTO Socio VALUES ('" + idSocio + "','" +
                       clave + "','" + nombre + "','" + apellido1 + "','" +
                       apellido2 + "','" + telefono + "','" +
                       correoElectronico + "','" + numeroCuenta + "','" +
                       direccion + "');");

33.             } catch (Exception ex) {
34.                   ex.printStackTrace();
35.                   return 0;
36.             }

37.    }

38. }
```

En esta clase se implementan los dos métodos definidos en la interfaz *FactoriaSocio*, que se utilizan para acceder a la base de datos del centro virtual, más concretamente para comprobar que un socio existe y que su clave es correcta, y para dar de alta a un nuevo socio del centro comercial.

El primer método (*validarSocio* en la línea 5) se encarga de acceder a la base de datos para comprobar que un login al sistema ha sido realizado de forma correcta.

Para realizar la comprobación lo primero que se hace es obtener la clave para el identificador del socio que ha tratado de entrar en el centro virtual, para ello se accede a la base de datos utilizando el objeto de la clase *Statement* que se creó en la clase *ConstantesServidor*, del cual se utiliza el método *executeQuery* para realizar una consulta SQL (Línea 6).

A continuación se comprueba que el socio exista, es decir que de la consulta anterior obtenga algún resultado (Línea 7). Si el socio existe se comprueba que la clave introducida coincide con la almacenada en la base de datos (Línea 12).

Una vez que el socio se ha validado ante el centro virtual se crea un carro de la compra (Línea 16), al que se le asocia un nuevo objeto socio que coincide con el socio obtenido de la base de datos (Línea 23). Al ORB se le conecta el carro de la compra (Línea 19) para que se le puedan realizar peticiones remotas, permitiendo así a los clientes añadir productos en su carro.

Finalmente se le devuelve al cliente de la factoría el carro creado para él (Línea 24). Si la autentificación falla se devolverá *null* para indicarlo.

El otro método (*altaSocio* en la línea 30), permite la realización del alta de un socio en el centro virtual. Este método realiza una inserción en la base de datos a partir de los datos que se le pasan como parámetros (Línea 32).

La interfaz *FactoriaComercio* se ha implementado de la siguiente forma:

FactoríaComercio.java

```
1. package Centro;
```

```
2. import java.sql.*;
3. import java.util.*;

4. public class FactoriaComercioImpl extends _FactoriaComercioImplBase {
5.     public String[] listarComercios() {
6.           try {
       // Se comienza realizando la consulta de los comercios existentes
7.     ResultSet resultado =
                    ConstantesServidor.consultas.executeQuery(
                    "SELECT nombre FROM comercio;");

                    // Consulta realizada
8.                  Vector comercios = new Vector();

             // A continuación se toma el nombre de todos los comercios
9.                  while (resultado.next())
10.                       comercios.add(resultado.getString("nombre"));

11.                 Object aux[] = comercios.toArray();
12.                 String aCom[] = new String[aux.length];
13.                 for (int i=0;i<aux.length;i++)
14.                       aCom[i]=(String)aux[i];

                    // Se devuelven los comercios encontrados
15.                 return aCom;
16.          } catch (SQLException ex) {
17.                 ex.printStackTrace();
18.          }
19.          return null;
20.   }

21.   public String direccionWeb(String nombre) {
22.          try {
             // Se comienza realizando la consulta del comercio pedido
23.                 ResultSet resultado =
                          ConstantesServidor.consultas.executeQuery(
                          "SELECT direccionWeb FROM comercio WHERE
                          nombre='" + nombre + "';");

                    // Consulta realizada
24.                 if (!resultado.next()) return null;

                    // Se devuelve la dirección correspondiente

25.                 return resultado.getString("direccionWeb");

26.          } catch (SQLException ex) {
27.                 ex.printStackTrace();
28.          }
29.          return null;
30.   }

31.   public Producto[] listarProductos(String Comercio) {
32.          Producto listaProductos[] = {};
33.          try {
 // Se comienza realizando la consulta de los productos de este comercio
34.                 ResultSet resultado =
                          ConstantesServidor.consultas.executeQuery(
                          "SELECT * FROM Producto WHERE idproducto IN
                          (SELECT" + " idproducto FROM
                          comercio_producto WHERE nombre_comercio = '"
                          + Comercio + "');");

                    // Consulta realizada
35.                 Vector productos = new Vector();

             // A continuación se toma el nombre de todos los comercios
36.                 while (resultado.next()) {
```

```
                        ProductoImpl nuevoProducto =
                    new ProductoImpl(resultado.getLong("idproducto"),
                    resultado.getString("nombre"),
                    resultado.getLong("precio"),
                    resultado.getString("caracteristicas"),
                    resultado.getLong("preciooferta"));

37.                     productos.add(nuevoProducto);
38.                 }
39.                 Object aux[] = productos.toArray();
40.                 listaProductos = new Producto[aux.length];
41.                 for (int i=0;i<aux.length;i++)
42.                     listaProductos[i]=(Producto)aux[i];

                    // Se devuelven los comercios encontrados
43.                 return listaProductos;
44.           } catch (SQLException ex) {
45.                 ex.printStackTrace();
46.           }


47.           return listaProductos;
48.    }
49. }
```

Con esta clase se pueden listar los comercios inscritos en el centro virtual, obtener la dirección Web correspondiente a un comercio, y obtener la lista de productos de un comercio si este utiliza la base de datos del centro virtual.

Para listar los comercios (Línea 4) se obtienen los nombres de la base de datos, estos se guardan en un *Vector* que posteriormente se transforma en un Array, ya que este es el tipo que se define al transformar el IDL a Java.

El método que obtiene la dirección Web (Línea 21) de un comercio es una simple consulta a la base de datos donde se encuentra esta información.

Para obtener la lista de Productos (Línea 31) se consulta la base de datos obteniendo todos los productos, que se devuelven al cliente en un Array de productos.

Finalmente la interfaz *CarroCompra* es implementada de la siguiente forma:

CarroCompraImpl.java

```
1. package Centro;

2. import java.util.*;
3. import Centro.CarroCompraPackage.*;

4. public class CarroCompraImpl extends _CarroCompraImplBase {
5.     Internauta refInternauta;
6.     ArrayList refProducto = new ArrayList();

7.     public Internauta refInternauta() {
8.           return refInternauta;
9.     }

10.    public void refInternauta(Internauta arg) {
11.          refInternauta = arg;
12.    }

13.    public ProductosCantidad[] refProducto(){
14.          return (ProductosCantidad[]) refProducto.toArray();
```

```
15.    }

16.    public void refProducto(ProductosCantidad[] arg) {
17.           refProducto.addAll(Arrays.asList(arg));
18.    }
19.    public void anadirProducto(Centro.Producto prod, int cantidad,
                                        String nombreComercio) {
              // Al añadir un producto se mantendrá la lista ordenada
              // alfabéticamente por nombre de producto
20.           if (cantidad<0) return;
                    // No se tratan cantidades menor que 0
21.           for (int i=0; i<refProducto.size(); i++) {
22.                 ProductosCantidad act =
                                  (ProductosCantidad) refProducto.get(i);
23.                 if (act.prod.nombre().compareTo(prod.nombre())<0)
24.                       continue;
25.                 if (act.prod.nombre().compareTo(prod.nombre())>0) {
26.                       ProductosCantidad nuevo =
                    new ProductosCantidad(prod,cantidad,nombreComercio);
27.                       refProducto.add(i,nuevo);
28.                       return;
29.                 }
30.                 if (act.prod.idProducto()==prod.idProducto() &&
                          act.nombreComercio.equals(nombreComercio)) {
                    // Si es el mismo producto del mismo comercio sólo
                    // se aumenta la cantidad de ese producto
31.                       act.cantidad+=cantidad;
32.                       return;
33.                 }
34.           }
              // Si no hay ningún producto aún lo añado al principio
35.                 ProductosCantidad nuevo =
              new ProductosCantidad(prod,cantidad,nombreComercio);

36.           refProducto.add(nuevo);

37.    }

38.    public void listarProductos(ListaProductosHolder prod,
                    ListaCantidadHolder cantidad,
                    ListaComerciosHolder comercios) {
39.           prod.value = new Producto[refProducto.size()];
40.           cantidad.value = new int[refProducto.size()];
41.           comercios.value = new String[refProducto.size()];

42.           for (int i=0; i<refProducto.size(); i++) {
43.                 ProductosCantidad act =
                          (ProductosCantidad) refProducto.get(i);
44.                 prod.value[i] = act.prod;
45.                 cantidad.value[i] = act.cantidad;
46.                 comercios.value[i] = act.nombreComercio;
47.           }
48.    }

49.    public void cambiarCantidad(Producto prod, String comercio,
                                  int cantidad) {
50.           if (cantidad<0) return;
                    // No se tratan cantidades menor que 0

51.           for (int i=0; i<refProducto.size(); i++) {
52.                 ProductosCantidad act =
                                  (ProductosCantidad) refProducto.get(i);
53.                 if (act.prod.idProducto()==prod.idProducto() &&
                          act.nombreComercio.equals(comercio)) {
     // Si es el mismo producto del mismo comercio se cambia la cantidad
       // Si la cantidad es 0 se quita el producto
54.                       if (cantidad==0) {
55.                             refProducto.remove(i);
```

```
56.                                 return;
57.                           }
58.                           act.cantidad=cantidad;
59.                           return;
60.                     }
60.               }
62.     }
63. }
```

Esta interfaz define dos atributos (Líneas 5 y 6), para almacenar el Internauta dueño de este carro, y la lista de productos que se almacenan en el carro.

Además de dichos atributos se definen tres métodos para el manejo del carro de la compra: inserciones, listados y modificaciones.

En primer lugar está el método *anadirProducto* (Línea 19) que servirá para añadir un nuevo producto en el carro de la compra. Para mantener los productos en el carro de la compra se utiliza una lista, en esta lista se mantienen los productos ordenados de tal forma que la búsqueda de un producto se acelere. Al añadir un nuevo producto, si éste existe se aumenta la cantidad almacenada, en otro caso se añade a la lista de tal forma que ésta permanezca ordenada.

El método *listarProductos* (Línea 38) obtiene una lista de cada producto con la cantidad que se ha seleccionado y el comercio del que se ha seleccionado.

Finalmente el método *cambiarCantidad* (Línea 49) cambia la cantidad seleccionada de un producto, la diferencia con *anadirCantidad* es que en *cambiarCantidad* se puede reducir el número de productos pero en el otro sólo se pueden añadir al carro de la compra.

Para lanzar el servidor en una máquina se utiliza la clase *ServidorCentro*, encargado de asociar las factorías con un nodo del servicio de nombres:

ServidorCentro.java

```
package Centro;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.io.*;

public class ServidorCentro {

       public static void main (String args[]) {
              // Creo un nuevo ORB
              ORB orb = ORB.init(args, null);

         // Se va a incluir la factoría de socios en el árbol de nombres.

              // En primer lugar se obtiene el contexto raíz
              NamingContext rootContext=null;
              try {
                     org.omg.CORBA.Object objRef =
                           orb.resolve_initial_references("NameService");
                     rootContext = NamingContextHelper.narrow(objRef);
              } catch (org.omg.CORBA.ORBPackage.InvalidName ex) {
                     System.err.println("Error, no se encuentra
                                                 el servicio de nombres");
                     System.exit(-1);
```

```
                }

            try {

                // A continuación se avanza por la rama del CentroVirtual
                NameComponent centroContext[] = new NameComponent[1];

                centroContext[0] = new NameComponent("CentroVirtual", "");

                NamingContext servidores;

                try {
                    // Se prueba a ver si existe
                    servidores =
                                NamingContextHelper.narrow(rootContext.
                                        resolve(centroContext));
                } catch (org.omg.CosNaming.NamingContextPackage.NotFound
                                ex) {
                    // Si no existe se crea
                    servidores =
                            rootContext.bind_new_context(centroContext);
                }

                // Se crea la rama para los productos
                centroContext[0] = new NameComponent("Productos", "");
                try {
                    // Se prueba a ver si existe
                    ConstantesServidor.productosNaming =
                                NamingContextHelper.narrow(rootContext.
                                        resolve(centroContext));
                } catch (org.omg.CosNaming.NamingContextPackage.NotFound
                            ex) {
                    // Si no existe se crea
                    ConstantesServidor.productosNaming =
                            rootContext.bind_new_context(centroContext);
                }

                // Se crea una factoría de socios
                FactoriaSocioImpl factSocio = new FactoriaSocioImpl();
                orb.connect(factSocio);

                // Y se asocia la factoría con un nombre.
                centroContext[0] = new NameComponent("FactoriaSocio", "");

                servidores.rebind(centroContext, factSocio);

                // Se crea una factoría de comercios
                FactoriaComercioImpl factComercio =
                            new FactoriaComercioImpl();
                orb.connect(factComercio);

                // Y se asocia la factoría con un nombre.
                centroContext[0] =
                    new NameComponent("FactoriaComercio", "");

                servidores.rebind(centroContext, factComercio);


                System.out.println("Listo");

                // Se espera a que se pulse enter para terminar la
                // ejecución del servidor
                try {
                    BufferedReader teclado;
                    teclado = new
                    BufferedReader(new InputStreamReader(System.in));
                    String linea=teclado.readLine();
                } catch (IOException ex) {}
```

```
                servidores.unbind(centroContext);

        } catch (org.omg.CORBA.UserException ex) {
                System.err.println("Excepción no esperada,
                                        finalización anormal");
                ex.printStackTrace();
        }

        }
}
```

4.2.3 Implementación de los clientes

En el servidor Web deben publicarse una serie de páginas Web que corresponderán con las accedidas por los clientes del centro virtual. En ellas el cliente podrá obtener la funcionalidad que aportan los servidores implementados en el apartado anterior.

Un cliente que entre en el centro virtual puede darse de alta, o si ya es un socio podrá realizar una visita autentificándose en primer lugar. Una vez dentro del centro virtual se podrán recorrer los diferentes comercios para obtener los productos deseados de cada uno de ellos, ver el contenido del carro de la compra, o modificar el contenido de éste.

Para ser utilizada por todos los clientes se ha desarrollado una clase para acceder al servicio de nombres, que es la funcionalidad común a todos ellos:

ConstantesCliente.java

```
package Centro;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class ConstantesCliente {
       static NamingContext obtenerNamingContextCentro
                                (java.applet.Applet ap) {
                // Se crea un nuevo ORB

                ORB orb = ORB.init(ap, null);

                // En primer lugar se obtiene el contexto raíz
                NamingContext rootContext=null;
                try {
                        org.omg.CORBA.Object objRef =
                        orb.resolve_initial_references("NameService");
                        rootContext = NamingContextHelper.narrow(objRef);
                } catch (org.omg.CORBA.ORBPackage.InvalidName ex) {
                        System.err.println("Error, no se encuentra el
                                                servicio de nombres");
                }

        NamingContext centroNamingContext=null;

        try {

                // A continuación se avanza por la rama del CentroVirtual
                NameComponent centroContext[] = new NameComponent[1];
```

```
                centroContext[0] = new NameComponent("CentroVirtual", "");

                try {
                       // Pruebo a ver si existe
                       centroNamingContext =
                                    NamingContextHelper.narrow(rootContext.
                                                  resolve(centroContext));
                } catch (org.omg.CosNaming.NamingContextPackage.NotFound
                                    ex) {
                       // Si no existe error
                       System.err.println("Error, no hay ningún Centro
                                                  Virtual funcionando");
                }


        } catch (org.omg.CORBA.UserException ex) {
                System.err.println("Excepción no esperada,
                                              finalización anormal");
                ex.printStackTrace();
        }
        return centroNamingContext;

        }
}
```

Veamos como ejemplo los pasos para el alta de un socio.

En primer lugar se desarrolla el applet que será el encargado de preguntar al usuario los datos necesarios para realizar el alta, pasando estos parámetros a la factoría correspondiente que será la encargada de realizar el alta en la base de datos.

El código para dicho applet sería el siguiente:

AltaSocio.java

```
1. package Centro;

2. import java.awt.*;
3. import java.awt.event.*;
4. import java.applet.*;
5. import javax.swing.*;
6. import org.omg.CosNaming.*;
7. import java.net.*;

8. public class AltaSocio extends JApplet {
       // Atributos que define la interfaz gráfica del applet (botones,
       // etiquetas y cuadros de texto)

9.   public String getParameter(String key, String def) {
10.    return isStandalone ? System.getProperty(key, def) :
11.      (getParameter(key) != null ? getParameter(key) : def);
12.  }


13.  public AltaSocio() {
14.  }
// Inicializa el applet

15.  public void init() {
16.    try {
17.    jbInit();
18.    }
19.    catch (Exception e) {
20.    e.printStackTrace();
```

```
21.    }
22.  }

23.  private void jbInit() throws Exception {
       // Se inicializan las posiciones de los componentes gráficos y
       // el comportamiento de los botones
24.  }

25.  public String getAppletInfo() {
26.    return "Applet para el alta de un socio";
27.  }

28.  public String[][] getParameterInfo() {
29.    return null;
30.  }

31.  void btnAceptar_actionPerformed(ActionEvent e) {
             // Se ha pulsado el botón aceptar y se debe dar de alta un
             // nuevo socio
32.    FactoriaSocio factoria=null;

33.    try {
34.          NameComponent factoriaName[] = new NameComponent[1];
35.          factoriaName[0]=new NameComponent("FactoriaSocio","");
36.          org.omg.CORBA.Object factoriaObj =
37.                ConstantesCliente.obtenerNamingContextCentro(this).
                        resolve(factoriaName);
38.          factoria = FactoriaSocioHelper.narrow(factoriaObj);
39.    } catch (org.omg.CORBA.UserException ex) {
40.          System.err.println("Excepción no esperada");
41.          ex.printStackTrace();
42.    }

43.    try {
44.      if (factoria.altaSocio(txtIdentificador.getText(),
             new String(txtClave.getPassword()),txtNombre.getText(),
             txtApellido1.getText(), txtApellido2.getText(),
             txtTelefono.getText(), txtEmail.getText(),
             txtNumeroCuenta.getText(), txtDireccion.getText())>0)

45.                getAppletContext().showDocument(new
                         URL(getCodeBase().
                                toString()+"AltaSocioRealizada.html"));
46.      else
47.                getAppletContext().showDocument(new
                         URL(getCodeBase().
                                toString()+"AltaSocioError.html"));
48.    } catch (MalformedURLException ex) {
49.          ex.printStackTrace();
50.    }
51.  }

52.  void btnCancelar_actionPerformed(ActionEvent e) {
             // Se ha pulsado el botón cancelar
53.    try {
54.          getAppletContext().showDocument(new
                   URL(getCodeBase().toString()+"Internauta.html"));
55.    } catch (MalformedURLException ex) {
56.          ex.printStackTrace();
57.    }
58.  }
59. }
```

Este applet tiene el aspecto que se puede ver en la figura 15, cuando ha sido cargado por un navegador.

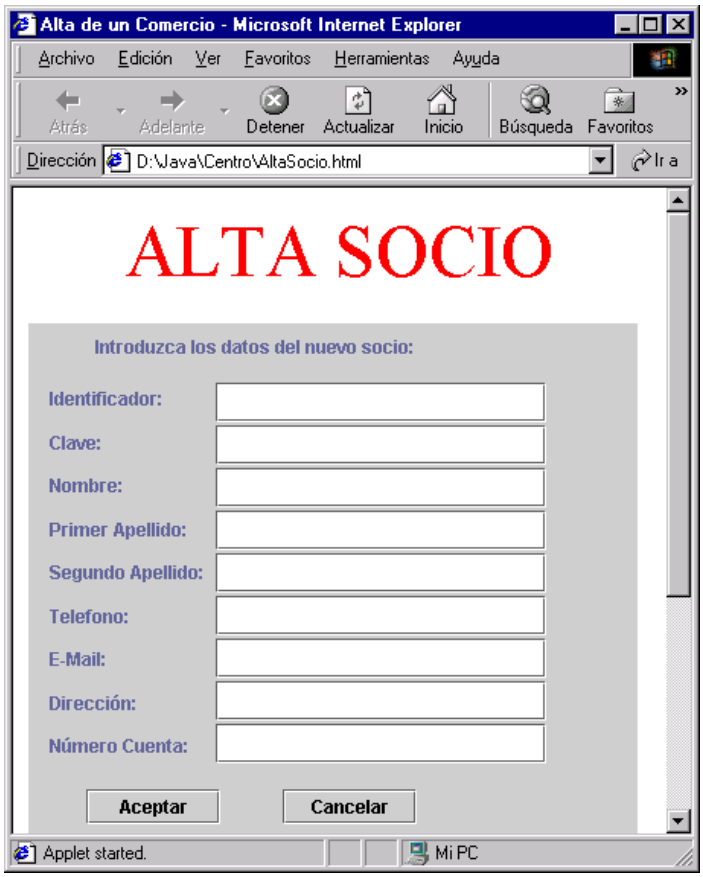

Figura 15 : Applet para el alta de un socio

En los diferentes cuadros de texto se introducen los datos del socio que se desea crear, una vez realizado esto se pulsa el botón aceptar.

Al pulsar el botón aceptar (Línea 31), lo primero que se hace es obtener una referencia a la factoría de socios (Líneas 32 a 42). La factoría de socios se encuentra asociada al contexto del centro virtual, con lo que basta con obtener una referencia a dicho contexto (Línea 37) y a partir de esta obtener la referencia a la factoría (Línea 36).

La referencia al contexto se obtiene utilizando una clase *contantesCliente* donde se encuentra el método que obtiene el *NamingContext* correspondiente al centro, tal como se explicó en el tutorial.

Una vez obtenida la factoría, basta con utilizar el método *anadirSocio* de dicha factoría para crear un nuevo socio (Línea 44).

Si el alta es realizada se muestra una página Web que indica que el alta se ha realizado correctamente (Línea 45), si no se puede dar de alta se muestra una página Web que indica esta situación (Línea 47).

4.3 Funcionamiento del Centro

Un internauta que desee convertirse en socio del centro virtual, deberá visitar en primer lugar la página principal del centro (http://www.servidorApache.com) y tras seleccionar los hipervínculos adecuados accederá a la página mostrada en la figura 15. Una vez introducidos los datos necesarios pasará a formar parte del centro virtual, siendo identificado por el *Identificador* y la *clave*.

Cuando un socio quiere realizar una compra entra en el centro virtual y accede a la página que se muestra en la figura 16 .

Figura 16 : Applet para la identificación de un socio

Donde debe introducir el identificador y la clave facilitada al darse de alta en el centro comercial. Cuando un internauta pulsa el botón aceptar de esta ventana se produce una invocación remota al método *validarSocio* de la clase *FactoriaSocio* que accederá a la base de datos para comprobar si los datos introducidos corresponden con algún socio del centro, en cuyo caso se genera un nuevo carro de la compra para dicho socio, el cual se devuelve al applet. El applet (en caso de ser positiva la identificación) muestra la página Web correspondiente a la URL:

http://servidorApache/cgi-bin/SeleccionCentro.cgi?IOR=IORdelCarro

El programa *SeleccionCentro.cgi* generará una página Web que se visualizará en el navegador como se muestra en la figura 17.

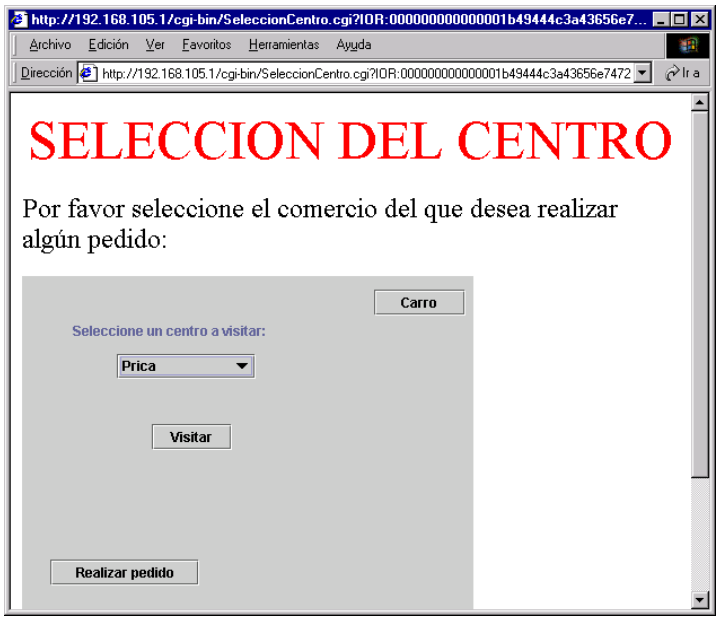

Figura 17 : Applet para la selección de un centro

El applet al inicializarse accede a la factoría de centros para obtener una lista de éstos. Cuando el socio selecciona un centro se visualiza la página Web correspondiente a dicho comercio, la cual se almacena en la base de datos, y es proporcionada por cada comercio al darse de alta en el centro virtual. Si el comercio seleccionó la opción de utilizar la página Web estándar ofrecida por el centro comercial virtual se llamará al CGI *nombreCentro.cgi* teniendo como parámetro el IOR del carro de la compra, el aspecto del browser será el de la figura 18.

Figura 18 : Applet para la selección de productos

Para ilustrar el proceso de compra puede verse en la figura 19 el diagrama de secuencia obtenido a partir de la implementación para inicializar el applet de la figura 17, y en la figura 20 el diagrama de secuencia para añadir un producto al carro de la compra.

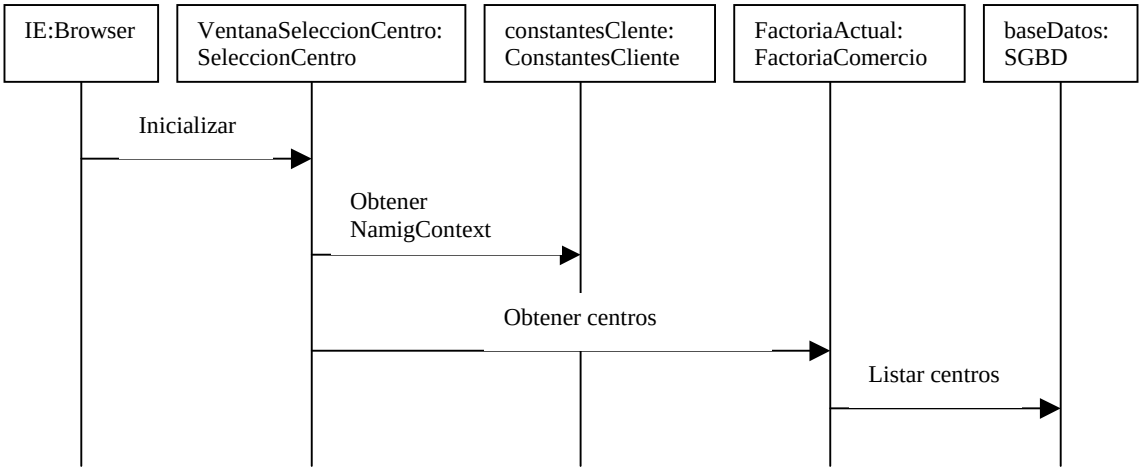

Figura 19 : Diagrama de secuencia para la obtención de la lista de centros

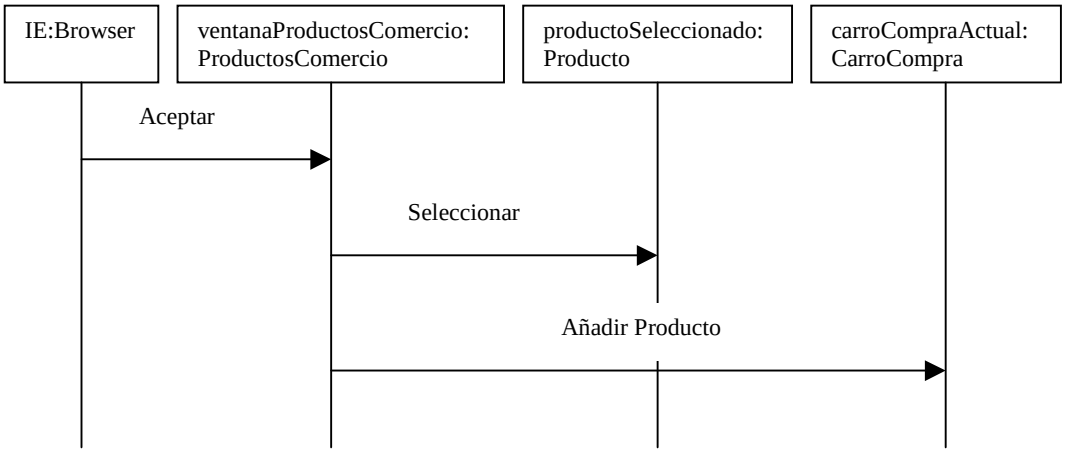

Figura 20 : Diagrama de secuencia para añadir un nuevo producto

4.4 Implementación en C++

Tras implementar el centro comercial virtual en Java, se decidió realizar la misma implementación en C++ para contrastar ambos lenguajes, y demostrar como el mismo programa puede ser realizado indistintamente en uno u otro, pudiendo mezclar posteriormente ambas implementaciones, es decir utilizando un servidor implementado

en cualquiera de los dos lenguajes, y los clientes de ambas implementaciones de forma indistinta.

Una ventaja de la aplicación en C++ es que el servidor en este lenguaje es más rápido que el mismo servidor en Java, ya que la aplicación no es interpretada. Por otro lado el servidor en Java es multiplataforma pudiendo ser portado a cualquier combinación de *hardware+sistema operativo* en el que se encuentren las *JDK1.2*. Por otro lado el cliente en C++ tiene menos problemas al no depender del navegador (si el navegador tiene algún *bug* el cliente no funciona), aunque debe ser instalado en el ordenador en el que se desee utilizar, lo cual no es necesario en el caso de utilizar un *applet*.

Para implementar el programa en C++ se ha utilizado *ORBacus* de la misma forma que se hizo en el capítulo 3. Para acceder a *postgresql* desde el programa se utilizan las librerías *pq++* que son distribuidas junto con el sistema gestor de bases de datos, debe tenerse en cuenta que se han utilizado las librerías que acompañan a la versión 6.5 de dicho gestor de bases de datos, las cuales no son compatibles con versiones anteriores.

Para la realización del cliente del centro comercial, se han utilizado las librerías gráficas *QT* desarrolladas por *Troll Tech,* las cuales tienen una licencia de uso gratuito si la aplicación desarrollada no tiene fines comerciales. Estas librerías están implementadas el lenguaje de programación C++, y pueden ser utilizadas en Linux, aunque con la licencia profesional (que no es gratuita) pueden ser utilizadas también en Windows.

El programa ha sido desarrollado siguiendo los mismos pasos, y con las mismas consideraciones que las indicadas en los capítulos 2 y 3, el código fuente puede encontrarse en el Apéndice.

Cabe destacar únicamente la implementación del *Socio*, ya que en Java el hecho de que heredara de *Internauta* no implicaba nada, es decir había que implementar de nuevo todos los métodos que se definieran en la clase base (En este caso para el acceso a los atributos de la interfaz).

En C++, al permitir el lenguaje herencia múltiple, se soluciona el problema, siendo la implementación del *Socio* la siguiente:

SocioImpl.h

```
1. #ifndef SOCIOIMPL_H
2. #define SOCIOIMPL_H

3. #include <OB/CORBA.h>
4. #include "CentroComercial_skel.h"
5. #include "InternautaImpl.h"

6. class Centro_Socio_Impl :
7.     public Centro_Socio_skel,
8.     public Centro_Internauta_Impl
9. {

10.    CORBA_String_var idSociovar, clavevar;

11. public:
```

```
12.    char *idSocio();
13.    void idSocio(const char *arg);
14.    char *clave();
15.    void clave(const char *arg);
16. };

17. #endif
```

Como se puede ver en este caso la clase que implementa la interfaz *Socio* hereda de dos clases, la primera (Línea 7) es la clase *skeleton* para la interfaz, generada automáticamente por el compilador de *IDL* de *ORBacus*, la segunda (Línea 8) es la implementación que se ha realizado para la interfaz *Internauta*, de tal forma que sólo es necesario implementar las operaciones definidas en la interfaz *Socio* y no las definidas en la interfaz *Internauta* como sucedía en el caso de Java.

Puede verse en la figura 21 el aspecto de una ventana en el cliente realizado en C++, y como se puede apreciar es muy similar al de la aplicación en Java.

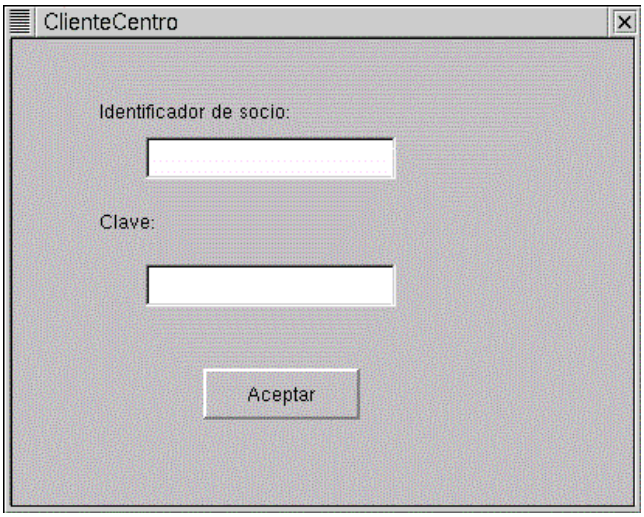

Figura 21 : Alta Socio en C++

5. Objetos persistentes

5.2 Introducción

Durante el desarrollo del centro comercial virtual en el capítulo 4, nos encontramos con una serie de clases que únicamente contenían datos, y cuyos objetos tenían su representación persistente en un gestor de bases de datos, como era *postgresql* en nuestro caso. Cada vez que se quería acceder a un objeto de una de estas clases se accedía a la factoría correspondiente, la cual generaba un nuevo objeto remoto a partir de los datos que se encontraban almacenados en la base de datos, esto conlleva una serie de problemas:

- Se generará más de un objeto con los mismos datos, con el consiguiente gasto innecesario de recursos en el servidor, y un gran número de accesos a la base de datos.

- Los objetos no son dados de baja una vez han dejado de utilizarse.

Para solucionar estos problemas se ha desarrollado una estrategia para la programación de aplicaciones en las que aparezcan *objetos persistentes*, es decir objetos que representen tuplas en las tablas de una base de datos.

Al desarrollar dicha estrategia se observó que la generación de clases necesaria para conseguir esos *objetos persistentes* era idéntica para cualquier aplicación, por lo que se desarrolló un compilador que a partir de una especificación en un subconjunto de *IDL* genera las clases necesarias para el servidor.

La estrategia se basa en tener una caché de objetos conectados al ORB, cuyos datos han sido obtenidos de la base de datos. Cuando se hace una petición de un objeto, se buscará dicho objeto en la caché, en caso de encontrarse en ésta se devolverá sin necesidad de acceder a la base de datos ni de generar un nuevo objeto en el servidor, si no, se crea el objeto remoto almacenándolo en la caché. Cuando la caché se llena se irán liberando los objetos que no se utilizan (utilizando algún algoritmo de reemplazo).

Para poder realizar esto será necesario que cada interfaz tenga una *clave* que identifique de forma única a cada objeto, esta clave será la *PRIMARY KEY* en la tabla de la base de datos.

El compilador se ha realizado utilizando las herramientas *lex* y *yacc*, y realizando la implementación en C++, con él se obtendrán las clases en Java necesarias para obtener un servidor como se verá a continuación. El código fuente del compilador se puede encontrar en el Apéndice, pero no será explicado al salir fuera del ámbito de este proyecto.

Durante la explicación se utilizará un pequeño ejemplo que se ha construido utilizando el compilador generado, dicho ejemplo será una biblioteca, donde se ha

tratado de abarcar el mayor número de posibilidades (varios tipos de datos, clientela, ...) en el menor número posible de interfaces.

5.2 DDL, un subconjunto de IDL

En la especificación del servicio de persistencia (especificación proporcionada por el OMG para independizar los objetos persistentes del sistema de almacenamiento) se comenta la posibilidad de utilizar un subconjunto de *IDL* para definir los datos que se almacenarán en la base de datos, a este subconjunto se le llama *DDL*.

Con *DDL* se pueden definir módulos e interfaces únicamente, y dentro de las interfaces sólo se permiten atributos. Los atributos pueden ser de cualquier tipo básico o del tipo de otra interfaz definida anteriormente.

El compilador realizado admite módulos dentro de módulos, modificadores de ámbito, definiciones anticipadas, los tipos básicos definidos por el OMG excepto el tipo *Object* al no poder representarlo en una base de datos. También se admite herencia entre interfaces.

Como consideración adicional al lenguaje *DDL*, se tendrá en cuenta que los atributos definidos como *readonly* se convertirán en la clave primaria de los objetos de dicha interfaz, por lo que es necesario definir al menos un atributo con dicho modificador.

5.3 Los ficheros generados

Al ejecutarse el compilador se le pedirá al usuario el nombre del fichero que contiene las interfaces en *DDL*, posteriormente se le pedirá el nombre de dos ficheros de salida, el primero correspondiente a las sentencias *SQL* necesarias para generar las tablas en la base de datos, y el segundo el nombre de un fichero *IDL* con las interfaces generadas por el compilador.

Para cada interfaz que el compilador detecte en el fichero de entrada se generarán las siguientes clases en Java, cada una en un fichero del mismo nombre que la clase generada:

- *interfazImpl.java*: La implementación para la interfaz CORBA que se define en el fichero *DDL*.

- *interfazFactoryImpl.java*: La implementación de la factoría de objetos para dicha interfaz. La factoría es la encargada de generar los objetos desde la base de datos y ofrecerlos al ORB y, por tanto, a los clientes.

- *interfazOA.java*: La clase encargada de transformar los datos sin formato de la interfaz en objetos concretos CORBA. Es utilizada por la factoría para implementar el acceso a bajo nivel a la base de datos.

- *interfazIteratorImpl.java*: La implementación de los iteradores para cada tabla. Se pueden seleccionar un conjunto de objetos que cumplan una condición y se obtienen todos ellos en orden.

- *Safeinterfaz.java*: Una clase para que los clientes accedan a los objetos remotos que los protege de los fallos de caché. Esta clase requiere la lectura del objeto de la base de datos, caso de que éste no esté disponible para el ORB. Estas clases son necesarias ya que las referencias CORBA a los objetos de la base de datos no se pueden mantener fijas a lo largo de todo el ciclo de vida de la aplicación, debido sobre todo a la entrada y la salida de éstos en/de la base de datos hace que se tengan que destruir y crear nuevos objetos para el ORB, con lo que las referencias cambian.

Finalmente se genera la clase *Servidor.java*. Esta clase contiene el método main que se encarga de dejar las factorías generadas en el servicio de nombres, a la espera de peticiones remotas.

Existen otras clases de utilidad en el paquete *dbcollection* (AbstractFactory.java, ChangeSensible.java, dbObjectAdaptor.java, servidorBD.java) que se irán mostrando conforme vayan siendo de interés para el desarrollo de la explicación.

5.3.1 biblioteca.ddl

La explicación se realizará basándose en una pequeña aplicación cuyo fichero *DDL* es el que se muestra a continuación:

biblioteca.ddl

```
1. module javi {
2. module biblioteca {
3.     interface autor;

4.     interface libro {
5.            readonly attribute string nombre;
6.            attribute autor autorLibro;
7.            attribute long ano;
8.     };

9.     interface editorial {
10.           readonly attribute string nombre;
11.           attribute string direccion;
12.    };

13.    interface autor {
14.           readonly attribute string nombre;
15.           readonly attribute string apellidos;
16.           attribute boolean vivo;
17.           attribute short edad;
18.    };

19. };
20. };
```

En este fichero de definición de datos se pueden encontrar dos módulos, uno dentro de otro, y en el de mayor profundidad se definen tres interfaces con atributos de

tipo *string, long, short* y *boolean*, y un atributo en la interfaz *libro* (Línea 6) que hace referencia a la interfaz *autor* que ha sido definida de forma anticipada en la línea 3.

Todas las interfaces tienen atributos de tipo *readonly* (en caso contrario el compilador daría un error), estando formada dicha clave por un atributo en el caso de *libro* y *editorial*, y por dos atributos en el caso de *autor*.

Durante el resto del capítulo se hará hincapié en las interfaces *libro* y *autor* por ser las que más aportan a la explicación.

5.3.2 biblio_tablas.sql

El compilador genera en primer lugar un fichero que contiene las sentencias *SQL* necesarias para crear las tablas que almacenarán los datos de los objetos persistentes.

biblio_tablas.sql

```
DROP TABLE javi_biblioteca_libro;
DROP TABLE javi_biblioteca_editorial;
DROP TABLE javi_biblioteca_autor;


CREATE TABLE javi_biblioteca_libro (
       nombre text,
       autorLibro_nombre text CONSTRAINT fk_autorLibro_nombre
                          REFERENCES javi_biblioteca_autor (nombre),
       autorLibro_apellidos text CONSTRAINT fk_autorLibro_apellidos
                          REFERENCES javi_biblioteca_autor (apellidos),
       ano int4,
       CONSTRAINT pk_javi_biblioteca_libro PRIMARY KEY (nombre)
       );

CREATE TABLE javi_biblioteca_editorial (
       nombre text,
       direccion text,
       CONSTRAINT pk_javi_biblioteca_editorial PRIMARY KEY (nombre)
       );

CREATE TABLE javi_biblioteca_autor (
       nombre text,
       apellidos text,
       vivo bool,
       edad int2,
       CONSTRAINT pk_javi_biblioteca_autor
                                       PRIMARY KEY (nombre, apellidos)
       );
```

Con este fichero se crean tres tablas, una para cada interfaz, el nombre de dichas tablas estará formado por el de los módulos a los que pertenece la interfaz más el nombre de la interfaz todo ello separado por el carácter '_', de esta forma no habrán dos tablas con el mismo nombre. En ellas se definen los campos que corresponden a los atributos del fichero de entrada, y se genera una clave primaria para cada tabla con los atributos del tipo *readonly*.

Cuando un atributo es del tipo de otra interfaz, como es el caso del atributo *autorLibro* en la interfaz *libro* que es del tipo *autor*, se añaden en la tabla *libro* los campos de la clave primaria de la tabla *autor*, con el nombre del campo de la tabla *libro* más el campo correspondiente en la tabla *autor* separados por el carácter '_'.

El fichero obtenido puede ser utilizado para generar las clases en el sistema gestor de bases de datos deseado, ya que éste cumple con el estándar *SQL*, por ejemplo para hacerlo en *postgresql* se utilizaría el programa *psql* de la siguiente forma:

```
> psql nombreBaseDatos
> \i biblio_tablas.sql
> \q
```

5.3.3 Fichero con las interfaces IDL

El segundo fichero que genera el compilador contiene una serie de interfaces utilizando el lenguaje *IDL* que serán necesarias para conseguir la persistencia:

biblio_generado.idl

```
#include "biblioteca.ddl"
module javi {
module biblioteca {
interface libroIterator {
       javi::biblioteca::libro next();
};
interface libroFactory {
       javi::biblioteca::libro getObject(in string nombre);
       void remove(in string nombre);
       boolean exists(in string nombre);
       libroIterator list(in string condition);
};
};
};

module javi {
module biblioteca {
interface editorialIterator {
       javi::biblioteca::editorial next();
};
interface editorialFactory {
       javi::biblioteca::editorial getObject(in string nombre);
       void remove(in string nombre);
       boolean exists(in string nombre);
       editorialIterator list(in string condition);
};
};
};

module javi {
module biblioteca {
interface autorIterator {
       javi::biblioteca::autor next();
};
interface autorFactory {
       javi::biblioteca::autor getObject(in string nombre,
                                         in string apellidos);
       void remove(in string nombre, in string apellidos);
       boolean exists(in string nombre, in string apellidos);
       autorIterator list(in string condition);
};
};
};
```

Se generan para cada interfaz del fichero de entrada dos nuevas interfaces que se corresponden con *interfazIterator* e *interfazFactory*, las cuales se encontrarán en el mismo módulo que la interfaz original.

La interfaz *Factory* implementa una interfaz en la que las tres primeras operaciones requieren la clave de la interfaz para:

- *getObject*: retornar un objeto del tipo específico, si no existe en la base de datos, lo crea como vacío.

- *remove*: borrar un objeto, es decir eliminar una tupla de la base de datos.

- *exists*: preguntar por la existencia de un objeto, es decir por la existencia de tupla en la base de datos.

Nótese que *libroFactory* y *editorialFactory* implementan las operaciones con un parámetro (ya que la clave sólo tiene un atributo) y *autorFactory* las implementa con dos parámetros (al tener dos atributos en su clave).

El último método, *list* retorna un *Iterator* para el tipo específico, tomando como parámetro una condición. Para dar mayor versatilidad la condición será el contenido directo de una cláusula *WHERE* en una consulta *SQL*.

Por otro lado la interfaz *Iterator* generada posee una única operación (*next*) que retornará el siguiente elemento de los obtenidos de la llamada a *list* o un nulo en CORBA.

El fichero *idl* generado se puede compilar con el traductor de *idl* a *Java* tal como se explicó en el capítulo 2, pero en este caso nos encontramos con un problema, ya que el fichero generado tiene un *include* al fichero con las interfaces originales, por lo que es necesario utilizar el preprocesador (es decir no hay que usar la opción *–fno-cpp*). Si no se dispone de ningún preprocesador para Windows se pueden utilizar herramientas de libre distribución para realizar esta tarea. En primer lugar se utilizará el preprocesador que acompaña al compilador de C++ en Linux quitándole a la salida los comentarios añadidos por el procesador, ya que en otro caso no funciona el compilador de *idl* a *Java*:

```
> /lib/cpp biblio_generado.idl | grep –v \# > prebiblio_generado.idl
```

A continuación ya podemos pasarle el programa *idltojava* al fichero generado, si queremos hacer esto sin necesidad de reiniciar la máquina en Windows, podemos utilizar el emulador para Linux llamado *wine*, copiando en el mismo directorio el programa *idltojava.exe* y ejecutando desde una terminal gráfica:

```
> wine "idltojava.exe -fno-cpp prebiblio_generado.idl"
```

5.3.4 interfazImpl.java

Para las interfaces definidas en el fichero *DDL* hay que generar una clase que implemente el acceso a los atributos, esta clase es obtenida de la herramienta desarrollada, de la siguiente forma:

libroImpl.java

```
package javi.biblioteca;
public class libroImpl extends _libroImplBase
       implements dbcollection.ChangeSensible {
       private String nombre;
       private autor autorLibro;
       private int ano;
       private boolean changed;
       public boolean changed() { return changed; }
       public libroImpl( String nombre, autor autorLibro, int ano) {
              changed = false;
              this.nombre = nombre;
              this.autorLibro = autorLibro;
              this.ano = ano;
       }

       public String nombre() { return nombre; }

       public autor autorLibro() { return autorLibro; }
       public void autorLibro( autor valor) {
              changed = true;
              autorLibro = valor;
       }

       public int ano() { return ano; }
       public void ano( int valor) {
              changed = true;
              ano = valor;
       }

}
```

autorImpl.java

```
package javi.biblioteca;
public class autorImpl extends _autorImplBase
       implements dbcollection.ChangeSensible {
       private String nombre;
       private String apellidos;
       private boolean vivo;
       private short edad;
       private boolean changed;
       public boolean changed() { return changed; }
       public autorImpl( String nombre, String apellidos,
                                 boolean vivo, short edad) {
              changed = false;
              this.nombre = nombre;
              this.apellidos = apellidos;
              this.vivo = vivo;
              this.edad = edad;
       }

       public String nombre() { return nombre; }

       public String apellidos() { return apellidos; }

       public boolean vivo() { return vivo; }
       public void vivo( boolean valor) {
              changed = true;
```

```
                vivo = valor;
        }

        public short edad() { return edad; }
        public void edad( short valor) {
                changed = true;
                edad = valor;
        }

}
```

Nótese cómo cada clase posee los atributos internos (*private*) de la interfaz correspondiente, y cómo existen sólo métodos de modificación para los campos que no son *readonly*, además se genera un constructor para inicializar todos los campos.

Estas clases implementan también una interfaz del paquete *dbcollection* llamada *ChangeSensible* que se utiliza para saber cuándo un objeto que está en memoria ha cambiado su estado, y por tanto una posterior invalidación debería escribir el nuevo contenido a la base de datos. La interfaz *dbcollection.ChangeSensible* se lista a continuación:

ChangeSensible.java

```
package dbcollection;

/**
 * Define el interfaz para los objetos que son sensibles al cambio,
 * como lo son aquellos objetos CORBA que
 * representan objetos relacionales
 */
public interface ChangeSensible
{
    /**
     * Metodo que indica el cambio
     * @return true si el objeto ha cambiado en alguno de sus
     * datos internos
     */
    public boolean changed();
}
```

5.3.5 interfazFactoryImpl.java

La implementación de esta interfaz está encargada únicamente de hacer de *wrapper* entre el cliente de la factoría y una factoría abstracta (*AbstractFactory* del paquete *dbcollection*).

libroFactoryImpl.java

```
package javi.biblioteca;
import java.util.Vector;

public class libroFactoryImpl extends _libroFactoryImplBase {
        private org.omg.CORBA.ORB orb;
        private libroOA oa;
        private dbcollection.AbstractFactory af;

        public libroFactoryImpl(org.omg.CORBA.ORB orb) {
                super();
                this.orb = orb;
                af=new dbcollection.AbstractFactory(orb, oa=new libroOA());
        }
```

```
        public libro getObject( String nombre) {
               Vector key = new Vector();
               key.add( nombre);
               return (libro)(af.getObject(key));
        }

        public void remove( String nombre) {
               Vector key = new Vector();
               key.add( nombre);
               af.remove(key);
        }

        public boolean exists( String nombre) {
               Vector key = new Vector();
               key.add( nombre);
               return af.exists(key);
        }

        public libroIterator list(String cond) {
               return new libroIteratorImpl(af,oa,cond,this,orb);
        }

        public void finalize() {
               af.finalize();
        }
}
```

autorFactoryImpl.java

```
package javi.biblioteca;
import java.util.Vector;

public class autorFactoryImpl extends _autorFactoryImplBase {
       private org.omg.CORBA.ORB orb;
       private autorOA oa;
       private dbcollection.AbstractFactory af;

       public autorFactoryImpl(org.omg.CORBA.ORB orb) {
              super();
              this.orb = orb;
              af=new dbcollection.AbstractFactory(orb, oa=new autorOA());
       }

       public autor getObject( String nombre, String apellidos) {
              Vector key = new Vector();
              key.add( nombre);
              key.add( apellidos);
              return (autor)(af.getObject(key));
       }

       public void remove( String nombre, String apellidos) {
              Vector key = new Vector();
              key.add( nombre);
              key.add( apellidos);
              af.remove(key);
       }

       public boolean exists( String nombre, String apellidos) {
              Vector key = new Vector();
              key.add( nombre);
              key.add( apellidos);
              return af.exists(key);
       }

       public autorIterator list(String cond) {
              return new autorIteratorImpl(af,oa,cond,this,orb);
       }
```

```
        public void finalize() {
              af.finalize();
        }
}
```

En cada clase *Factory* se crea un método *finalice*, este método debería ser invocado por la máquina virtual Java al eliminar el objeto de memoria, sin embargo tras realizar algunas pruebas se ha comprobado que no es así, por lo que es necesario que sea invocado explícitamente. La función de este método es únicamente invocar el correspondiente método *finalice* en la clase *AbstractFactory* para que limpie la cache antes de la finalización.

Esta clase utiliza la clase *dbcollection.AbstractFactory* para implementar toda su funcionalidad. *AbstractFactory* es la encargada de mantener la caché y de conectar y desconectar objetos con el ORB. Es abstracta en el sentido de que no tiene en cuenta el tipo de los objetos que guarda en su caché (los trata a todos al nivel *java.lang.Object*, como los contenedores nativos Java). Sólo necesita un objeto (*java.util.Vector*) *key* que identifica a cada objeto de forma unívoca. Se utiliza un objeto *Vector* como clave porque una clave puede estar formada por varios atributos, debe tenerse en cuenta que para poder almacenar un valor en el vector es necesario convertirlo a un objeto (*java.lang.Integer, ...*).

La clase *AbstractFactory* puede manejar los objetos de su colección simplemente teniendo en cuenta que obtiene una clave para identificarlos, y que también le servirá para buscarlos en la base de datos. Así esta clase puede mantener la caché y conectar y desconectar los objetos del ORB.

AbstractFactory.java

```
package dbcollection;

import java.util.Vector;

/**
 * Esta clase implementa una factoria abstracta. Los objetos de una
 * tabla relacional son muy facilmente encapsulables en un tipo
 * <code>Vector</code>. El adaptador
 * de base de datos adecuado <code>dbObjectAdaptor</code> se utilizara
 * para acceder a la base de datos con el tipo adecuado
 * A su vez, las factorias especificas
 * seran clientes de esta y solo haran de interfaz con CORBA para
 * conservar los tipos.
 */


public class AbstractFactory
{
    // Variables internas
    private dbObjectAdaptor dbOA;   // Interfaz objeto-relacinal
    private org.omg.CORBA.ORB orb;  // El ORB


    static final private int CACHE_MAX = 2048;
    private java.util.Hashtable cache =
                                new java.util.Hashtable(CACHE_MAX);

    // Constructor. Se encarga de conectarse a la base de datos
    public AbstractFactory(org.omg.CORBA.ORB orb, dbObjectAdaptor dbOA)
    {
        this.dbOA = dbOA;
```

```
        this.orb = orb;
    }

    public java.lang.Object getObject(Vector key)
    {
        if (cache.containsKey(key)) {
            return cache.get(key);
        } else {
            java.lang.Object tmpObj = dbOA.retrieve(key);
            cacheIt(key,tmpObj);
            return tmpObj;
        }
    }

    private void cacheIt(Vector key, java.lang.Object obj)
    {
        if (cache.size() > CACHE_MAX)
        {
            java.lang.Object tmpKey;
            java.util.Enumeration e = cache.keys();

            tmpKey = e.nextElement();
            Vector key2 = new Vector();
            key2.add(tmpKey);
            this.store(key2);
        }

        // Conecta al ORB
        orb.connect((org.omg.CORBA.Object)obj);
        cache.put(key,obj);
    }

    // Este es un interfaz de bajo nivel para acceder directamente a la
    // cache. Los iteradores extraen objetos de la base de datos y los
    // intentan introducir en la cache.
    public java.lang.Object cacheObject(Vector key,java.lang.Object obj)
    {
        if (cache.containsKey(key)) {
            return cache.get(key);
        } else {
            cacheIt(key,obj);
            return obj;
        }
    }

    public void remove(Vector key)
    {
        this.store(key);
        dbOA.remove(key);
    }

    private void store(Vector key)
    {
        java.lang.Object tmpObj;

        if (cache.containsKey(key))
        {
            tmpObj = cache.get(key);
            if (((ChangeSensible)tmpObj).changed()){
                dbOA.store(tmpObj);
            }
            // Desconecta del ORB.
            orb.disconnect((org.omg.CORBA.Object)tmpObj);

            // Borra de la cache
            cache.remove(key);
        }
    }
```

```
    public boolean exists(Vector key)
    {
        return cache.containsKey(key) || dbOA.exists(key);
    }

    public void finalize()
    {
        java.util.Vector tmpKey;
        java.util.Enumeration e = cache.keys();

        while (e.hasMoreElements())
        {
            tmpKey = (java.util.Vector)e.nextElement();
            this.store(tmpKey);
        }
    }
}
```

El método *cacheObject* será de utilidad para los iteradores, que se verán más adelante.

Esta clase utiliza a su vez otra que implementa la interfaz *dbcollection.dbObjectAdaptor* para lograr acceder a la base de datos a un bajo nivel, de tal forma que se acceda a la tabla adecuada y se generen objetos CORBA del tipo adecuado que puedan ser conectados al ORB.

5.3.6 interfazOA.java

Esta es la clase que implementa la interfaz Java *dbObjectAdaptor*. Esta clase es la encargada de desempaquetar la clave del *Vector* que recibe como parámetro, acceder a la base de datos con comandos *SQL* y obtener el objeto del tipo CORBA específico a partir de los datos de las consultas. Después se lo devolverá al *AbstractFactory* encapsulado otra vez como un *java.lang.Object*.

La interfaz *dbcollection.dbObjectAdaptor* es la siguiente:

dbObjectAdaptor.java

```
package dbcollection;

/**
 * El interfaz dbObjectAdaptor identifica a los objetos que pueden hacer
 * de puente entre un objeto CORBA y una base de datos. Para cada clase
 * CORBA destinada a contener un objeto relacional, se crea una clase
 * que implementa este interfaz. Estas clases son necesarias debido
 * a la falta de genericidad en Java. Esto viene obligado debido a que
 * si queremos tener una clase como "AbstractFactory" que pueda tratar
 * con cualquier tipo de objetos, por un lado, tenemos que darle objetos
 * de tipo "Object", pero, por otro, deben ser realmente objetos del
 * tipo CORBA adecuado. Finalmente, se producira una conversion para
 * enviarla al cliente en la clase "ClientFactory".
 */
public interface dbObjectAdaptor extends java.util.Enumeration
{
    /**
     * Retorna un objeto (CORBA) del tipo especifico representado por
     * el <code>java.lang.Object</code> retornado. La implementacion
     * especifica de estos metodos dependen de la base de datos que se
```

```
         * este utilizando (SQL, ficheros, etc.) y del tipo de objetos a
         * retornar.
         * @param key Clave primaria del objeto relacional en un vector
         * @return El correspondiente objeto CORBA
         */
        java.lang.Object retrieve(java.util.Vector key);

        /**
         * Almacena un objeto del tipo especifico en la base de datos.
         * Notese que estas llamadas solo se realizaran para objetos en los
         * que se necesite su guardado, porque hayan sido modificados. Los
         * objetos CORBA correspondientes a las bases de datos implementan
         * el interfaz <code>ChangeSensible</code>, el cual puede ser
         * utilizado para preguntarles si se han modificado, y en ese caso,
         * ser guardados por la <code>AbstractFactory</code> cuando sea
         * necesario.
         * @param obj El objeto CORBA a ser almacenado
         */
        void store(java.lang.Object obj);

        /**
         * Elimina un objeto de la base de datos.
         * @param key La clave del objeto relacional a ser eliminado
         */
        void remove(java.util.Vector key);

        /**
         * Comprueba la existencia en la base de datos de un objeto.
         * @param key La clave del objeto relacional a ser comprobado
         * @return true si el objeto existe en la base de datos
         */
        boolean exists(java.util.Vector key);

    }
```

Esta interfaz extiende la interfaz *java.util.Enumeration*, lo cual será de utilidad para los iteradores que se mostrarán en el siguiente capítulo. En resumen, las factorías específicas utilizan a la *AbstractFactory* para obtener facilidades de caching y registro de objetos CORBA, y esta última utiliza a su vez a las clases *dbObjectAdaptor* para conseguir acceso a las bases de datos e independizarse del tipo de los objetos que maneja en su caché.

A continuación se encuentran los ficheros generados para las interfaces *libro* y *autor*.

libroOA.java

```
package javi.biblioteca;
import java.util.Vector;
import java.util.Enumeration;
import java.sql.*;

public class libroOA implements dbcollection.dbObjectAdaptor {
       private Statement localSt;
       private Statement pSt;
       private ResultSet pRs;
       private boolean hasMore = false;

       public libroOA() {
              try {
                     localSt =
                     dbcollection.servidorBD.baseDatos.createStatement();
                     pSt =
                     dbcollection.servidorBD.baseDatos.createStatement();
              } catch (SQLException ex) {
```

```
                throw new
                    RuntimeException("Fallo en la base de datos");
        }
    }

    public Object retrieve(Vector key) {
        try {
            ResultSet rs = localSt.executeQuery("SELECT * FROM
                javi_biblioteca_libro WHERE nombre = '" +
                key.get(0).toString() + "'");
            if (!rs.next()) {
                if (!((String)key.get(0)).equals("")) {
                    localSt.executeUpdate("INSERT INTO
                            javi_biblioteca_libro VALUES ('"
                            + key.get(0).toString() + "' ,
                            '' , '' , '0')");
                    return new
                libroImpl((String)(key.get(0)) , null , 0);
                } else
                    return null;
            } else {
                Vector key2 = new Vector();
                key2.add(( rs.getString(2)));
                key2.add(( rs.getString(3)));
                autor p2 = (autor)(new
                    autorOA()).retrieve(key2);
                return new libroImpl(rs.getString(1), p2,
                                rs.getInt(4));
            }
        } catch (SQLException ex) {
            return null;
        }
    }

    public void store(Object o) {
        libro obj = (libro) o;
        try {
            localSt.executeUpdate("UPDATE javi_biblioteca_libro
                SET autorLibro_nombre = '" +
                ((obj.autorLibro()==null) ? "" :
                obj.autorLibro().nombre()) + "',
                autorLibro_apellidos = '" +
                ((obj.autorLibro()==null) ? "" :
                obj.autorLibro().apellidos()) + "', ano = '"
                + obj.ano() + "' WHERE nombre = '" +
                obj.nombre() + "'");
        } catch (SQLException ex) {
            return;
        }
    }

    public void remove(Vector key) {
        try {
            localSt.executeUpdate("DELETE FROM
                javi_biblioteca_libro WHERE nombre = '" +
                key.get(0).toString() + "'");
        } catch (SQLException ex) {
            return;
        }
    }

    public boolean exists(Vector key) {
        try {
            ResultSet rs = localSt.executeQuery("SELECT * FROM
                javi_biblioteca_libro WHERE nombre = '" +
                key.get(0).toString() + "'");
            return rs.next();
        } catch (SQLException ex) {
```

```
                        return false;
                }
        }

        public boolean hasMoreElements() {
                return hasMore;
        }

        public Enumeration list(String cond) {
                try {
                        String queryStr = "SELECT * FROM
                                javi_biblioteca_libro";
                        if (!cond.equals("")) {
                                queryStr = queryStr + " WHERE " + cond;
                        }
                        pRs = pSt.executeQuery(queryStr);
                        hasMore = pRs.next();
                        return (Enumeration) this;
                } catch (SQLException ex) {
                        throw new
                           RuntimeException("Error en la Base de Datos");
                }
        }

        public Object nextElement() {
                if (!hasMore) { throw new
                                java.util.NoSuchElementException(); }
                try {
                        Vector key2 = new Vector();
                        key2.add(( pRs.getString(2)));
                        key2.add(( pRs.getString(3)));
                        autor p2 = (autor)(new autorOA()).retrieve(key2);
                        Object tmpObj = new libroImpl(pRs.getString(1), p2,
                                pRs.getInt(4));
                        hasMore = pRs.next();
                        return tmpObj;
                } catch (SQLException ex) {
                        throw new
                           RuntimeException("Error en la Base de Datos");
                }
        }

}
```

autorOA.java

```
package javi.biblioteca;
import java.util.Vector;
import java.util.Enumeration;
import java.sql.*;

public class autorOA implements dbcollection.dbObjectAdaptor {
        private Statement localSt;
        private Statement pSt;
        private ResultSet pRs;
        private boolean hasMore = false;

        public autorOA() {
                try {
                        localSt =
                        dbcollection.servidorBD.baseDatos.createStatement();
                        pSt =
                        dbcollection.servidorBD.baseDatos.createStatement();
                } catch (SQLException ex) {
                        throw new
                           RuntimeException("Fallo en la base de datos");
                }
        }
```

```
public Object retrieve(Vector key) {
	try {
		ResultSet rs = localSt.executeQuery("SELECT * FROM
			javi_biblioteca_autor WHERE nombre = '" +
			key.get(0).toString() + "' AND apellidos = '"
			+ key.get(1).toString() + "'");
		if (!rs.next()) {
			if (!((String)key.get(0)).equals("")) {
				localSt.executeUpdate("INSERT INTO
					javi_biblioteca_autor VALUES ('"
					+ key.get(0).toString() + "' ,
					'" + key.get(1).toString() + "'
					, 'false' , '0')");
				return new
					autorImpl((String)(key.get(0)) ,
					(String)(key.get(1)) , false ,
					(short)0);
			} else
				return null;
		} else {
			return new autorImpl(rs.getString(1),
				rs.getString(2), rs.getBoolean(3),
				rs.getShort(4));
		}
	} catch (SQLException ex) {
		return null;
	}
}

public void store(Object o) {
	autor obj = (autor) o;
	try {
		localSt.executeUpdate("UPDATE javi_biblioteca_autor
			SET vivo = '" + obj.vivo() + "', edad = '" +
			obj.edad() + "' WHERE nombre = '" +
			obj.nombre() + "' AND apellidos = '" +
			obj.apellidos() + "'");
	} catch (SQLException ex) {
		return;
	}
}

public void remove(Vector key) {
	try {
		localSt.executeUpdate("DELETE FROM
			javi_biblioteca_autor WHERE nombre = '" +
			key.get(0).toString() + "' AND apellidos = '"
			+ key.get(1).toString() + "'");
	} catch (SQLException ex) {
		return;
	}
}

public boolean exists(Vector key) {
	try {
		ResultSet rs = localSt.executeQuery("SELECT * FROM
			javi_biblioteca_autor WHERE nombre = '" +
			key.get(0).toString() + "' AND apellidos = '"
			+ key.get(1).toString() + "'");
		return rs.next();
	} catch (SQLException ex) {
		return false;
	}
}

public boolean hasMoreElements() {
	return hasMore;
```

```
        }

        public Enumeration list(String cond) {
            try {
                String queryStr = "SELECT * FROM
                    javi_biblioteca_autor";
                if (!cond.equals("")) {
                    queryStr = queryStr + " WHERE " + cond;
                }
                pRs = pSt.executeQuery(queryStr);
                hasMore = pRs.next();
                return (Enumeration) this;
            } catch (SQLException ex) {
                throw new
                   RuntimeException("Error en la Base de Datos");
            }
        }

        public Object nextElement() {
            if (!hasMore) { throw new
                java.util.NoSuchElementException(); }
            try {
                Object tmpObj = new autorImpl(pRs.getString(1),
                    pRs.getString(2), pRs.getBoolean(3),
                    pRs.getShort(4));
                hasMore = pRs.next();
                return tmpObj;
            } catch (SQLException ex) {
                throw new
                   RuntimeException("Error en la Base de Datos");
            }
        }

    }
```

Lo que muestran estas clases es cómo obtener desde la base de datos los datos requeridos por la clase *AbstractFactory*.

Estas clases implementan los métodos necesarios para almacenar y recuperar objetos a partir de tablas relacionales. Cabe destacar que se definen dos *Statement*, uno para las consultas que la clase necesite y otro que se utilizará para implementar un servicio de iterador. Como la clase implementa la interfaz *Enumeration* es necesario definir el método *nextElement* que es el único definido en dicha interfaz.

Puede verse en la clase *libroOA* como se trata de forma especial el campo del tipo *autor*, ya que debe obtenerse el objeto correspondiente cuando se accede a un libro, y debe almacenarse su valor en la tabla *autor*. Se ha desarrollado el compilador para permitir varios campos con referencias a otras interfaces, sin que ello de fallo en el compilador de Java.

5.3.7 interfazIteratorImpl.java

Esta clase da servicio de iteración a los clientes CORBA de la base de datos. La interfaz es parecida a la que se encuentra en el servicio de nombres con su *BindingIterator*. La factoría tiene un método *list* que retorna un iterador que se puede ir recorriendo con su método *next*.

En la implementación se utiliza directamente el *dbObjectAdaptor* de la factoría que la crea y se generan objetos que son introducidos en la caché de la factoría a través del método *cacheObject*. Este método, por su parte, retorna el objeto que ya estuviera en caché (si lo estaba) y si no, inserta en la misma el objeto recién creado desde la base de datos.

libroIteratorImpl.java

```
package javi.biblioteca;
import dbcollection.AbstractFactory;
import java.util.Vector;
import java.util.Enumeration;

public class libroIteratorImpl extends _libroIteratorImplBase {
	private String cond;
	private AbstractFactory af;
	private Enumeration e;
	private libroFactory fac;
	private org.omg.CORBA.ORB orb;

	public libroIteratorImpl(AbstractFactory af, libroOA oa,
		String cond, libroFactory fac, org.omg.CORBA.ORB orb) {
		e = oa.list(cond);
		this.af = af;
		this.cond = cond;
		this.fac = fac;
		this.orb = orb;
	}

	public libro next() {
		if (!e.hasMoreElements())
			{ orb.disconnect(this); return null; }
		Object tmpObj = e.nextElement();
		 String nombre = ((libro)tmpObj).nombre();
		Vector key = new Vector();
		key.add( nombre);
		return (libro)(af.cacheObject(key,tmpObj));
	}
}
```

autorIteratorImpl.java

```
package javi.biblioteca;
import dbcollection.AbstractFactory;
import java.util.Vector;
import java.util.Enumeration;

public class autorIteratorImpl extends _autorIteratorImplBase {
	private String cond;
	private AbstractFactory af;
	private Enumeration e;
	private autorFactory fac;
	private org.omg.CORBA.ORB orb;

	public autorIteratorImpl(AbstractFactory af, autorOA oa,
		String cond, autorFactory fac, org.omg.CORBA.ORB orb) {
		e = oa.list(cond);
		this.af = af;
		this.cond = cond;
		this.fac = fac;
		this.orb = orb;
	}

	public autor next() {
		if (!e.hasMoreElements())
			{ orb.disconnect(this); return null; }
		Object tmpObj = e.nextElement();
```

```
                String nombre = ((autor)tmpObj).nombre();
                String apellidos = ((autor)tmpObj).apellidos();
                Vector key = new Vector();
                key.add( nombre);
                key.add( apellidos);
                return (autor)(af.cacheObject(key,tmpObj));
        }
}
```

Estas clases utilizan el método *list* de *dbObjectAdaptor* para generar un *java.util.Enumeration* (que resulta ser el mismo *dbObjectAdaptor* pero visto como otra de las interfaces que implementa). Después, implementan su interfaz en función de este último.

Es importante destacar que en el método *next*, cuando ya no quedan más objetos que devolver se devuelve *null* (que corresponde con *nil* en CORBA), pero antes de devolver dicho objeto se debe desconectar el iterador del ORB, porque en otro caso quedará en memoria permanentemente aunque a partir del momento en que devuelve nulo ya no se debería volver a utilizar. Para comprobar esto se ha utilizado la opción *–Xrunhprof* de la máquina virtual Java proporcionada por Sun (*java*) que provoca un volcado de los objetos en memoria al finalizar el programa (entre otras cosas). Si no se desconecta del ORB se van creando objetos *Iterator* que se quedan en memoria, sin embargo, si se desconecta estos objetos son recogidos por el *garbage collection.*

5.3.8 Safeinterface.java

Debido a que los objetos están de forma dinámica entrando y saliendo del ORB, no se puede asegurar que una referencia a un objeto no se haga inválida al cabo de algún tiempo. Para ello, se han creado unas clases que actúan cono *escudo* frente a este cambio de la referencia. Básicamente, guardan en su interior la referencia más reciente, junto con los valores de la clave para ese objeto, necesarios para recuperar desde la factoría una referencia válida para el objeto. El escudo lo realizan gracias a la captura de la excepción *CORBA::OBJECT_NOT_EXIST*. He aquí las clases generadas de forma automática:

Safelibro.java

```
package javi.biblioteca;
public class Safelibro {
       public libro ref;
       private libroFactory factory;
       private String nombre;

       public Safelibro(libroFactory factory, String nombre,
                     autor autorLibro, int ano) {
              this.factory = factory;
              this.nombre = nombre;
              ref = factory.getObject(nombre);
              autorLibro(autorLibro);
              ano(ano);
       }

       public Safelibro(libroFactory factory, String nombre) {
              this.factory = factory;
              this.nombre = nombre;
              ref = factory.getObject(nombre);
       }
```

```
	public Safelibro(libroFactory factory, libro ref) {
		this.factory = factory;
		this.ref = ref;
		this.nombre = nombre();
	}

	public String nombre() {
		while (true) {
			try {
				return ref.nombre();
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre);
			}
		}
	}

	public autor autorLibro() {
		while (true) {
			try {
				return ref.autorLibro();
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre);
			}
		}
	}

	public void autorLibro( autor value) {
		while (true) {
			try {
				ref.autorLibro(value);
				return;
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre);
				autorLibro(value);
			}
		}
	}

	public int ano() {
		while (true) {
			try {
				return ref.ano();
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre);
			}
		}
	}

	public void ano( int value) {
		while (true) {
			try {
				ref.ano(value);
				return;
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre);
				ano(value);
			}
		}
	}

}
```

Safeautor.java

```
package javi.biblioteca;
public class Safeautor {
	public autor ref;
```

```
private autorFactory factory;
private String nombre;
private String apellidos;

public Safeautor(autorFactory factory, String nombre,
              String apellidos, boolean vivo, short edad) {
       this.factory = factory;
       this.nombre = nombre;
       this.apellidos = apellidos;
       ref = factory.getObject(nombre, apellidos);
       vivo(vivo);
       edad(edad);
}

public Safeautor(autorFactory factory, String nombre,
              String apellidos) {
       this.factory = factory;
       this.nombre = nombre;
       this.apellidos = apellidos;
       ref = factory.getObject(nombre, apellidos);
}

public Safeautor(autorFactory factory, autor ref) {
       this.factory = factory;
       this.ref = ref;
       this.nombre = nombre();
       this.apellidos = apellidos();
}

public String nombre() {
       while (true) {
              try {
                     return ref.nombre();
              } catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
                     ref = factory.getObject(nombre, apellidos);
              }
       }
}

public String apellidos() {
       while (true) {
              try {
                     return ref.apellidos();
              } catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
                     ref = factory.getObject(nombre, apellidos);
              }
       }
}

public boolean vivo() {
       while (true) {
              try {
                     return ref.vivo();
              } catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
                     ref = factory.getObject(nombre, apellidos);
              }
       }
}

public void vivo( boolean value) {
       while (true) {
              try {
                     ref.vivo(value);
                     return;
              } catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
                     ref = factory.getObject(nombre, apellidos);
                     vivo(value);
              }
```

```
		}
	}

	public short edad() {
		while (true) {
			try {
				return ref.edad();
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre, apellidos);
			}
		}
	}

	public void edad( short value) {
		while (true) {
			try {
				ref.edad(value);
				return;
			} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {
				ref = factory.getObject(nombre, apellidos);
				edad(value);
			}
		}
	}

}
```

En estas clases hay tres constructores:

1. Recibe la factoría y cada uno de los atributos del objeto, con los atributos que forman la clave obtiene el objeto de la base de datos o lo crea, y entonces modifica el resto de atributos.

2. Recibe la factoría y sólo los atributos que forman la clave, si la interfaz no tiene atributos además de los de la clave, no se genera este constructor. Hace lo mismo que el anterior pero sin modificar nada.

3. Recibe la factoría y una referencia a un objeto ya obtenido, es útil cuando se está utilizando el iterador.

5.3.9 Servidor.java

Esta clase se encarga de crear una factoría de cada tipo, y de asociarlas en el servicio de nombres con un contexto de tal forma que el árbol de contextos coincida con el árbol de módulos en las interfaces.

El código generado para el ejemplo de la biblioteca es el siguiente:

Servidor.java

```
package javi.biblioteca;
import org.omg.CORBA.*;
import java.io.*;
import org.omg.CosNaming.*;

public class Servidor {
	public static void main(String args[]) {
		ORB orb = ORB.init(args,null);
		NamingContext rootContext=null;
```

```
try {
      org.omg.CORBA.Object objRef =
         orb.resolve_initial_references("NameService");
      rootContext = NamingContextHelper.narrow(objRef);
} catch (org.omg.CORBA.ORBPackage.InvalidName ex) {
      System.err.println("Error, no se encuentra el
                                 servicio de nombres");
      System.exit(-1);
}
try {
      NameComponent modContext[] = new NameComponent[1];
      NamingContext servidores = rootContext;
      modContext[0] = new NameComponent("javi", "");
      try {
            servidores =
                  NamingContextHelper.narrow(
                        servidores.resolve(modContext));
      } catch
            (org.omg.CosNaming.NamingContextPackage.
                  NotFound ex) {
            servidores =
               servidores.bind_new_context(modContext);
      }
      modContext[0] = new NameComponent("biblioteca", "");
      try {
            servidores =
                  NamingContextHelper.narrow(servidores.
                        resolve(modContext));
      } catch
            (org.omg.CosNaming.NamingContextPackage.
                  NotFound ex) {
            servidores =
               servidores.bind_new_context(modContext);
      }
      libroFactoryImpl factlibro = new
                  libroFactoryImpl(orb);
      orb.connect(factlibro);
      modContext[0] = new
            NameComponent("libroFactoryImpl", "");
      servidores.rebind(modContext, factlibro);
      editorialFactoryImpl facteditorial = new
            editorialFactoryImpl(orb);
      orb.connect(facteditorial);
      modContext[0] = new
            NameComponent("editorialFactoryImpl", "");
      servidores.rebind(modContext, facteditorial);
      autorFactoryImpl factautor = new
            autorFactoryImpl(orb);
      orb.connect(factautor);
      modContext[0] = new
            NameComponent("autorFactoryImpl", "");
      servidores.rebind(modContext, factautor);
      System.out.println("Servidor  a la escucha");
      System.out.println("Pulse ENTER para finalizar");
      try {
            BufferedReader teclado;
            teclado = new BufferedReader(new
                  InputStreamReader(System.in));
            String linea=teclado.readLine();
      } catch (IOException ex) {}
      factlibro.finalize();
      facteditorial.finalize();
      factautor.finalize();
} catch (org.omg.CORBA.UserException ex) {
      System.err.println("Excepcion no esperada,
                                 finalizacion anormal");
      ex.printStackTrace();
}
```

```
        }
}
```

Es importante invocar el método *finalice* para cada factoría antes de la finalización del programa, pues en otro caso los objeto que se encuentren en caché no serán actualizados en la base de datos.

En caso de querer generar otro servidor es aconsejable hacerlo partiendo de éste para evitar algún error como el comentado en el párrafo anterior.

5.3.10 Cliente.java

Para probar el funcionamiento de las clases generadas por el compilador se ha desarrollado un pequeño cliente en modo texto, al que se le pueden introducir comandos para obtener un objeto (libro, editorial o autor), modificar sus datos, o listar todos los objetos de un tipo, haciendo uso del iterador.

Por el tamaño de esta clase el código fuente no se ha incluido a continuación y puede verse en el Apéndice con el resto de clases que no han sido añadidas.

Aunque en el ejemplo que se ha presentado no se dé ningún caso de herencia, ésta está soportada por el compilador, habiendo sido probada con otros ejemplos.

Finalmente destacar que para realizar la aplicación únicamente ha sido necesario desarrollar el fichero con la especificación en *DDL* y el cliente, el resto de clases son generadas automáticamente por el compilador.

6. Conclusiones

El OMG definió en 1990 el estándar CORBA, desde entonces se han realizado una gran cantidad de implementaciones de dicho estándar por diferentes compañías de software como por ejemplo *Borland* (*Visibroker*).

El OMG se ha encargado de ampliar y perfeccionar el estándar en sus diferentes versiones, la versión 2.x ha sido la más utilizada, de hecho es la versión implementada por las *JDK1.2* y por el *ORBacus*. Próximamente el OMG lanzará la versión 3 del estándar CORBA, la cual puede obtenerse de las páginas Web del grupo (www.omg.org).

Un gran número de compañías utilizan actualmente alguna de las múltiples implementaciones de CORBA para desarrollar software, algunos ejemplos son:

- **Gnome**: El entorno de ventanas desarrollado para Linux, que ha sufrido una gran aceptación, siendo incluido en la mayoría de las distribuciones actuales (Debian, RedHat, SuSe, ...).

- **Oracle Web Server**: Un servidor Web desarrollado por Oracle que facilita el acceso a la base de datos de dicha compañía, además de proporcionar un gran rendimiento.

En los capítulos anteriores se explica como funcionaría un programa que utilice una implementación de CORBA, y cómo ésta ayuda al desarrollo del programa (facilitando la división del problema en varios niveles), y sobre todo a la distribución de la carga de éste entre varios nodos. Se ha utilizado la implementación desarrollada por *Sun* (*JDK1.2*) y la desarrollada por *Object-Oriented Concepts Inc.* (*ORBacus*) para la distribución de los servidores por una red de computadoras, y para el acceso por parte de los clientes a dichos servidores.

En el desarrollo de la aplicación para un centro comercial virtual, además de CORBA, se han mezclado varias tecnologías con gran éxito:

- **Bases de datos**: Se ha utilizado el gestor de bases de datos para Linux *postgresql*.

- **Acceso a bases de datos desde Java**: Para acceder a la base de datos desde un programa Java se han utilizado las librerías incluidas con *Postgresql* que se basan en el estándar *JDBC* definido por Sun para el acceso a bases de datos.

- **Acceso a bases de datos desde C++**: En el caso de C++ se han utilizado las librerías *pq++* también incluidas con *Postgresql*.

- **Servidor Web**: Como servidor Web se ha utilizado *Apache*.

- **Java**: Se utiliza Java para el desarrollo de los clientes y los servidores, lo cual provee a la aplicación de una gran portabilidad al ser utilizable en cualquier plataforma que tenga una implementación de las JDK 1.2.

- **C++**: También se ha desarrollado la aplicación en C++, contrastando de esta forma este lenguaje con su nuevo competidor, Java.

- **CGI**: Esta es la parte del desarrollo más oscura, ya que la utilización de un programa en Perl para mantener el estado entre diferentes páginas Web en el navegador del cliente es una solución compleja e ineficiente. Sería necesario que se desarrollara una interfaz más depurada entre un applet y el navegador, de tal forma que se pudieran mantener variables entre diferentes páginas Web.

CORBA facilita en gran medida el desarrollo de aplicaciones distribuidas al proveer una capa intermedia que se encarga del manejo de los objetos distribuidos, permite el acceso a dichos objetos de forma transparente a su localización, y provee mecanismos para descubrir qué objetos servidores existen gracias al servicio de nombres. Además contempla otros servicios adicionales que no son implementados por todos los fabricantes como el servicio de eventos.

Como inconvenientes a CORBA cabe destacar que se encuentra aún en desarrollo con lo que se pueden encontrar implementaciones del estándar incompletas, soluciones propietarias a problemas del estándar y rápidas modificaciones realizadas por el OMG sobre dicho estándar para tratar de solucionar dichos problemas.

En resumen CORBA es una tecnología con gran futuro, que facilita la programación distribuida, y por tanto la reutilización de componentes, en este caso incluso la reutilización de objetos, ya que estos pueden ser descubiertos mediante el servicio de nombres y utilizados mediante invocaciones dinámicas por clientes que en principio no fueron pensados para utilizarlos.

En cuanto al lenguaje de programación utilizado (C++ o Java) en mi opinión personal creo que Java permite un desarrollo más rápido y correcto, ya que no se producen tantos errores al no tener que manejar las referencias, por esta misma razón es más sencillo detectar los errores producidos al programar. Además un programa realizado con Java es multiplataforma, y permite una liberación de la instalación al desarrollar el cliente con *Applets*. Por otro lado la ejecución final de la aplicación es más rápida si el programa fue desarrollado en C++.

En mi opinión personal CORBA se establecerá como un estándar en la programación distribuida superando a cualquiera de sus competidores (DCOM, RMI, ...) por sus características (independencia del lenguaje de programación, independencia de la plataforma, facilidad de uso, ...) y por la gran cantidad de servicios que proporciona (muchos de ellos aún en desarrollo). Además un punto a su favor ha sido la gran aceptación por parte de las compañías desarrolladoras de entornos de programación como Borland o Sun.

Apéndice A. Código fuente del TicTacToe

A.1 Primera aproximación

A continuación se encuentra el código fuente para el servidor y el cliente de *TicTacToe*, en la primera de las aproximaciones realizadas, es decir la que corresponde con los capítulos 3.3 a 3.6

TicTacToe.idl

```
module TicTacToe {
	enum ficha {circulo,cruz,vacia};
	typedef ficha tipoTablero[3][3];
	enum estadoJuego {victoriaCirculo, victoriaCruz, tablas, jugando,
inicializando};

	exception CasillaOcupada {};
	exception YaHayDosJugadores {};
	exception JuegoParado {};

	interface Juego {

		readonly attribute tipoTablero tablero;
		readonly attribute ficha turno;
		readonly attribute estadoJuego estado;

		tipoTablero anadirJugador(out ficha usar)
			raises(YaHayDosJugadores);

		tipoTablero ponerFicha(in short x, in short y)
			raises(JuegoParado, CasillaOcupada);
	};
};
```

JuegoImpl.java

```
package TicTacToe;

public class JuegoImpl extends _JuegoImplBase {
	protected TicTacToe.ficha[][] tablero;
	protected TicTacToe.ficha turno;
	protected TicTacToe.estadoJuego estado;
	protected int numeroJugadores = 0;


	public JuegoImpl() {
		super();

		inicializaTablero();
		estado=estadoJuego.inicializando;
		turno=ficha.circulo;
	}

	public TicTacToe.ficha[][] tablero() {
		return tablero;
	}

	public TicTacToe.ficha turno() {
		return turno;
	}
```

```
	public TicTacToe.estadoJuego estado() {
		return estado;
	}

	public   TicTacToe.ficha[][]   anadirJugador(TicTacToe.fichaHolder
usar)
		throws TicTacToe.YaHayDosJugadores {
		if (numeroJugadores==2) {
			System.out.println("ERROR :Jugador no Añadido, juego
completo");
			throw new TicTacToe.YaHayDosJugadores();
		}

		if (numeroJugadores==0) {
			usar.value=ficha.circulo;
			System.out.println("Jugador 1 Añadido");
		}
		else {
			System.out.println("Jugador 2 Añadido");
			usar.value=ficha.cruz;
		}

		numeroJugadores++;

		if (numeroJugadores==2) {
			estado = estadoJuego.jugando;
			turno = ficha.circulo;
		}

		return tablero;
	}

	public TicTacToe.ficha[][] ponerFicha(short x, short y)
		throws TicTacToe.JuegoParado, TicTacToe.CasillaOcupada {

		if (x<0 || x>2 || y<0 || y>2)
			throw new CasillaOcupada();

		if (!tablero[x][y].equals(ficha.vacia))
			throw new CasillaOcupada();

		if (!estado.equals(estadoJuego.jugando))
			throw new JuegoParado();

		tablero[x][y]=turno;

		ficha f=vencedor();

		if (f.equals(ficha.cruz)) {
			estado=estadoJuego.victoriaCruz;
		}
		else if (f.equals(ficha.circulo)) {
			estado=estadoJuego.victoriaCirculo;
		}
		else {
			if (tablerolleno())
				estado=estadoJuego.tablas;
			else {
				if (turno.equals(ficha.cruz))
					turno=ficha.circulo;
				else
					turno=ficha.cruz;
			}
		}

		return tablero;
	}
```

```
	private void inicializaTablero() {
		tablero = new ficha[3][3];
		for (int i=0; i<3; i++)
			for (int j=0; j<3; j++)
				tablero[i][j]=ficha.vacia;
	}

	private ficha vencedor() {
		if		(tablero[0][0].equals(ficha.cruz)		&&
tablero[0][1].equals(ficha.cruz) && tablero[0][2].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[1][0].equals(ficha.cruz)		&&
tablero[1][1].equals(ficha.cruz) && tablero[1][2].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[2][0].equals(ficha.cruz)		&&
tablero[2][1].equals(ficha.cruz) && tablero[2][2].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[0][0].equals(ficha.cruz)		&&
tablero[1][0].equals(ficha.cruz) && tablero[2][0].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[0][1].equals(ficha.cruz)		&&
tablero[1][1].equals(ficha.cruz) && tablero[2][1].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[0][2].equals(ficha.cruz)		&&
tablero[1][2].equals(ficha.cruz) && tablero[2][2].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[0][0].equals(ficha.cruz)		&&
tablero[1][1].equals(ficha.cruz) && tablero[2][2].equals(ficha.cruz))
			return ficha.cruz;
		if		(tablero[0][2].equals(ficha.cruz)		&&
tablero[1][1].equals(ficha.cruz) && tablero[2][0].equals(ficha.cruz))
			return ficha.cruz;

		if		(tablero[0][0].equals(ficha.circulo)		&&
tablero[0][1].equals(ficha.circulo) && tablero[0][2].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[1][0].equals(ficha.circulo)		&&
tablero[1][1].equals(ficha.circulo) && tablero[1][2].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[2][0].equals(ficha.circulo)		&&
tablero[2][1].equals(ficha.circulo) && tablero[2][2].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[0][0].equals(ficha.circulo)		&&
tablero[1][0].equals(ficha.circulo) && tablero[2][0].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[0][1].equals(ficha.circulo)		&&
tablero[1][1].equals(ficha.circulo) && tablero[2][1].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[0][2].equals(ficha.circulo)		&&
tablero[1][2].equals(ficha.circulo) && tablero[2][2].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[0][0].equals(ficha.circulo)		&&
tablero[1][1].equals(ficha.circulo) && tablero[2][2].equals(ficha.circulo))
			return ficha.circulo;
		if		(tablero[0][2].equals(ficha.circulo)		&&
tablero[1][1].equals(ficha.circulo) && tablero[2][0].equals(ficha.circulo))
			return ficha.circulo;

		return ficha.vacia;
	}

	private boolean tablerolleno() {
		for (int i=0;i<3;i++)
			for(int j=0;j<3;j++)
				if (tablero[i][j].equals(ficha.vacia))
					return false;
		return true;
	}
```

```
}
```

JuegoServer.java

```
package TicTacToe;

import java.io.*;
import org.omg.CORBA.*;

public class JuegoServer {

      public static void main(String[] args) {
         try {
             ORB orb = ORB.init(args,null);
             Juego objJuego = new JuegoImpl();
             orb.connect(objJuego);

             PrintWriter salida = new PrintWriter (new FileWriter (new
File ("IOR.txt")));
             salida.print(orb.object_to_string(objJuego));
             salida.flush();
             salida.close();

             System.out.println("Juego listo");

             // Espero a que se pulse enter para terminar la ejecución
del servidor
             try {
                    BufferedReader teclado;
                    teclado       =       new       BufferedReader(new
InputStreamReader(System.in));
                    String linea=teclado.readLine();
             } catch (IOException ex) {}

             System.out.println("Finalizado el juego");
          } catch (IOException ex) {
             System.err.println("No  se  puede  escribir  el  IOR  en  el
fichero");
             ex.printStackTrace();
          } catch (Exception ex) {
             System.err.println("No se puede crear el servidor");
             ex.printStackTrace();
          }
      }
}
```

JugadorAplic.java

```
package TicTacToe;

import javax.swing.UIManager;

public class JugadorAplic {
  boolean packFrame = false;

  //Construct the application

  public JugadorAplic(String args[]) {
    VentanaAplic frame = new VentanaAplic(args);
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their
layout
    if (packFrame)
      frame.pack();
```

```
          else
            frame.validate();
          frame.setVisible(true);
        }
      //Main method

        public static void main(String[] args) {
          try  {
            UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
            //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
            //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
          }
          catch (Exception e) {
          }
          new JugadorAplic(args);
        }
      }
```

TicTacToe.html

```
      <HTML>
      <HEAD>
      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
      <TITLE>
      Pagina Web para jugar al Tic Tac Toe
      </TITLE>
      </HEAD>
      <BODY>
      Juego TicTacToe distribuido.<BR>
      <!--"CONVERTED_APPLET"-->
      <!-- CONVERTER VERSION 1.0 -->
      <OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
      WIDTH = 400 HEIGHT = 300 NAME = "TicTacToe" ALIGN = middle VSPACE = 0
HSPACE  =  0    codebase="http://java.sun.com/products/plugin/1.2/jinstall-12-
win32.cab#Version=1,2,0,0">
      <PARAM NAME = CODE VALUE = "TicTacToe.JugadorApplet" >
      <PARAM NAME = NAME VALUE = "TicTacToe" >

      <PARAM NAME="type" VALUE="application/x-java-applet;version=1.2">
      <COMMENT>
      <EMBED    type="application/x-java-applet;version=1.2"    java_CODE    =
"TicTacToe.JugadorApplet" NAME = "TicTacToe" WIDTH = 400 HEIGHT = 300 ALIGN =
middle         VSPACE         =         0         HSPACE         =         0
pluginspage="http://java.sun.com/products/plugin/1.2/plugin-
install.html"><NOEMBED></COMMENT>

      </NOEMBED></EMBED>
      </OBJECT>

      <!--
      <APPLET  CODE = "TicTacToe.JugadorApplet" WIDTH = 400 HEIGHT = 300 NAME
= "TicTacToe" ALIGN = middle VSPACE = 0 HSPACE = 0 >


      </APPLET>
      -->
      <!--"END_CONVERTED_APPLET"-->

      </BODY>
      </HTML>
```

JugadorApplet.java

```
package TicTacToe;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.omg.CORBA.*;
import java.io.*;
import java.applet.*;
import java.util.Properties;

public class JugadorApplet extends Applet {
  Label estado = new Label();
  Label error = new Label();
  Label casillas[][] = {{new Label(), new Label(), new Label()},
                {new Label(), new Label(), new Label()},
                {new Label(), new Label(), new Label()}};

      Juego juego;
      Timer tiempo;
      ficha turnoAntes;
      ficha suFicha;
      estadoJuego estadoAntes;

  //Construct the frame

  public void init() {
    try  {
      jbInit();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
       try {
             ORB orb = ORB.init(this,null);
             File ficheroEntrada = new File("D:\\Java\\IOR.txt");
             FileReader entrada = new FileReader( ficheroEntrada);
             char IOR[] = new char[(int)ficheroEntrada.length()];
             if                                    (entrada.read(IOR,0,
IOR.length)<ficheroEntrada.length()) {
                    System.err.println("No se ha podido leer el IOR");
                    estado.setText("ERROR");
                    return;
             }
             org.omg.CORBA.Object  objJuego  =  orb.string_to_object(new
String(IOR));
             juego = JuegoHelper.narrow(objJuego);
             fichaHolder aux = new fichaHolder();
             turnoAntes = juego.turno();
             estadoAntes = juego.estado();
             ficha tabl[][] = juego.anadirJugador(aux);
             suFicha=aux.value;
             dibujarTablero(tabl);
             tiempo = new Timer(1000,new java.awt.event.ActionListener()
{
            public void actionPerformed(ActionEvent e) {
            tiempoTranscurrido(e);
            }
            });
             tiempo.start();
       } catch (YaHayDosJugadores ex) {
             System.err.println("Ya hay dos jugadores");
             estado.setText("ERROR");
/*     } catch (IOException ex) {
             System.err.println("No se ha podido leer el IOR");
             estado.setText("ERROR");*/
       } catch (Exception ex) {
```

```
                    System.err.println("No  se  ha  podido  conectar  con  el
servidor");
                    estado.setText("ERROR");
                    ex.printStackTrace();
                }
            }
        //Component initialization

          private void jbInit() throws Exception  {
            this.setLayout(null);
            this.setSize(new Dimension(400, 300));
            estado.setBounds(new Rectangle(42, 25, 100, 20));
            error.setBounds(new Rectangle(150, 25, 160, 20));

            for (int i=0;i<3;i++)
              for (int j=0;j<3;j++) {
                casillas[i][j].setBackground(Color.blue);
                casillas[i][j].setForeground(Color.red);
                casillas[i][j].setFont(new Font("Times new roman",0,30));
              }

            error.setBackground(Color.white);
            estado.setBackground(Color.white);

            for (int i=0;i<3;i++)
              for (int j=0;j<3;j++)
                {
                  final int x=i;
                  final int y=j;
                  casillas[i][j].addMouseListener(new
java.awt.event.MouseAdapter() {
                    public void mouseClicked(MouseEvent e) {
                    label4_mouseClicked(e,x,y);
                    }
                    });
                }

            this.add(estado, null);
            this.add(error, null);
            casillas[0][0].setBounds(new Rectangle(42, 65, 40, 40));
            this.add(casillas[0][0], null);
            casillas[0][1].setBounds(new Rectangle(87, 65, 40, 40));
            this.add(casillas[0][1], null);
            casillas[0][2].setBounds(new Rectangle(131, 65, 40, 40));
            this.add(casillas[0][2], null);
            casillas[1][0].setBounds(new Rectangle(42, 116, 40, 40));
            this.add(casillas[1][0], null);
            casillas[1][1].setBounds(new Rectangle(87, 116, 40, 40));
            this.add(casillas[1][1], null);
            casillas[1][2].setBounds(new Rectangle(131, 116, 40, 40));
            this.add(casillas[1][2], null);
            casillas[2][0].setBounds(new Rectangle(42, 167, 40, 40));
            this.add(casillas[2][0], null);
            casillas[2][1].setBounds(new Rectangle(87, 167, 40, 40));
            this.add(casillas[2][1], null);
            casillas[2][2].setBounds(new Rectangle(131, 167, 40, 40));
            this.add(casillas[2][2], null);

          }

               private void dibujarTablero(ficha[][] tablero) {

                     for (int i=0;i<3;i++)
                           for (int j=0;j<3;j++) {
                                 if (tablero[i][j]==ficha.cruz)
                                       casillas[i][j].setText(" X");
                                 else if (tablero[i][j]==ficha.circulo)
                                       casillas[i][j].setText(" O");
```

```
                                else
                                        casillas[i][j].setText("");
                                casillas[i][j].invalidate();
                        }
                if (estadoAntes.equals(estadoJuego.victoriaCirculo)) {
                        if (suFicha.equals(ficha.circulo))
                                estado.setText("Ha vencido");
                        else
                                estado.setText("Ha perdido");
                }
                else if (estadoAntes.equals(estadoJuego.victoriaCruz)) {
                        if (suFicha.equals(ficha.cruz))
                                estado.setText("Ha vencido");
                        else
                                estado.setText("Ha perdido");
                }
                else if (estadoAntes.equals(estadoJuego.tablas)) {
                        estado.setText("Tablas");
                }
                else if (estadoAntes.equals(estadoJuego.inicializando)) {
                        estado.setText("Espere jugador");
                }
                else {
                        if (turnoAntes.equals(suFicha))
                                estado.setText("Su turno");
                        else
                                estado.setText("Espere su turno");
                }
        }

        public void tiempoTranscurrido(ActionEvent e) {
                if (!estadoAntes.equals(juego.estado())) {
                        estadoAntes=juego.estado();
                        dibujarTablero(juego.tablero());
                        if (!estadoAntes.equals(estadoJuego.jugando))
                                tiempo.stop();
                }
                else if (!turnoAntes.equals(juego.turno())) {
                        turnoAntes=juego.turno();
                        dibujarTablero(juego.tablero());
                }
        }

    //Get Applet information

      public String getAppletInfo() {
        return "Applet Information";
      }
    //Get parameter info

      public String[][] getParameterInfo() {
        return null;
      }

        void label4_mouseClicked(MouseEvent e,int i,int j) {
                if (!estadoAntes.equals(estadoJuego.jugando))
                        return;

                try {
                        error.setText("");
                        dibujarTablero(juego.ponerFicha((short)i,
(short)j));
                } catch (TicTacToe.JuegoParado ex) {
                        error.setText("No puede colocar fichas");
                } catch (CasillaOcupada ex) {
                        error.setText("Mala posición");
                }
        }
```

```
}
```

A.2 Uso de Callback

El código que se muestra a continuación corresponde con el juego *TicTacToe* al añadirle el uso de *callback's*, capítulo 3.7

TicTacToe.idl

```
module TicTacToe {
	enum ficha {circulo,cruz,vacia};
	enum estadoJugador {jugando, victoria, tablas, derrota};
	typedef ficha tablero[3][3];

	exception CasillaOcupada {};
	exception JugadorInvalido {};
	exception ErrorEnServidor {};
	exception YaHayDosJugadores {};

	interface Jugador {
		readonly attribute boolean meToca;
		readonly attribute tablero tab;
		readonly attribute estadoJugador estado;
		readonly attribute ficha usada;

		oneway void turno(in boolean es,in tablero tab);
		oneway void fin(in boolean victoria, in boolean tablas, in
tablero tab);
		void ponerFicha(in short x, in short y)
			raises(ErrorEnServidor,CasillaOcupada);
	};

	interface Juego {
		readonly attribute tablero tab;

		tablero anadirJugador(in Jugador jug,out ficha usar)
			raises(YaHayDosJugadores);
		tablero ponerFicha(in Jugador jug, in short x, in short y)
			raises(JugadorInvalido,CasillaOcupada);
	};
};
```

JuegoImpl.java

```
package TicTacToe;

import TicTacToe.*;
import org.omg.CORBA.*;

public class JuegoImpl extends _JuegoImplBase {
  private ficha[][] tablero;
  private Jugador jug1,jug2;
  private estadoJugador estado;
  private int numeroServidor;
  private ORB orb;
  private String IORJug1;
  private String IORJug2;
  private String IORturno;

  public JuegoImpl(ORB orb) {
    super();

    this.orb=orb;
```

```
          inicializaTablero();
          estado=estadoJugador.jugando;
        }

        private void inicializaTablero() {
          tablero = new ficha[3][3];
          for (int i=0; i<3; i++)
            for (int j=0; j<3; j++)
              tablero[i][j]=ficha.vacia;
        }

        //readonly attribute tablero tab;
        public ficha[][] tab() {
          return tablero;
        }

        //tablero anadirJugador(in Jugador jug,out ficha usar);
          //raises(YaHayDosJugadores);
        public ficha[][] anadirJugador(Jugador jug, fichaHolder usar)
          throws TicTacToe.YaHayDosJugadores {

          if (jug1==null) {
            jug1=jug;
            usar.value=ficha.cruz;
            IORJug1=orb.object_to_string(jug1);
            System.out.println("Jugador 1 Añadido: " + IORJug1);
          }
          else if (jug2==null) {
            jug2=jug;
            usar.value=ficha.circulo;
            IORJug2=orb.object_to_string(jug2);
            System.out.println("Jugador 2 Añadido: "+ IORJug2);
            jug1.turno(true,tablero);
            jug2.turno(false,tablero);
            IORturno=IORJug1;
          }
          else {
            System.out.println("ERROR :Jugador no Añadido");
            throw new TicTacToe.YaHayDosJugadores();
          }

          return tablero;
        }

        private ficha vencedor() {
          if   (tablero[0][0]==ficha.cruz   &&   tablero[0][1]==ficha.cruz   &&
tablero[0][2]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[1][0]==ficha.cruz   &&   tablero[1][1]==ficha.cruz   &&
tablero[1][2]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[2][0]==ficha.cruz   &&   tablero[2][1]==ficha.cruz   &&
tablero[2][2]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[0][0]==ficha.cruz   &&   tablero[1][0]==ficha.cruz   &&
tablero[2][0]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[0][1]==ficha.cruz   &&   tablero[1][1]==ficha.cruz   &&
tablero[2][1]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[0][2]==ficha.cruz   &&   tablero[1][2]==ficha.cruz   &&
tablero[2][2]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[0][0]==ficha.cruz   &&   tablero[1][1]==ficha.cruz   &&
tablero[2][2]==ficha.cruz)
            return ficha.cruz;
          if   (tablero[0][2]==ficha.cruz   &&   tablero[1][1]==ficha.cruz   &&
tablero[2][0]==ficha.cruz)
```

```
            return ficha.cruz;

          if (tablero[0][0]==ficha.circulo && tablero[0][1]==ficha.circulo &&
tablero[0][2]==ficha.circulo)
            return ficha.circulo;
          if (tablero[1][0]==ficha.circulo && tablero[1][1]==ficha.circulo &&
tablero[1][2]==ficha.circulo)
            return ficha.circulo;
          if (tablero[2][0]==ficha.circulo && tablero[2][1]==ficha.circulo &&
tablero[2][2]==ficha.circulo)
            return ficha.circulo;
          if (tablero[0][0]==ficha.circulo && tablero[1][0]==ficha.circulo &&
tablero[2][0]==ficha.circulo)
            return ficha.circulo;
          if (tablero[0][1]==ficha.circulo && tablero[1][1]==ficha.circulo &&
tablero[2][1]==ficha.circulo)
            return ficha.circulo;
          if (tablero[0][2]==ficha.circulo && tablero[1][2]==ficha.circulo &&
tablero[2][2]==ficha.circulo)
            return ficha.circulo;
          if (tablero[0][0]==ficha.circulo && tablero[1][1]==ficha.circulo &&
tablero[2][2]==ficha.circulo)
            return ficha.circulo;
          if (tablero[0][2]==ficha.circulo && tablero[1][1]==ficha.circulo &&
tablero[2][0]==ficha.circulo)
            return ficha.circulo;

          return ficha.vacia;
        }

        private boolean tablerolleno() {
          for (int i=0;i<3;i++)
            for(int j=0;j<3;j++)
              if (tablero[i][j]==ficha.vacia)
                return false;
          return true;
        }

        //tablero ponerFicha(in Jugador jug, in short x, in short y);
          //raises(JugadorInvalido,CasillaOcupada);
        public ficha[][] ponerFicha(Jugador jug, short x,short y)
          throws TicTacToe.JugadorInvalido, CasillaOcupada {

          if (!estado.equals(estadoJugador.jugando))
            throw new TicTacToe.JugadorInvalido();

          if (x<0 || x>2 || y<0 || y>2)
            throw new CasillaOcupada();

          if (tablero[x][y]!=ficha.vacia)
            throw new CasillaOcupada();

          if (jug1==null || jug2==null)
            throw new TicTacToe.JugadorInvalido();

             String IORJug = orb.object_to_string(jug);

          if (!IORJug.equals(IORturno))
            throw new TicTacToe.JugadorInvalido();

          if (jug.usada().equals(jug1.usada()))
            tablero[x][y]=ficha.cruz;
          else
            tablero[x][y]=ficha.circulo;

          ficha f=vencedor();

          if (f.equals(ficha.cruz)) {
```

```
            jug1.fin(true,false,tablero);
            jug2.fin(false,false,tablero);
            estado=estadoJugador.victoria;
          }
          else if (f.equals(ficha.circulo)) {
            jug1.fin(false,false,tablero);
            jug2.fin(true,false,tablero);
            estado=estadoJugador.derrota;
          }
          else {
            if (tablerolleno()) {
              jug1.fin(false,true,tablero);
              jug2.fin(false,true,tablero);
              estado=estadoJugador.tablas;
            }
            else {
              if (jug.usada().equals(jug1.usada())) {
                jug2.turno(true,tablero);
                jug1.turno(false,tablero);
               IORturno=IORJug2;
              }
              else {
                jug1.turno(true,tablero);
                jug2.turno(false,tablero);
               IORturno=IORJug1;
              }
            }
          }

          return tablero;
        }

      }
```

JugadorImpl.java

```
      package TicTacToe;

      import TicTacToe.*;
      import org.omg.CosNaming.*;
      import java.io.*;

      public class JugadorImpl extends _JugadorImplBase {
            private boolean meToca;
            private ficha[][] tablero;
            private Juego juego;
            private estadoJugador estado;
            private ficha usada;
            private JugadorInter jugador;

            public JugadorImpl(org.omg.CORBA.ORB orb,JugadorInter jug)
                  throws TicTacToe.YaHayDosJugadores {

                  super();

                  fichaHolder fh=new fichaHolder();
                  meToca=false;
               try {
                  File ficheroEntrada = new File("D:\\Java\\IOR.txt");
                  FileReader entrada = new FileReader( ficheroEntrada);
                  char IOR[] = new char[(int)ficheroEntrada.length()];
                  if                                    (entrada.read(IOR,0,
IOR.length)<ficheroEntrada.length()) {
                        System.err.println("No se ha podido leer el IOR");
                  }
                  org.omg.CORBA.Object  objJuego  =  orb.string_to_object(new
String(IOR));
```

```
                juego = JuegoHelper.narrow(objJuego);

                tablero=juego.anadirJugador(this,fh);
            } catch (Exception ex) {
                System.err.println("Error al conectar con el servidor");
            }

                usada=fh.value;
                estado=estadoJugador.jugando;
                jugador=jug;
        }

        // oneway void turno(in tablero tab);
        public void turno(boolean es,ficha[][] tab) {
                tablero=tab;
                meToca=es;
                jugador.estadoCambiado(estado,meToca);
        }

        //readonly attribute boolean meToca;
        public boolean meToca() {
                return meToca;
        }

        //readonly attribute estadoJugador estado;
        public estadoJugador estado() {
                return estado;
        }

        //readonly attribute tablero tab;
        public ficha[][] tab() {
                return tablero;
        }

        //oneway void fin(in boolean victoria, in boolean tablas, in
tablero tab);
        public void fin(boolean victoria, boolean tablas,ficha[][] tab) {
                if (victoria)
                        estado=estadoJugador.victoria;
                else if (tablas)
                        estado=estadoJugador.tablas;
                else
                        estado=estadoJugador.derrota;
                tablero=tab;
                jugador.estadoCambiado(estado,meToca);
        }

        //void ponerFicha(in short x, in short y)
                //raises(ErrorEnServidor,CasillaOcupada);
        public void ponerFicha(short x,short y)
                throws TicTacToe.ErrorEnServidor, CasillaOcupada {

                try {
                        tablero=juego.ponerFicha(this,x,y);
                } catch (TicTacToe.JugadorInvalido e) {
                        throw new TicTacToe.ErrorEnServidor();
                }
        }

        //readonly attribute ficha usada;
        public ficha usada() {
                return usada;
        }
    }
```

JugadorInter.java

```
package TicTacToe;

public interface JugadorInter {
  public void estadoCambiado(estadoJugador est,boolean meToca);
}
```

JuegoServer.java

```
package TicTacToe;

import java.io.*;
import org.omg.CORBA.*;

public class JuegoServer {

       public static void main(String[] args) {
          try {
              ORB orb = ORB.init(args,null);
              Juego objJuego = new JuegoImpl(orb);
              orb.connect(objJuego);

              PrintWriter salida = new PrintWriter (new FileWriter (new
File ("IOR.txt")));
              salida.print(orb.object_to_string(objJuego));
              salida.flush();
              salida.close();

              System.out.println("Juego listo");

              // Espero a que se pulse enter para terminar la ejecución
del servidor
              try {
                     BufferedReader teclado;
                     teclado       =       new       BufferedReader(new
InputStreamReader(System.in));
                     String linea=teclado.readLine();
              } catch (IOException ex) {}

              System.out.println("Finalizado el juego");
           } catch (IOException ex) {
              System.err.println("No  se  puede  escribir  el  IOR  en  el
fichero");
              ex.printStackTrace();
           } catch (Exception ex) {
              System.err.println("No se puede crear el servidor");
              ex.printStackTrace();
           }
       }
}
```

JugadorTextoAplic.java

```
package TicTacToe;

import java.awt.*;
import java.awt.event.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import javax.swing.*;
import java.io.*;

public class JugadorTextoAplic implements JugadorInter {
  private TicTacToe.Jugador jugador;
  private BufferedReader teclado;
```

```
  static public void main(String[] args) {
    (new JugadorTextoAplic(args)).jugar();
  }

  public JugadorTextoAplic(String[] args) {
    try {
      ORB orb = ORB.init(args,null);
      jugador = new JugadorImpl(orb,this);
      teclado = new BufferedReader(new InputStreamReader(System.in));
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }

  private void dibujarTablero() {
    ficha f=jugador.usada();
    ficha[][] tablero = jugador.tab();

    System.out.println();
    for (int i=0;i<3;i++) {
      for (int j=0;j<3;j++)
        {
        if (j!=0)
          System.out.print("|");
         if (tablero[i][j]==ficha.cruz)
           System.out.print(" X ");
         else if (tablero[i][j]==ficha.circulo)
           System.out.print(" O ");
         else
           System.out.print("   ");
        }
      System.out.println();
    }
  }

  public void estadoCambiado(estadoJugador est,boolean meToca) {
    dibujarTablero();
    if (est.equals(estadoJugador.jugando)) {
      if (meToca) {
       String entradaX, entradaY;
       int valorX, valorY;
        System.out.println("Su turno");
       while (true) {
         try {
           System.out.print("Coordenada X (1-3): ");
           entradaX = teclado.readLine();
           valorX = Integer.parseInt(entradaX);
//         if (valorX<1 || valorX>3) continue;
           System.out.print("Coordenada Y (1-3): ");
           entradaY = teclado.readLine();
           valorY = Integer.parseInt(entradaY);
//         if (valorY<1 || valorY>3) continue;
            jugador.ponerFicha((short)(valorY-1), (short)(valorX-1));
            dibujarTablero();
           break;
         } catch(NumberFormatException ex) {
           continue;
          } catch (TicTacToe.ErrorEnServidor ex) {
            System.out.println("El servidor ha fallado");
          } catch (CasillaOcupada ex) {
            System.out.println("Casilla Ocupada, o inexistente");
          } catch (IOException ex) {
           ex.printStackTrace();
         }

       }
```

```
        }
        else
          System.out.println("Espere su turno");
      }
      else if (est.equals(estadoJugador.victoria))
        System.out.println("Ha vencido");
      else if (est.equals(estadoJugador.derrota))
        System.out.println("Ha perdido");
      else if (est.equals(estadoJugador.tablas))
        System.out.println("TABLAS");
    }

    public void jugar() {
      dibujarTablero();
      System.out.println("Espere ...");
      while (true);
    }

  }
```

JugadorApplet.java

```
package TicTacToe;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class JugadorApplet extends Applet implements JugadorInter {
  private TicTacToe.Jugador jugador;

  boolean isStandalone = false;
  Label casillas[][] = {{new Label(), new Label(), new Label()},
                {new Label(), new Label(), new Label()},
                {new Label(), new Label(), new Label()}};
  Label estado = new Label();
  Label error = new Label();
//Get a parameter value

  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }

  //Construct the applet

//  public JugadorApplet() {
//  }
//Initialize the applet

  public void init() {
    try {
    jbInit();
    }
    catch (Exception e) {
    e.printStackTrace();
    }
      try {
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(this,null);
      jugador = new JugadorImpl(orb,this);
    }
    catch (Exception e) {
      e.printStackTrace();
    }

  }
  //static {
```

```
        //  try {
        //                                      //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
        //                                      //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
        //                                       UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
        //  }
        //  catch (Exception e) {}
        //}
      //Component initialization

        private void jbInit() throws Exception {
          this.setLayout(null);
          this.setSize(new Dimension(400, 300));
          estado.setBounds(new Rectangle(42, 25, 100, 20));
          error.setBounds(new Rectangle(150, 25, 160, 20));

          for (int i=0;i<3;i++)
            for (int j=0;j<3;j++) {
              casillas[i][j].setBackground(Color.blue);
              casillas[i][j].setForeground(Color.red);
              casillas[i][j].setFont(new Font("Times new roman",0,30));
            }

          error.setBackground(Color.white);
          estado.setBackground(Color.white);
             estado.setText("Espere...");

          for (int i=0;i<3;i++)
            for (int j=0;j<3;j++)
              {
                final int x=i;
                final int y=j;
                casillas[i][j].addMouseListener(new
java.awt.event.MouseAdapter() {
                  public void mouseClicked(MouseEvent e) {
                  label4_mouseClicked(e,x,y);
                  }
                  });
              }

          this.add(estado, null);
          this.add(error, null);
          casillas[0][0].setBounds(new Rectangle(42, 65, 40, 40));
          this.add(casillas[0][0], null);
          casillas[0][1].setBounds(new Rectangle(87, 65, 40, 40));
          this.add(casillas[0][1], null);
          casillas[0][2].setBounds(new Rectangle(131, 65, 40, 40));
          this.add(casillas[0][2], null);
          casillas[1][0].setBounds(new Rectangle(42, 116, 40, 40));
          this.add(casillas[1][0], null);
          casillas[1][1].setBounds(new Rectangle(87, 116, 40, 40));
          this.add(casillas[1][1], null);
          casillas[1][2].setBounds(new Rectangle(131, 116, 40, 40));
          this.add(casillas[1][2], null);
          casillas[2][0].setBounds(new Rectangle(42, 167, 40, 40));
          this.add(casillas[2][0], null);
          casillas[2][1].setBounds(new Rectangle(87, 167, 40, 40));
          this.add(casillas[2][1], null);
          casillas[2][2].setBounds(new Rectangle(131, 167, 40, 40));
          this.add(casillas[2][2], null);

        }
      //Get Applet information

        public String getAppletInfo() {
          return "Applet Information";
```

```
  }
//Get parameter info

  public String[][] getParameterInfo() {
    return null;
  }

  private void dibujarTablero() {
    ficha f=jugador.usada();
    ficha[][] tablero = jugador.tab();

    for (int i=0;i<3;i++)
      for (int j=0;j<3;j++)
        {
          if (tablero[i][j]==ficha.cruz)
          {
            System.out.println("Ficha cruz");
            casillas[i][j].setText(" X");
          }
          else if (tablero[i][j]==ficha.circulo)
            casillas[i][j].setText(" O");
          else
            casillas[i][j].setText("");
          casillas[i][j].invalidate();
        }
  }

  void label4_mouseClicked(MouseEvent e,int i,int j) {
    try {
      error.setText("");
      jugador.ponerFicha((short)i, (short)j);
      dibujarTablero();
    } catch (TicTacToe.ErrorEnServidor ex) {
      error.setText("No puede colocar fichas");
    } catch (CasillaOcupada ex) {
      error.setText("Mala posición");
    }
  }

  public void estadoCambiado(estadoJugador est,boolean meToca) {
    dibujarTablero();
    if (est.equals(estadoJugador.jugando)) {
      if (meToca)
        estado.setText("Su turno");
      else
        estado.setText("Espere su turno");
    }
    else if (est.equals(estadoJugador.victoria))
      estado.setText("Ha vencido");
    else if (est.equals(estadoJugador.derrota))
      estado.setText("Ha perdido");
    else if (est.equals(estadoJugador.tablas))
      estado.setText("TABLAS");
  }
}
```

A.3 Servicio de nombres

Los ficheros modificados para que el juego *TicTacToe* usase el servicio de nombres (capítulo 3.8) pueden verse a continuación.

JugadorImpl.java

```
package TicTacToe;

import TicTacToe.*;
import org.omg.CosNaming.*;

public class JugadorImpl extends _JugadorImplBase {
  private boolean meToca;
  private ficha[][] tablero;
  private Juego juego;
  private estadoJugador estado;
  private ficha usada;
  private JugadorInter jugador;

  public JugadorImpl(org.omg.CORBA.ORB orb,JugadorInter jug)
    throws TicTacToe.YaHayDosJugadores {
    super();

    fichaHolder fh=new fichaHolder();
    meToca=false;
    try {
      org.omg.CORBA.Object obj =
orb.resolve_initial_references("NameService");
      NamingContext root_context = NamingContextHelper.narrow(obj);
      if (root_context==null) {
        System.out.println("Error: Servicio de nombres no encontrado");
      }

      NameComponent nombre[] = new NameComponent[1];
      nombre[0]=new NameComponent("TicTacToe","");
      NamingContext servidores;

      System.out.println("Servicio de nombres encontrado");

      try {
        servidores =
NamingContextHelper.narrow(root_context.resolve(nombre));
      } catch (org.omg.CosNaming.NamingContextPackage.NotFound ex) {
        System.out.println("Error: No existe ningún servidor de
TicTacToe");
        return;
      }

      org.omg.CORBA.Object juegoObj;
      // Dos formas de encontrar el servidor:
      // 1. Buscar en todos los nombres conocidos
        // Servidor + i

/*      int i=0;
      while (true) {
        try {
          nombre[0]=new NameComponent("Servidor" + i,"");
          juegoObj=servidores.resolve(nombre);
          juego=JuegoHelper.narrow(juegoObj);
          tablero=juego.anadirJugador(this,fh);
          // Si lo consigue asociar termina
          break;
        } catch (Exception ex) {
          // Mientras no lo pueda asociar porque exista otro
          // servidor continua intentandolo
          i++;
          if (i==100) {
            System.out.println("Error: No existe ningún servidor de
TicTacToe");
          }
        }
      }
```

```
            System.out.println("Encontrado servidor : Servidor" + i);
            */

          // 2. Obtener la lista de servidores activos

          BindingListHolder lista = new BindingListHolder();
          BindingIteratorHolder listaiter = new BindingIteratorHolder();

          servidores.list(100,lista,listaiter);

          int i=0;
          while (i<lista.value.length) {
            try {
              juegoObj=servidores.resolve(lista.value[i].binding_name);
              juego=JuegoHelper.narrow(juegoObj);
              tablero=juego.anadirJugador(this,fh);
              break;
            } catch (Exception ex) {
              juego=null;
              i++;
            }
          }
          if (juego==null) {
            System.out.println("Error:   No   existe   ningún   servidor   de
TicTacToe");
          }

          }catch (Exception ex) {
            ex.printStackTrace();
          }

          usada=fh.value;
          estado=estadoJugador.jugando;
          jugador=jug;
        }
        // oneway void turno(in tablero tab);
        public void turno(boolean es,ficha[][] tab) {
          tablero=tab;
          meToca=es;
          jugador.estadoCambiado(estado,meToca);
        }

        //readonly attribute boolean meToca;
        public boolean meToca() {
          return meToca;
        }

        //readonly attribute estadoJugador estado;
        public estadoJugador estado() {
          return estado;
        }

        //readonly attribute tablero tab;
        public ficha[][] tab() {
          return tablero;
        }

        //oneway void fin(in boolean victoria, in boolean tablas, in tablero
tab);
        public void fin(boolean victoria, boolean tablas,ficha[][] tab) {
          if (victoria)
            estado=estadoJugador.victoria;
          else if (tablas)
            estado=estadoJugador.tablas;
          else
            estado=estadoJugador.derrota;
          tablero=tab;
          jugador.estadoCambiado(estado,meToca);
```

```
  }

  //void ponerFicha(in short x, in short y)
      //raises(ErrorEnServidor,CasillaOcupada);
  public void ponerFicha(short x,short y)
    throws TicTacToe.ErrorEnServidor, CasillaOcupada {

    try {
      tablero=juego.ponerFicha(this,x,y);
    } catch (TicTacToe.JugadorInvalido e) {
      throw new TicTacToe.ErrorEnServidor();
    }
  }

  //readonly attribute ficha usada;
  public ficha usada() {
    return usada;
  }
}
```

JuegoServer.java

```
package TicTacToe;

import java.util.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;

public class JuegoServer {

  public static void main(String[] args) {
    try {
      ORB orb = ORB.init(args,null);
      org.omg.CORBA.Object obj;
      try {
         obj = orb.resolve_initial_references("NameService");
      } catch (Exception ex) {
        System.out.println("Error: Servicio de nombres no encontrado");
        System.exit(-1);
        return;
      }
      NamingContext root_context = NamingContextHelper.narrow(obj);
      if (root_context==null) {
        System.out.println("Error: Servicio de nombres no encontrado");
        System.exit(-1);
      }
      TicTacToe.Juego implObject = new JuegoImpl(orb);
      NameComponent nombre[] = new NameComponent[1];
      nombre[0]=new NameComponent("TicTacToe","");
      NamingContext servidores;
      try {
        servidores                                                    =
NamingContextHelper.narrow(root_context.resolve(nombre));
      } catch (org.omg.CosNaming.NamingContextPackage.NotFound ex) {
        servidores = root_context.bind_new_context(nombre);
      }
      int i=0;
      while (true) {
        try {
          nombre[0]=new NameComponent("Servidor" + i,"");
          servidores.bind(nombre,implObject);
          // Si lo consigue asociar termina
          break;
        } catch (org.omg.CosNaming.NamingContextPackage.AlreadyBound ex)
{
          // Mientras no lo pueda asociar porque exista otro
          // servidor continua intentandolo
```

```
                i++;
                if (i==100) {
                  System.out.println("Error: No se pueden tener más de 100
servidores");
                  System.exit(-1);
                }
              }
            }

            System.out.println(implObject+ " está listo. Como Servidor" + i);
            try {
              int x = System.in.read();
            } catch (Exception ex) {}
            System.out.println("Ha salido");
            servidores.unbind(nombre);
          }
          catch (Exception e) {
            e.printStackTrace();
          }
        }
      }
```

A.4 TicTacToe en C++

Para la implementación en C++, se desarrolló el siguiete código.

JuegoImpl.h

```
#ifndef JUEGOIMPL_H
#define JUEGOIMPL_H

#include <OB/CORBA.h>
#include "TicTacToe_skel.h"

class TicTacToe_Juego_Impl : public TicTacToe_Juego_skel {

TicTacToe_tablero_var tablero;
TicTacToe_Jugador_var jug1,jug2;
TicTacToe_estadoJugador estado;
CORBA_ORB_var orb;
CORBA_String_var IORJug1;
CORBA_String_var IORJug2;
CORBA_String_var IORturno;
void inicializaTablero();
TicTacToe_ficha vencedor();
bool tablerolleno();

public:

TicTacToe_Juego_Impl(CORBA_ORB_ptr orbArg);
TicTacToe_tablero_slice* tab();

TicTacToe_tablero_slice*    anadirJugador(TicTacToe_Jugador_ptr    jug,
TicTacToe_ficha& usar);

TicTacToe_tablero_slice*      ponerFicha(TicTacToe_Jugador_ptr      jug,
CORBA_Short x, CORBA_Short y);
};

#endif
```

JuegoImpl.cpp

```
#include "JuegoImpl.h"
```

```
      #include <string.h>

      TicTacToe_Juego_Impl::TicTacToe_Juego_Impl(CORBA_ORB_ptr orbArg) {
          orb=orbArg;
          inicializaTablero();
          estado=TicTacToe_jugando;
          jug1=TicTacToe_Jugador::_nil();
          jug2=TicTacToe_Jugador::_nil();
      }

      void TicTacToe_Juego_Impl::inicializaTablero() {
          tablero = new TicTacToe_ficha[3][3];
          for (int i=0; i<3; i++)
            for (int j=0; j<3; j++)
              tablero[i][j]=TicTacToe_vacia;
      }

      TicTacToe_tablero_slice* TicTacToe_Juego_Impl::tab() {
         return TicTacToe_tablero_dup(tablero);
      }

      TicTacToe_tablero_slice*
TicTacToe_Juego_Impl::anadirJugador(TicTacToe_Jugador_ptr                    jug,
TicTacToe_ficha& usar) {
          if (CORBA_is_nil(jug1)) {
            jug1=TicTacToe_Jugador::_duplicate(jug);
            usar=TicTacToe_cruz;
            IORJug1=CORBA_string_dup(orb->object_to_string(jug1));
            cout << "Jugador 1 Anadido: " << IORJug1 << endl;
          }

          else if (CORBA_is_nil(jug2)) {
            jug2=TicTacToe_Jugador::_duplicate(jug);
            usar=TicTacToe_circulo;
            IORJug2=CORBA_string_dup(orb->object_to_string(jug2));
            cout << "Jugador 2 Anadido: " << IORJug2 << endl;
            jug1->turno(true,tablero);
            jug2->turno(false,tablero);
            IORturno=CORBA_string_dup(IORJug1);
          }
          else {
             cerr << "ERROR :Jugador no Anadido" << endl;
             throw TicTacToe_YaHayDosJugadores();
          }

          return TicTacToe_tablero_dup(tablero);

      }

      TicTacToe_ficha TicTacToe_Juego_Impl::vencedor() {
          if  (tablero[0][0]==TicTacToe_cruz  &&  tablero[0][1]==TicTacToe_cruz
&& tablero[0][2]==TicTacToe_cruz)
            return TicTacToe_cruz;
          if  (tablero[1][0]==TicTacToe_cruz  &&  tablero[1][1]==TicTacToe_cruz
&& tablero[1][2]==TicTacToe_cruz)
            return TicTacToe_cruz;
          if  (tablero[2][0]==TicTacToe_cruz  &&  tablero[2][1]==TicTacToe_cruz
&& tablero[2][2]==TicTacToe_cruz)
            return TicTacToe_cruz;
          if  (tablero[0][0]==TicTacToe_cruz  &&  tablero[1][0]==TicTacToe_cruz
&& tablero[2][0]==TicTacToe_cruz)
            return TicTacToe_cruz;
          if  (tablero[0][1]==TicTacToe_cruz  &&  tablero[1][1]==TicTacToe_cruz
&& tablero[2][1]==TicTacToe_cruz)
            return TicTacToe_cruz;
          if  (tablero[0][2]==TicTacToe_cruz  &&  tablero[1][2]==TicTacToe_cruz
&& tablero[2][2]==TicTacToe_cruz)
            return TicTacToe_cruz;
```

```
          if (tablero[0][0]==TicTacToe_cruz && tablero[1][1]==TicTacToe_cruz
&& tablero[2][2]==TicTacToe_cruz)
            return TicTacToe_cruz;
          if (tablero[0][2]==TicTacToe_cruz && tablero[1][1]==TicTacToe_cruz
&& tablero[2][0]==TicTacToe_cruz)
            return TicTacToe_cruz;

          if               (tablero[0][0]==TicTacToe_circulo               &&
tablero[0][1]==TicTacToe_circulo && tablero[0][2]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[1][0]==TicTacToe_circulo               &&
tablero[1][1]==TicTacToe_circulo && tablero[1][2]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[2][0]==TicTacToe_circulo               &&
tablero[2][1]==TicTacToe_circulo && tablero[2][2]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[0][0]==TicTacToe_circulo               &&
tablero[1][0]==TicTacToe_circulo && tablero[2][0]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[0][1]==TicTacToe_circulo               &&
tablero[1][1]==TicTacToe_circulo && tablero[2][1]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[0][2]==TicTacToe_circulo               &&
tablero[1][2]==TicTacToe_circulo && tablero[2][2]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[0][0]==TicTacToe_circulo               &&
tablero[1][1]==TicTacToe_circulo && tablero[2][2]==TicTacToe_circulo)
            return TicTacToe_circulo;
          if               (tablero[0][2]==TicTacToe_circulo               &&
tablero[1][1]==TicTacToe_circulo && tablero[2][0]==TicTacToe_circulo)
            return TicTacToe_circulo;

          return TicTacToe_vacia;
      }

      bool TicTacToe_Juego_Impl::tablerolleno() {
          for (int i=0;i<3;i++)
            for(int j=0;j<3;j++)
              if (tablero[i][j]==TicTacToe_vacia)
                return false;
          return true;

      }

      TicTacToe_tablero_slice*
TicTacToe_Juego_Impl::ponerFicha(TicTacToe_Jugador_ptr   jug,   CORBA_Short   x,
CORBA_Short y) {
          if (estado!=TicTacToe_jugando)
            throw TicTacToe_JugadorInvalido();

          if (x<0 || x>2 || y<0 || y>2)
            throw TicTacToe_CasillaOcupada();

          if (tablero[x][y]!=TicTacToe_vacia)
            throw TicTacToe_CasillaOcupada();

          if (CORBA_is_nil(jug1) || CORBA_is_nil(jug2))
            throw TicTacToe_JugadorInvalido();

          CORBA_String_var        IORJug        =        CORBA_string_dup(orb-
>object_to_string(jug));

          if (strcmp(IORJug,IORturno)) {
             cout << " No te toca " << endl;
            throw TicTacToe_JugadorInvalido();
          }

          if (!strcmp(IORJug,IORJug1))
```

```
            tablero[x][y]=TicTacToe_cruz;
          else
            tablero[x][y]=TicTacToe_circulo;
          TicTacToe_ficha f=vencedor();

          if (f==TicTacToe_cruz) {
            jug1->fin(true,false,tablero);
            jug2->fin(false,false,tablero);
            estado=TicTacToe_victoria;
          }
          else if (f==TicTacToe_circulo) {
            jug1->fin(false,false,tablero);
            jug2->fin(true,false,tablero);
            estado=TicTacToe_derrota;
          }
          else {
            if (tablerolleno()) {
              jug1->fin(false,true,tablero);
              jug2->fin(false,true,tablero);
              estado=TicTacToe_tablas;
            }
            else {
              if (!strcmp(IORJug,IORJug1)) {
                jug2->turno(true,tablero);
                jug1->turno(false,tablero);
               IORturno=IORJug2;
              }
              else {
                jug1->turno(true,tablero);
                jug2->turno(false,tablero);
               IORturno=IORJug1;
              }
            }
          }

          return TicTacToe_tablero_dup(tablero);
        }
```

JugadorImpl.h

```
        #ifndef JUGADORIMPL_H
        #define JUGADORIMPL_H

        #include <OB/CORBA.h>
        #include "TicTacToe_skel.h"
        #include "JugadorInter.h"

        class TicTacToe_Jugador_Impl : public TicTacToe_Jugador_skel
        {

        CORBA_Boolean meTocavar;
        TicTacToe_tablero_var tablerovar;
        TicTacToe_Juego_var juegovar;
        TicTacToe_estadoJugador estadovar;
        TicTacToe_ficha usadavar;
        TicTacToe_JugadorInter *jugadorvar;

        public:

        TicTacToe_Jugador_Impl(CORBA_ORB_ptr orb,TicTacToe_JugadorInter *jug);
        void turno(CORBA_Boolean es, const TicTacToe_tablero tab);
        CORBA_Boolean meToca();
        TicTacToe_estadoJugador estado();
        TicTacToe_tablero_slice* tab();
        void   fin(CORBA_Boolean   victoria,   CORBA_Boolean   tablas,   const
TicTacToe_tablero tab);
        void ponerFicha(CORBA_Short x, CORBA_Short y);
```

```
      TicTacToe_ficha usada();

      };

      #endif
```

JugadorImpl.cpp

```
      #include "JugadorImpl.h"
      #include <OB/CosNaming.h>

      TicTacToe_Jugador_Impl::TicTacToe_Jugador_Impl(CORBA_ORB_ptr
orb,TicTacToe_JugadorInter *jug) {
         TicTacToe_ficha fh;
         meTocavar=false;

         CORBA_Object_var obj;

         try
         {
            obj = orb -> resolve_initial_references("NameService");
         }
         catch(const CORBA_ORB::InvalidName&)
         {
            cerr << "No puede resolver 'NameService'" << endl;
            return;
         }

         if(CORBA_is_nil(obj))
         {
            cerr << "'NameService' es una referencia a nil" << endl;
            return;
         }

         CosNaming_NamingContext_var               rootContext               =
CosNaming_NamingContext::_narrow(obj);

         if(CORBA_is_nil(rootContext))
         {
            cerr << "'NameService' no es un servidor de nombres"
              << endl;
            return;
         }

         try
         {
            CosNaming_Name nombre;
            nombre.length(1);
            nombre[0].id = CORBA_string_dup("TicTacToe");
            nombre[0].kind = CORBA_string_dup("");

            CosNaming_NamingContext_var servidores;

            try {
              servidores    =    CosNaming_NamingContext::_narrow(rootContext-
>resolve(nombre));
            } catch (CosNaming_NamingContext::NotFound ex) {
              cerr << "Error: No existe ningun servidor de TicTacToe" << endl;
              return;
            }
            CORBA_Object_var juegoObj;
            CosNaming_BindingList_var lista;
            CosNaming_BindingIterator_var listaiter;
            servidores->list(100,lista,listaiter);

            int i=0;
```

```
            while (i<lista->length()) {
               try {
                  juegoObj=servidores->resolve(lista[i].binding_name);
                  juegovar=TicTacToe_Juego::_narrow(juegoObj);
                  orb->connect(this);
                  tablerovar=juegovar->anadirJugador(this,fh);
                  break;
               } catch (const CORBA_SystemException &) {
                  juegovar=TicTacToe_Juego::_nil();
                  i++;
               }
            }
          if (CORBA_is_nil(juegovar)) {
            cerr << "Error: No existe ningun servidor de TicTacToe" << endl;
            return;
          }

         }
         catch (const CosNaming_NamingContext::NotFound& ex)
         {
            cerr << "Producida la excepcion 'NotFound' (";
            switch(ex.why)
            {
             case CosNaming_NamingContext::missing_node:
                                       cerr << "nodo no encontrado";
                                    break;

             case CosNaming_NamingContext::not_context:
                                       cerr << "no contexto";
                                    break;

             case CosNaming_NamingContext::not_object:
                                       cerr << "no objeto";
                                    break;
            }
            cerr << ")" << endl;
            return;
         }
         catch(const CosNaming_NamingContext::CannotProceed&)
         {
            cerr << "Producida la excepcion 'CannotProceed'" << endl;
            return;
         }
         catch(const CosNaming_NamingContext::InvalidName&)
         {
            cerr << "Producida la excepcion 'InvalidName'" << endl;
            return;
         }
         catch(const CosNaming_NamingContext::AlreadyBound&)
         {
            cerr << "Producida la excepcion 'AlreadyBound'" << endl;
            return;
         }
         catch(const CosNaming_NamingContext::NotEmpty&)
         {
            cerr << "Producida la excepcion 'NotEmpty'" << endl;
            return;
         }

         usadavar=fh;
         estadovar=TicTacToe_jugando;
         jugadorvar=jug;
      }

      void     TicTacToe_Jugador_Impl::turno(CORBA_Boolean     es,     const
TicTacToe_tablero tab) {
          tablerovar=TicTacToe_tablero_dup(tab);
          meTocavar=es;
```

```
        jugadorvar->estadoCambiado(estadovar,meTocavar);
    }

    CORBA_Boolean TicTacToe_Jugador_Impl::meToca() {
        return meTocavar;
    }

    TicTacToe_estadoJugador TicTacToe_Jugador_Impl::estado() {
       return estadovar;
    }

    TicTacToe_tablero_slice* TicTacToe_Jugador_Impl::tab() {
       return TicTacToe_tablero_dup(tablerovar);
    }

    void  TicTacToe_Jugador_Impl::fin(CORBA_Boolean  victoria,  CORBA_Boolean
tablas, const TicTacToe_tablero tab) {
        if (victoria)
          estadovar=TicTacToe_victoria;
        else if (tablas)
          estadovar=TicTacToe_tablas;
        else
          estadovar=TicTacToe_derrota;
        tablerovar=TicTacToe_tablero_dup(tab);
        jugadorvar->estadoCambiado(estadovar,meTocavar);

    }

    void TicTacToe_Jugador_Impl::ponerFicha(CORBA_Short x, CORBA_Short y) {
        try {
          tablerovar=juegovar->ponerFicha(this,x,y);
        } catch (const TicTacToe_JugadorInvalido& ) {
          throw TicTacToe_ErrorEnServidor();
        }
    }

    TicTacToe_ficha TicTacToe_Jugador_Impl::usada() {
       return usadavar;
    }
```

JugadorInter.h

```
    #ifndef JUGADORINTER_H
    #define JUGADORINTER_H

    #include <OB/CORBA.h>
    #include "TicTacToe.h"

    class TicTacToe_JugadorInter {
    public:
       virtual    void    estadoCambiado(TicTacToe_estadoJugador    est,bool
meToca)=0;
    };

    #endif
```

JugadorTextoAplic.h

```
    #ifndef JUGADORTEXTOAPLIC_H
    #define JUGADORTEXTOAPLIC_H

    #include <OB/CORBA.h>
    #include "TicTacToe.h"
    #include "JugadorInter.h"

    class JugadorTextoAplic : public TicTacToe_JugadorInter
    {
    TicTacToe_Jugador_var jugador;
```

```
      void dibujarTablero();
      CORBA_BOA_var boa;

      public:

      JugadorTextoAplic(int narg, char *arg[]);
      void estadoCambiado(TicTacToe_estadoJugador est,bool meToca);
      void jugar();
      };

      #endif
```

JugadorTextoAplic.cpp

```
      #include "JugadorTextoAplic.h"
      #include "JugadorImpl.h"
      #include <string>

      JugadorTextoAplic::JugadorTextoAplic(int narg, char *arg[]) {
            CORBA_ORB_var orb = CORBA_ORB_init(narg,arg);
            boa = orb->BOA_init(narg,arg);
            boa->init_servers();
            jugador = new TicTacToe_Jugador_Impl(orb,this);
      }


      void JugadorTextoAplic::dibujarTablero() {
         TicTacToe_ficha f=jugador->usada();
         TicTacToe_tablero_var tablero = jugador->tab();

         cout << endl;
         for (int i=0;i<3;i++) {
            for (int j=0;j<3;j++)
               {
                  if (j!=0)
                     cout << "|";
                  if (tablero[i][j]==TicTacToe_cruz)
                     cout << " X ";
                  else if (tablero[i][j]==TicTacToe_circulo)
                     cout << " O ";
                  else
                     cout << "   ";
               }
            cout << endl;
         }
      }

      void  JugadorTextoAplic::estadoCambiado(TicTacToe_estadoJugador  est,bool
meToca) {
         dibujarTablero();
         if (est==TicTacToe_jugando) {
            cout << "Jugando" << endl;
            if (meToca) {
               int valorX, valorY;
               cout << "Su turno" << endl;
               while (true) {
                  try {
                     cout << "Coordenada X (1-3): ";
                     cin >> valorX;
                     //if (valorX<1 || valorX>3) continue;
                     cout << "Coordenada Y (1-3): ";
                     cin >> valorY;
                     //if (valorY<1 || valorY>3) continue;
                     jugador->ponerFicha(valorY-1, valorX-1);
                     dibujarTablero();
                     break;
                  } catch (TicTacToe_ErrorEnServidor ex) {
```

```
                  cerr << "El servidor ha fallado" << endl;
               } catch (TicTacToe_CasillaOcupada ex) {
                  cerr << "Casilla Ocupada, o inexistente" << endl;
               }
            }
         }
         else
            cout << "Espere su turno" << endl;
      }
      else if (est==TicTacToe_victoria)
         cout << "Ha vencido" << endl;
      else if (est==TicTacToe_derrota)
         cout << "Ha perdido" << endl;
      else if (est==TicTacToe_tablas)
         cout << "TABLAS" << endl;
   }

   void JugadorTextoAplic::jugar() {
      dibujarTablero();
      cout << "Espere ..." << endl;
      boa->impl_is_ready(CORBA_ImplementationDef::_nil());
      while (true);
   }

   void main(int narg, char *arg[]) {
      JugadorTextoAplic aplic(narg,arg);
      aplic.jugar();
   }
```

JuegoServer.cpp

```
   #include <OB/CORBA.h>
   #include <OB/CosNaming.h>
   #include "TicTacToe.h"
   #include "JuegoImpl.h"
   #include <signal.h>

   CORBA_BOA_var boa;
   void finalizar(int a);

   int main(int narg,char *arg[])
   {

      CORBA_ORB_var orb = CORBA_ORB_init(narg,arg);
      boa = orb->BOA_init(narg,arg);

      // Uso el servicio de nombres
      CORBA_Object_var obj;

      try
      {
         obj = orb -> resolve_initial_references("NameService");
      }
      catch(const CORBA_ORB::InvalidName&)
      {
         cerr << arg[0] << ": no puede resolver 'NameService'" << endl;
         return 1;
      }

      if(CORBA_is_nil(obj))
      {
         cerr << arg[0] << ": 'NameService' es una referencia a nil" <<
endl;
         return 1;
      }
```

```
         CosNaming_NamingContext_var               rootContext               =
CosNaming_NamingContext::_narrow(obj);

         if(CORBA_is_nil(rootContext))
         {
            cerr << arg[0]
              << ": 'NameService' no es un servidor de nombres"
              << endl;
            return 1;
         }

         try
         {
            TicTacToe_Juego_var implObject = new TicTacToe_Juego_Impl(orb);

            orb->connect(implObject);

            CosNaming_Name nombre;
            nombre.length(1);
            nombre[0].id = CORBA_string_dup("TicTacToe");
            nombre[0].kind = CORBA_string_dup("");

            CosNaming_NamingContext_var servidores;

            try {
              servidores    =    CosNaming_NamingContext::_narrow(rootContext-
>resolve(nombre));
            } catch (CosNaming_NamingContext::NotFound ex) {
              servidores = rootContext->bind_new_context(nombre);
            }
            int i=0;
            while (true) {
              try {
                nombre[0].id = CORBA_string_dup("Servidor");
                nombre[0].id += i;
                servidores->bind(nombre,implObject);
                // Si lo consigue asociar termina
                break;
              } catch (CosNaming_NamingContext::AlreadyBound ex) {
                // Mientras no lo pueda asociar porque exista otro
                // servidor continua intentandolo
                i++;
                if (i==100) {
                   cerr << "Error: No se pueden tener mas de 100 servidores"
<< endl;
                   return 1;
                }
              }
            }

            cout << "Esta listo. Como Servidor" << i << endl;
            cout << "Pulse CTRL+Z para finalizar" << endl;
            signal(SIGTSTP,finalizar);

            boa->impl_is_ready(CORBA_ImplementationDef::_nil());

            servidores->unbind(nombre);

         }
         catch (const CosNaming_NamingContext::NotFound& ex)
         {
            cerr << arg[0] << ": Producida la excepcion 'NotFound' (";
            switch(ex.why)
            {
              case CosNaming_NamingContext::missing_node:
                                        cerr << "nodo no encontrado";
                                    break;
```

```
            case CosNaming_NamingContext::not_context:
                                    cerr << "no contexto";
                                 break;

            case CosNaming_NamingContext::not_object:
                                    cerr << "no objeto";
                                 break;
            }
            cerr << ")" << endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::CannotProceed&)
         {
            cerr << arg[0] << ": Producida la excepcion 'CannotProceed'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::InvalidName&)
         {
            cerr << arg[0] << ": Producida la excepcion 'InvalidName'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::AlreadyBound&)
         {
            cerr << arg[0] << ": Producida la excepcion 'AlreadyBound'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::NotEmpty&)
         {
            cerr << arg[0] << ": Producida la excepcion 'NotEmpty'" << endl;
            return 1;
         }

         return 0;
      }

      void finalizar(int a) {
         boa->deactivate_impl(CORBA_ImplementationDef::_nil());
      }
```

Makefile para TicTacToe

```
      CPPFLAGS = -I.

      all: JuegoServer JugadorTextoAplic

      JuegoServer: TicTacToe.o TicTacToe_skel.o JuegoImpl.o JuegoServer.o
            g++  TicTacToe.o  TicTacToe_skel.o  JuegoImpl.o  JuegoServer.o  -o
JuegoServer -lOB -lCosNaming


      JugadorTextoAplic:    JugadorTextoAplic.o    JugadorImpl.o    TicTacToe.o
TicTacToe_skel.o
            g++      JugadorTextoAplic.o      JugadorImpl.o      TicTacToe.o
TicTacToe_skel.o -o JugadorTextoAplic -lOB -lCosNaming
```

Apéndice B. Código fuente del centro comercial virtual

El código para el centro comercial virtual, tanto en Java como en C++ puede encontrarse en el siguente apéndice.

B.1 Centro comercial virtual en Java

creartablas.sql

```
DROP TABLE socio;
DROP TABLE comercio_producto;
DROP TABLE comercio;
DROP TABLE producto;

CREATE TABLE socio (
       idsocio      text CONSTRAINT pk_socio PRIMARY KEY,
       clave text
       );

CREATE TABLE comercio (
       nombre text CONSTRAINT pk_comercio PRIMARY KEY,
       direccion text,
       telefono1 text,
       telefono2 text,
       fax text,
       correoElectronico text,
       direccionWeb text
       );

CREATE TABLE producto (
       idProducto int4 CONSTRAINT pk_producto PRIMARY KEY,
       nombre text,
       precio int4,
       caracteristicas text,
       precioOferta int4
       );

CREATE TABLE comercio_producto (
       nombre_comercio  text  CONSTRAINT  fk_nombre_comercio  REFERENCES
comercio (nombre),
       idProducto  int4  CONSTRAINT  fk_idProducto  REFERENCES  producto
(idProducto),
       CONSTRAINT  pk_comercio_producto  PRIMARY  KEY  (nombre_comercio,
idProducto)
       );
```

insertardatos.sql

```
DELETE FROM socio;
DELETE FROM comercio_producto;
DELETE FROM comercio;
DELETE FROM producto;

INSERT INTO socio VALUES ('Javi', '123');
INSERT INTO socio VALUES ('Mari', '456');

INSERT INTO comercio VALUES ('Prica', 'C/ Perez', '968212363', '', '',
'', 'http://192.168.105.1/cgi-bin/prica.cgi');
INSERT INTO comercio VALUES ('Continente', 'C/ Juan', '968215463', '',
'', '', 'http://192.168.105.1/cgi-bin/continente.cgi');
INSERT  INTO  comercio  VALUES  ('El  Corte  Ingles',  'C/  Gran  Via',
'968215442', '', '', '', 'http://192.168.105.1/cgi-bin/corteingles.cgi');
```

```
      INSERT INTO comercio VALUES ('Mercadona', 'C/ Santa Isabel',
'968217838', '', '', '', 'http://192.168.105.1/cgi-bin/mercadona.cgi');
      INSERT INTO comercio VALUES ('Que', 'C/ Gloria', '968218421', '', '',
'', 'http://192.168.105.1/cgi-bin/que.cgi');

      INSERT INTO producto VALUES (1,'Pila',240,'Alcalinas',225);
      INSERT INTO producto VALUES (2,'Leche',110,'Pascual',105);
      INSERT INTO producto VALUES (3,'Ordenador',200000,'PII 200',180000);
      INSERT INTO producto VALUES (4,'TV',55000,'18 Pulgadas',50000);
      INSERT INTO producto VALUES (5,'Video',34000,'Dos cabezales',32000);
      INSERT INTO producto VALUES (6,'Camiseta',3500,'Azul',3200);
      INSERT INTO producto VALUES (7,'Cinta',300,'Video',270);

      INSERT INTO comercio_producto VALUES ('Prica',1);
      INSERT INTO comercio_producto VALUES ('Prica',3);
      INSERT INTO comercio_producto VALUES ('Prica',7);
      INSERT INTO comercio_producto VALUES ('Continente',1);
      INSERT INTO comercio_producto VALUES ('Continente',2);
      INSERT INTO comercio_producto VALUES ('Continente',4);
      INSERT INTO comercio_producto VALUES ('Continente',5);
      INSERT INTO comercio_producto VALUES ('Continente',6);
      INSERT INTO comercio_producto VALUES ('Continente',7);
```

index.html

```
      <!doctype html public "-//w3c//dtd html 4.0 transitional//en">
      <html>
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
         <meta name="Generator" content="Microsoft Word 97">
         <meta  name="Template"  content="D:\Archivos  de  programa\Microsoft
Office\Office\html.dot">
         <meta   name="GENERATOR"   content="Mozilla/4.5   [en]   (WinNT;   I)
[Netscape]">
         <title>CENTRO VIRTUAL</title>
      </head>
      <body link="#0000FF" vlink="#800080">

      <center><font            color="#FF0000"><font            size=+4>CENTRO
VIRTUAL</font></font></center>

      <p><br>
      <br>
      <br>
      <p><img SRC="Image1.gif" BORDER=0 height=189 width=265>
      <br> 
      <br> 
      <br>
      <center>
      <p><font size=+3>Gracias por visitar nuestro centro comercial</font>
      <p><font size=+2>Podr&aacute; obtener todos los servicios deseados con
      solo hacer click con el rat&oacute;n</font>.
      <br> 
      <br> 
      <p><font    size=+2>A    continuaci&oacute;n    debe    seleccionar    su
situaci&oacute;n:</font>
      <p>  <font                  size=+3>                   <a
href="Empresa.html">Empresa</a></font>
      <p><font size=+3>    <a href="Cliente.html">Cliente del
      centro</a></font></center>

      <p><br>

      </body>
      </html>
```

AltaSocio.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="Author" content="Javi">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Alta de un Comercio</title>
</head>

<body>

<p align="center"><font color="#FF0000" size="7">ALTA SOCIO</font>
</p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p>
<!--"CONVERTED_APPLET"-->
<!-- CONVERTER VERSION 1.0 -->
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = "400" HEIGHT = "375"
codebase="http://java.sun.com/products/plugin/1.2/jinstall-12-
win32.cab#Version=1,2,0,0">
<PARAM NAME = CODE VALUE = "Centro.AltaSocio" >

<PARAM NAME="type" VALUE="application/x-java-applet;version=1.2">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.2" java_CODE =
"Centro.AltaSocio" WIDTH = "400" HEIGHT = "375"
pluginspage="http://java.sun.com/products/plugin/1.2/plugin-
install.html"><NOEMBED></COMMENT>

</NOEMBED></EMBED>
</OBJECT>

<!--
<APPLET  CODE = "Centro.AltaSocio" WIDTH = "400" HEIGHT = "375" >


</APPLET>
-->
<!--"END_CONVERTED_APPLET"-->

</p>


<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<table border="0" width="100%">
    <tr>
        <td><a href="Internauta.html">Atras</a></td>
        <td><a href="index.html">Inicio</a></td>
    </tr>
</table>

<p align="right" dir="ltr"> </p>
</body>
</html>
```

AltaSocioError.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="Author" content="Javi">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Alta de un Comercio</title>
</head>

<body>

<p align="center"><font color="#FF0000" size="7">ERROR EN EL ALTA DEL
SOCIO</font>
</p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<table border="0" width="100%">
    <tr>
        <td><a href="AltaSocio.html">Atras</a></td>
        <td><a href="index.html">Inicio</a></td>
    </tr>
</table>

<p align="right" dir="ltr"> </p>
</body>
</html>
```

AltaSocioRealizada.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="Author" content="Javi">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Alta de un Comercio</title>
</head>

<body>

<p    align="center"><font    color="#FF0000"    size="7">ALTA    SOCIO
REALIZADA</font>
</p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>

<p align="left" dir="ltr"> </p>
```

```
<table border="0" width="100%">
    <tr>
        <td><a href="AltaSocio.html">Atras</a></td>
        <td><a href="index.html">Inicio</a></td>
    </tr>
</table>

<p align="right" dir="ltr"> </p>
</body>
</html>
```

Cliente.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="Author" content="Javi">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Cliente del centro</title>
</head>

<body>

<p align="center"><font color="#FF0000" size="7">CLIENTE DEL
CENTRO</font> </p>

<p> </p>

<p> </p>

<p><font size="5">Bienvenido a nuestro centro comercial,
esperamos que sea de su agrado.</font> <br>
  </p>

<p><font size="5">Por favor, ¿es usted socio del centro virtual?</font>
</p>

<p><font size="5">    </font><a href="Socio.html"><font
size="5">Sí, soy socio</font></a> </p>

<p><font size="5">    </font><a
href="Internauta.html"><font size="5">No, aún no soy socio</font></a>
</p>

<p>  </p>

<p><font size="5">Para cualquier duda o sugerencia no dude en
enviarla a:   </font><a href="mailto:jcrs1@alu.um.es">WebMaster</a>
</p>

<p> </p>

<p align="right"><a href="index.html">Inicio</a></p>
</body>
</html>
```

ErrorSocio.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
```

```
<title></title>
</head>

<body>

<p align="center"><font color="#FF0000" size="7">ERROR SOCIO</font></p>

<p> </p>

<p>El socio que ha introducido no es correcto. (O no existe el
socio o la clave es erronea).</p>

<p> </p>

<p> </p>

<table border="0" width="100%">
    <tr>
        <td><a href="Socio.html">Atras</a></td>
        <td><a href="index.html">Inicio</a></td>
    </tr>
</table>
</body>
</html>
```

InscripcionSocio.html

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
   <meta name="Author" content="Javi">
   <meta   name="GENERATOR"   content="Mozilla/4.5   [en]   (WinNT;   I)
[Netscape]">
   <title>Inscripcion de un Socio</title>
</head>
<body>

<center><font   color="#FF0000"><font   size=+4>INSCRIPCI&Oacute;N   DE
SOCIO</font></font></center>

<p><br>
<br>
</body>
</html>
```

Internauta.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="Author" content="Javi">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Internauta</title>
</head>

<body>

<p align="center"><font color="#FF0000" size="7">INTERNAUTA</font>
</p>

<p><font color="#000000" size="5">Gracias por visitar nuestro
centro comercial.</font> </p>
```

```
<p><font color="#000000" size="5">Si está interesado puede pasar
a formar parte del centro como un nuevo socio, obteniendo de esta
forma las ventajas que esto conlleva (compra sin necesidad de
rellenar formularios, ofertas para socios, catálogos gratuitos,
regalos por volumen de compra, ...)</font> </p>

<p><font color="#000000" size="5">Aunque puede realizar una
compra sin ser socio, o simplemente pasear por nuestro centro
virtual.</font> </p>

<p align="left"><font color="#000000" size="5">¿Qué desea hacer?</font>
</p>

<p align="center"><font size="5">
<a href="AltaSocio.html">Inscripción como socio</a>
</font> </p>

<p align="center"><font size="5">Paseo por el centro comercial</font>
</p>

<p> </p>

<p> </p>

<table border="0" width="100%">
    <tr>
        <td><a href="Cliente.html">Atras</a></td>
        <td><a href="index.html">Inicio</a></td>
    </tr>
</table>
</body>
</html>
```

Prica.html

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title></title>
</head>

<body>

<h1 align="center"><font color="#FF0000" size="7">Prica</font></h1>

<p> </p>

<p><font size="5">Bienvenido al comercio, esperamos que la visita
sea de su agrado.</font></p>

<p><font size="5"></font> </p>

<p><!--"CONVERTED_APPLET"-->
<!-- CONVERTER VERSION 1.0 -->
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = "400" HEIGHT = "300"
codebase="http://java.sun.com/products/plugin/1.2/jinstall-12-
win32.cab#Version=1,2,0,0">
<PARAM NAME = CODE VALUE = "Centro.ProductosComercio" >

<PARAM NAME="type" VALUE="application/x-java-applet;version=1.2">
<PARAM NAME = "COMERCIO" VALUE ="Prica">
<PARAM NAME = "IOR" VALUE ="&lt;!--IOR--&gt;">
```

```
      <COMMENT>
      <EMBED    type="application/x-java-applet;version=1.2"    java_CODE    =
"Centro.ProductosComercio"  WIDTH  =  "400"  HEIGHT  =  "300"   COMERCIO  =  "Prica"
IOR                          =                          "&lt;!--IOR--&gt;"
pluginspage="http://java.sun.com/products/plugin/1.2/plugin-
install.html"><NOEMBED></COMMENT>

      </NOEMBED></EMBED>
      </OBJECT>

      <!--
      <APPLET  CODE = "Centro.ProductosComercio" WIDTH = "400" HEIGHT = "300"
>
      <PARAM NAME = "COMERCIO" VALUE ="Prica">
      <PARAM NAME = "IOR" VALUE ="&lt;!--IOR--&gt;">


      </APPLET>
      -->
      <!--"END_CONVERTED_APPLET"-->
      </p>
      </body>
      </html>
```

Socio.html

```
      <html>

      <head>
      <meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1">
      <meta name="Author" content="Javi">
      <meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
      <title>Socio</title>
      </head>

      <body>

      <p align="center"><font color="#FF0000" size="7">SOCIO</font> </p>

      <p> </p>

      <h5><font size="5">Gracias por su visita, por favor identifíquese
      en el centro:</font></h5>

      <p> </p>

      <p><object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
      codebase="http://java.sun.com/products/plugin/1.2/jinstall-12-
win32.cab#Version=1,2,0,0"
      align="baseline" border="0" width="400" height="300"><param
      name="CODE" value="Centro.IdentificaSocio"><param name="type"
      value="application/x-java-applet;version=1.2"><COMMENT><embed
align="baseline" border="0" width="400" height="300" type="application/x-java-
applet;version=1.2"                    java_code="Centro.IdentificaSocio"
pluginspage="http://java.sun.com/products/plugin/1.2/plugin-
install.html"><noembed></COMMENT> alt="Por favor actualice la versión de su
explorador u obtenga uno compatible con Java" Por favor actualice la versión
de su explorador u obtenga uno compatible con Java </noembed></object> <!--
      <applet code="Centro.IdentificaSocio" width="128" height="128">Por favor
actualice la versión de su explorador u obtenga uno compatible con
Java</applet>
      --> </p>

      <p> </p>

      <p> </p>
```

```
<table border="0" width="100%">
    <tr>
        <td><a href="Cliente.html">Atras</a></td>
        <td><a href="index.html">Inicio</a></td>
    </tr>
</table>
</body>
</html>
```

VisitaCentro.html

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
   <meta name="Author" content="Javi">
   <meta   name="GENERATOR"   content="Mozilla/4.5   [en]   (WinNT;   I)
[Netscape]">
   <title>Centro Comercial</title>
</head>
<body>

<center><font            color="#FF0000"><font            size=+4>CENTRO
COMERCIAL</font></font></center>

<p><br>
<br>
</body>
</html>
```

CarroCompra.cgi

```
#!/usr/bin/perl

open (BASEHTML,"CarroCompra.html");
$tmp=$/;
undef $/;
$docu = <BASEHTML>;
close (BASEHTML);

$entrada=$ENV{'QUERY_STRING'};
$docu =~ s/<!--IOR-->/$entrada/g;
print "Content-type: text/html\n\n";
print $docu;
$/=$tmp;
```

Continente.cgi

```
#!/usr/bin/perl

open (BASEHTML,"Continente.html");
$tmp=$/;
undef $/;
$docu = <BASEHTML>;
close (BASEHTML);

$entrada=$ENV{'QUERY_STRING'};
$docu =~ s/<!--IOR-->/$entrada/g;
print "Content-type: text/html\n\n";
print $docu;
$/=$tmp;
```

Prica.cgi

```
#!/usr/bin/perl

open (BASEHTML,"Prica.html");
$tmp=$/;
undef $/;
$docu = <BASEHTML>;
close (BASEHTML);

$entrada=$ENV{'QUERY_STRING'};
$docu =~ s/<!--IOR-->/$entrada/g;
print "Content-type: text/html\n\n";
print $docu;
$/=$tmp;
```

SeleccionCentro.cgi

```
#!/usr/bin/perl

open (BASEHTML,"SeleccionCentro.html");
$tmp=$/;
undef $/;
$docu = <BASEHTML>;
close (BASEHTML);

$entrada=$ENV{'QUERY_STRING'};
$docu =~ s/<!--IOR-->/$entrada/g;
print "Content-type: text/html\n\n";
print $docu;
$/=$tmp;
```

CentroComercial.idl

```
module Centro {
	interface Internauta {
		attribute string nombre, apellido1, apellido2, telefono,
correoElectronico, numeroCuenta, direccion;
	};

	interface Socio : Internauta {
		attribute string idSocio, clave;
	};

	interface PeriodoAvisos {
		attribute string tiempo;
		attribute boolean porPedido;
	};

	interface Producto {
		attribute string nombre, caracteristicas;
		attribute long long precio, precioOferta, idProducto;
	};

	interface Comercio {
		typedef sequence<Producto> ListaProductos;

		attribute string nombre, direccion, telefono1, telefono2,
fax, correoElectronico, direccionWeb;
		attribute PeriodoAvisos refPeriodoAvisos;
		attribute ListaProductos refProducto;
	};

	struct ProductosCantidad {
		Producto prod;
```

```
                long cantidad;
                string nombreComercio;
        };
        typedef sequence<ProductosCantidad> ListaProductosCantidad;

        interface CarroCompra {
                typedef sequence<long> ListaCantidad;
                typedef sequence<string> ListaComercios;

                attribute Internauta refInternauta;
                attribute ListaProductosCantidad refProducto;

                void anadirProducto(in Producto prod, in long cantidad, in
string nombreComercio);
                void   listarProductos(out   ListaProductos   prod,   out
ListaCantidad cantidad, out ListaComercios comercios);
                void cambiarCantidad(in Producto prod, in string comercio,
in long cantidad);
        };

        interface FactoriaSocio {
                CarroCompra validarSocio(in string id, in string clave);
                long  altaSocio(in  string  idSocio,  in  string  clave,  in
string nombre, in string apellido1, in string apellido2, in string telefono,
in string correoElectronico, in string numeroCuenta, in string direccion);
        };

        interface FactoriaComercio {
                typedef sequence<string> ListaComercios;

                ListaComercios listarComercios();
                string direccionWeb(in string nombre);
                ListaProductos listarProductos(in string Comercio);
        };

        interface FactoriaFactura {
                void nuevaFactura(in CarroCompra carro);
        };
};
```

CarroCompraImpl.java

```
package Centro;

import java.util.*;
import Centro.CarroCompraPackage.*;

public class CarroCompraImpl extends _CarroCompraImplBase {
        Internauta refInternauta;
        ArrayList refProducto = new ArrayList();

        public Internauta refInternauta() {
                return refInternauta;
        }

        public void refInternauta(Internauta arg) {
                refInternauta = arg;
        }

        public ProductosCantidad[] refProducto(){
                return (ProductosCantidad[]) refProducto.toArray();
        }

        public void refProducto(ProductosCantidad[] arg) {
                refProducto.addAll(Arrays.asList(arg));
        }
```

```
            public void anadirProducto(Centro.Producto prod, int cantidad,
String nombreComercio) {
                  // Al añadir un producto mantendré la lista ordenada
alfabéticamente por
                  // nombre de producto
                  if (cantidad<0) return;          // No trato cantidades
menor que 0
                  for (int i=0; i<refProducto.size(); i++) {
                        ProductosCantidad act = (ProductosCantidad)
refProducto.get(i);
                        if (act.prod.nombre().compareTo(prod.nombre())<0)
continue;
                        if (act.prod.nombre().compareTo(prod.nombre())>0) {
                              ProductosCantidad nuevo = new
ProductosCantidad(prod,cantidad,nombreComercio);
                              refProducto.add(i,nuevo);
                              return;
                        }
                        if (act.prod.idProducto()==prod.idProducto() &&
act.nombreComercio.equals(nombreComercio)) {
                              // Si es el mismo producto del mismo comercio
sólo aumento la
                              // cantidad de ese producto
                              act.cantidad+=cantidad;
                              return;
                        }
                  }
                  // Si no hay ningún producto aún lo añado al principio
                  ProductosCantidad nuevo = new
ProductosCantidad(prod,cantidad,nombreComercio);
                  refProducto.add(nuevo);

            }

            public void listarProductos(ListaProductosHolder prod,
ListaCantidadHolder cantidad, ListaComerciosHolder comercios) {
                  prod.value = new Producto[refProducto.size()];
                  cantidad.value = new int[refProducto.size()];
                  comercios.value = new String[refProducto.size()];

                  for (int i=0; i<refProducto.size(); i++) {
                        ProductosCantidad act = (ProductosCantidad)
refProducto.get(i);
                        prod.value[i] = act.prod;
                        cantidad.value[i] = act.cantidad;
                        comercios.value[i] = act.nombreComercio;
                  }
            }

            public void cambiarCantidad(Producto prod, String comercio, int
cantidad) {
                  if (cantidad<0) return;          // No trato cantidades
menor que 0

                  for (int i=0; i<refProducto.size(); i++) {
                        ProductosCantidad act = (ProductosCantidad)
refProducto.get(i);
                        if (act.prod.idProducto()==prod.idProducto() &&
act.nombreComercio.equals(comercio)) {
                              // Si es el mismo producto del mismo comercio
cambio la cantidad
                              // Si la cantidad es 0 quito el producto
                              if (cantidad==0) {
                                    refProducto.remove(i);
                                    return;
                              }
                              act.cantidad=cantidad;
                              return;
```

```
                        }
                }
        }
}
```

FactoriaComercioImpl.java

```
package Centro;

import java.sql.*;
import java.util.*;

public class FactoriaComercioImpl extends _FactoriaComercioImplBase {
        public String[] listarComercios() {
                try {
                        // Comienzo realizando la consulta de los comercios
existentes
                        ResultSet                resultado                =
ConstantesServidor.consultas.executeQuery(
                                "SELECT nombre FROM comercio;");

                        // Consulta realizada

                        Vector comercios = new Vector();

                        //  A  continuación  tomo  el  nombre  de  todos  los
comercios
                        while (resultado.next())
                                comercios.add(resultado.getString("nombre"));

                        Object aux[] = comercios.toArray();

                        String aCom[] = new String[aux.length];

                        for (int i=0;i<aux.length;i++)
                                aCom[i]=(String)aux[i];

                        // Devuelvo los comercios encontrados
                        return aCom;
                } catch (SQLException ex) {
                        ex.printStackTrace();
                }
                return null;
        }

        public String direccionWeb(String nombre) {
                try {
                        //  Comienzo  realizando  la  consulta  del  comercio
pedido
                        ResultSet                resultado                =
ConstantesServidor.consultas.executeQuery(
                                "SELECT  direccionWeb  FROM  comercio  WHERE
nombre='" + nombre + "';");

                        // Consulta realizada

                        if (!resultado.next()) return null;

                        // Devuelvo la dirección correspondiente


                        return resultado.getString("direccionWeb");

                } catch (SQLException ex) {
                        ex.printStackTrace();
                }
                return null;
```

```
		}

		public Producto[] listarProductos(String Comercio) {
			Producto listaProductos[] = {};

			try {
				// Comienzo realizando la consulta de los productos de este comercio
				ResultSet resultado = ConstantesServidor.consultas.executeQuery(
					"SELECT * FROM Producto WHERE idproducto IN (SELECT" +
					" idproducto FROM comercio_producto WHERE nombre_comercio = '" +
					Comercio + "');");

				// Consulta realizada

				Vector productos = new Vector();

				// A continuación tomo el nombre de todos los comercios
				while (resultado.next()) {
					// Hay que comprobar si ya existe el producto en el Servidor
					// de nombre, si existe tomo el existente, si no existe lo
					// creo y lo asocio en el servidor de nombres

					ProductoImpl nuevoProducto = new ProductoImpl(resultado.getLong("idproducto"),
						resultado.getString("nombre"), resultado.getLong("precio"),
						resultado.getString("caracteristicas"), resultado.getLong("preciooferta"));

					productos.add(nuevoProducto);
				}

				Object aux[] = productos.toArray();

				listaProductos = new Producto[aux.length];

				for (int i=0;i<aux.length;i++)
					listaProductos[i]=(Producto)aux[i];

				// Devuelvo los comercios encontrados
				return listaProductos;
			} catch (SQLException ex) {
				ex.printStackTrace();
			}


			return listaProductos;
		}
	}
```

FactoriaSocioImpl.java

```
package Centro;

import java.sql.*;
import org.omg.CORBA.*;

public class FactoriaSocioImpl extends _FactoriaSocioImplBase {

	public CarroCompra validarSocio(String id, String clave) {
```

```
                    try {
                          // Empieza la validación
                          ResultSet                resultado                =
ConstantesServidor.consultas.executeQuery(
                                "SELECT clave FROM socio WHERE idSocio='" +
id + "';");

                          // Consulta realizada

                          // Lo primero compruebo que hay alguna fila en la
tabla correspondiente
                          // a este socio -> que esté dado de alta
                          if (! resultado.next()) {
                                // No existe tal socio
                                System.out.println("No ha devuelto ninguna
fila");
                                return null;
                          }

                          // Si el socio existe compruebo que la clave es
correcta
                          String               claveVerdadera               =
resultado.getString("clave");
                          if (!clave.equals(claveVerdadera)) {
                                System.out.println("Clave            erronea,
introducida: " + clave + " debería ser: " + claveVerdadera);
                                // La clave no es correcta
                                return null;
                          }

                          // Tras la validación creo un carro de la compra
para este socio
                          CarroCompraImpl carro = new CarroCompraImpl();
                          String args[] = {};
                          ORB orb = ORB.init(args, null);
                          orb.connect(carro);

                          // Creo el socio que estará asociado al carro
                          SocioImpl socio = new SocioImpl();
                          socio.idSocio(id);
                          socio.clave(clave);

                          // Asocio el socio al carro
                          carro.refInternauta(socio);

                          // Devuelvo el carro de la compra
                          return carro;
                    } catch (SQLException ex) {
                          ex.printStackTrace();
                    }
                    return null;
              }

              public int altaSocio(String idSocio, String clave, String nombre,
String apellido1, String apellido2, String telefono, String correoElectronico,
String numeroCuenta, String direccion) {
                    try {
                          return ConstantesServidor.consultas.executeUpdate(
                                "INSERT INTO Socio VALUES ('" + idSocio +
"','" + clave + "','" + nombre + "','" + apellido1 + "','" + apellido2 + "','"
+ telefono + "','" + correoElectronico + "','" + numeroCuenta + "','" +
direccion + "');");
                    } catch (Exception ex) {
                          ex.printStackTrace();
                          return 0;
                    }

              }
```

```
	}
```

InternautaImpl.java

```
	package Centro;

	public class InternautaImpl extends _InternautaImplBase {
		String nombre, apellido1, apellido2, telefono, correoElectronico,
numeroCuenta, direccion;

		public String nombre() {
			return nombre;
		}
		public void nombre(String arg) {
			nombre=arg;
		}
		public String apellido1() {
			return apellido1;
		}
		public void apellido1(String arg) {
			apellido1=arg;
		}
		public String apellido2() {
			return apellido2;
		}
		public void apellido2(String arg) {
			apellido2=arg;
		}
		public String telefono() {
			return telefono;
		}
		public void telefono(String arg) {
			telefono=arg;
		}
		public String correoElectronico() {
			return correoElectronico;
		}
		public void correoElectronico(String arg) {
			correoElectronico=arg;
		}
		public String numeroCuenta() {
			return numeroCuenta;
		}
		public void numeroCuenta(String arg) {
			numeroCuenta=arg;
		}
		public String direccion() {
			return direccion;
		}
		public void direccion(String arg) {
			direccion = arg;
		}
	}
```

ProductoImpl.java

```
	package Centro;

	public class ProductoImpl extends _ProductoImplBase {
		String nombre, caracteristicas;
		long precio, precioOferta, idProducto;

		public ProductoImpl(long idProducto, String nombre, long precio,
String caracteristicas, long precioOferta) {
			this.idProducto = idProducto;
```

```
                this.nombre = nombre;
                this.precio = precio;
                this.caracteristicas = caracteristicas;
                this.precioOferta = precioOferta;
        }

        public String nombre() {
                return nombre;
        }
        public void nombre(String arg) {
                nombre = arg;
        }
        public String caracteristicas() {
                return caracteristicas;
        }
        public void caracteristicas(String arg) {
                caracteristicas = arg;
        }
        public long precio() {
                return precio;
        }
        public void precio(long arg) {
                precio = arg;
        }
        public long precioOferta() {
                return precioOferta;
        }
        public void precioOferta(long arg) {
                precioOferta = arg;
        }
        public long idProducto() {
                return idProducto;
        }
        public void idProducto(long arg) {
                idProducto = arg;
        }
}
```

SocioImpl.java

```
package Centro;

public class SocioImpl extends _SocioImplBase {

        // Parte que viene del Internauta
        String nombre, apellido1, apellido2, telefono, correoElectronico,
numeroCuenta, direccion;

        public String nombre() {
                return nombre;
        }
        public void nombre(String arg) {
                nombre=arg;
        }
        public String apellido1() {
                return apellido1;
        }
        public void apellido1(String arg) {
                apellido1=arg;
        }
        public String apellido2() {
                return apellido2;
        }
        public void apellido2(String arg) {
                apellido2=arg;
        }
        public String telefono() {
```

```
		return telefono;
	}
	public void telefono(String arg) {
		telefono=arg;
	}
	public String correoElectronico() {
		return correoElectronico;
	}
	public void correoElectronico(String arg) {
		correoElectronico=arg;
	}
	public String numeroCuenta() {
		return numeroCuenta;
	}
	public void numeroCuenta(String arg) {
		numeroCuenta=arg;
	}
	public String direccion() {
		return direccion;
	}
	public void direccion(String arg) {
		direccion = arg;
	}


	// Parte del Socio
	String idSocio, clave;

	public String idSocio() {
		return idSocio;
	}
	public void idSocio(String arg) {
		idSocio=arg;
	}
	public String clave() {
		return clave;
	}
	public void clave(String arg) {
		clave=arg;
	}

}
```

AltaSocio.java

```
package Centro;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//import com.sun.java.swing.*;
import javax.swing.*;
import org.omg.CosNaming.*;
import java.net.*;

//import com.sun.java.swing.UIManager;
public class AltaSocio extends JApplet {
  boolean isStandalone = false;
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JTextField txtNombre = new JTextField();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  JTextField txtIdentificador = new JTextField();
  JLabel jLabel5 = new JLabel();
  JTextField txtApellido1 = new JTextField();
  JLabel jLabel6 = new JLabel();
```

```
        JTextField txtDireccion = new JTextField();
        JLabel jLabel7 = new JLabel();
        JTextField txtEmail = new JTextField();
        JLabel jLabel8 = new JLabel();
        JTextField txtTelefono = new JTextField();
        JLabel jLabel9 = new JLabel();
        JTextField txtApellido2 = new JTextField();
        JLabel jLabel10 = new JLabel();
        JTextField txtNumeroCuenta = new JTextField();
        JButton btnAceptar = new JButton();
        JButton btnCancelar = new JButton();
        JPasswordField txtClave = new JPasswordField();
      //Get a parameter value

        public String getParameter(String key, String def) {
          return isStandalone ? System.getProperty(key, def) :
            (getParameter(key) != null ? getParameter(key) : def);
        }

        //Construct the applet

        public AltaSocio() {
        }
      //Initialize the applet

        public void init() {
          try {
          jbInit();
          }
          catch (Exception e) {
          e.printStackTrace();
          }
        }
        //static {
        //  try {
        //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
        //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
        //                                          UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
        //  }
        //  catch (Exception e) {}
        //}
      //Component initialization

        private void jbInit() throws Exception {
          jLabel1.setText("Introduzca los datos del nuevo socio:");
          jLabel1.setBounds(new Rectangle(44, 3, 315, 25));
          jLabel2.setText("Identificador:");
          jLabel2.setBounds(new Rectangle(14, 37, 96, 24));
          txtNombre.setBounds(new Rectangle(123, 95, 217, 26));
          jLabel3.setBounds(new Rectangle(14, 94, 96, 24));
          jLabel4.setBounds(new Rectangle(14, 66, 96, 24));
          txtIdentificador.setBounds(new Rectangle(123, 39, 217, 26));
          jLabel5.setBounds(new Rectangle(14, 123, 96, 24));
          txtApellido1.setBounds(new Rectangle(123, 123, 217, 26));
          jLabel6.setBounds(new Rectangle(14, 151, 107, 24));
          txtDireccion.setBounds(new Rectangle(123, 235, 217, 26));
          jLabel7.setBounds(new Rectangle(14, 180, 96, 24));
          txtEmail.setBounds(new Rectangle(123, 207, 217, 26));
          jLabel8.setBounds(new Rectangle(14, 208, 96, 24));
          txtTelefono.setBounds(new Rectangle(123, 179, 217, 26));
          jLabel9.setBounds(new Rectangle(14, 237, 96, 24));
          txtApellido2.setBounds(new Rectangle(123, 151, 217, 26));
          jLabel10.setBounds(new Rectangle(14, 265, 96, 24));
          txtNumeroCuenta.setBounds(new Rectangle(123, 263, 217, 26));
          txtClave.setBounds(new Rectangle(123, 67, 217, 26));
```

```
          btnAceptar.setText("Aceptar");
          btnAceptar.setBounds(new Rectangle(38, 306, 88, 23));
          btnAceptar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              btnAceptar_actionPerformed(e);
            }
          });
          btnCancelar.setBounds(new Rectangle(167, 306, 88, 23));
          btnCancelar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              btnCancelar_actionPerformed(e);
            }
          });
          btnCancelar.setText("Cancelar");
          jLabel10.setText("Número Cuenta:");
          jLabel9.setText("Dirección:");
          jLabel8.setText("E-Mail:");
          jLabel7.setText("Telefono:");
          jLabel6.setText("Segundo Apellido:");
          jLabel5.setText("Primer Apellido:");
          jLabel4.setText("Clave:");
          jLabel3.setText("Nombre:");
          this.getContentPane().setLayout(null);
          this.setSize(400,375);
          this.getContentPane().add(jLabel1, null);
          this.getContentPane().add(jLabel2, null);
          this.getContentPane().add(jLabel4, null);
          this.getContentPane().add(txtIdentificador, null);
          this.getContentPane().add(jLabel3, null);
          this.getContentPane().add(txtNombre, null);
          this.getContentPane().add(jLabel5, null);
          this.getContentPane().add(txtApellido1, null);
          this.getContentPane().add(jLabel6, null);
          this.getContentPane().add(jLabel7, null);
          this.getContentPane().add(jLabel8, null);
          this.getContentPane().add(jLabel9, null);
          this.getContentPane().add(txtApellido2, null);
          this.getContentPane().add(txtTelefono, null);
          this.getContentPane().add(txtEmail, null);
          this.getContentPane().add(txtDireccion, null);
          this.getContentPane().add(jLabel10, null);
          this.getContentPane().add(txtNumeroCuenta, null);
          this.getContentPane().add(btnCancelar, null);
          this.getContentPane().add(btnAceptar, null);
          this.getContentPane().add(txtClave, null);
        }
      //Get Applet information

        public String getAppletInfo() {
          return "Applet Information";
        }
      //Get parameter info

        public String[][] getParameterInfo() {
          return null;
        }

        void btnAceptar_actionPerformed(ActionEvent e) {

            FactoriaSocio factoria=null;

            try {
                  NameComponent factoriaName[] = new NameComponent[1];
                  factoriaName[0]=new NameComponent("FactoriaSocio","");
                  org.omg.CORBA.Object             factoriaObj             =
ConstantesCliente.obtenerNamingContextCentro(this).resolve(factoriaName);
                  factoria = FactoriaSocioHelper.narrow(factoriaObj);
            } catch (org.omg.CORBA.UserException ex) {
```

```
                    System.err.println("Excepción  no  esperada,  finalización
anormal");
                    ex.printStackTrace();
              }

              try {
                    if       (factoria.altaSocio(txtIdentificador.getText(),new
String(txtClave.getPassword()),txtNombre.getText(),
                          txtApellido1.getText(),      txtApellido2.getText(),
txtTelefono.getText(), txtEmail.getText(),
                          txtNumeroCuenta.getText(),
txtDireccion.getText())>0)

                          getAppletContext().showDocument(new
URL(getCodeBase().toString()+"AltaSocioRealizada.html"));
                    else
                          getAppletContext().showDocument(new
URL(getCodeBase().toString()+"AltaSocioError.html"));
              } catch (MalformedURLException ex) {
                    ex.printStackTrace();
              }
        }

        void btnCancelar_actionPerformed(ActionEvent e) {
              try {
                    getAppletContext().showDocument(new
URL(getCodeBase().toString()+"Internauta.html"));
              } catch (MalformedURLException ex) {
                    ex.printStackTrace();
              }


        }
      }
```

ComparadorProductos.java

```
      package Centro;

      import java.util.*;

      // Esta clase la utilizo para comparar productos y de esa forma
      // poder mostarlos de forma ordenada

      public class ComparadorProductos implements Comparator {
            public int compare(Object o1, Object o2) {
                  Producto p1,p2;
                  p1=(Producto)o1;
                  p2=(Producto)o2;
                  return p1.nombre().compareTo(p2.nombre());
            }
            public boolean equals(Object obj) {
                  return false;
            }
      }
```

ConstantesCliente.java

```
      package Centro;

      import org.omg.CORBA.*;
      import org.omg.CosNaming.*;

      public class ConstantesCliente {
            static                                              NamingContext
obtenerNamingContextCentro(java.applet.Applet ap) {
```

```
                    // Creo un nuevo ORB

                    //ORB orb = ORB.init(ap, null);
                    String args[] = {};
                    ORB orb= ORB.init(args,null);

                    // En primer lugar obtengo el contexto raíz
                    NamingContext rootContext=null;
                    try {
                          org.omg.CORBA.Object objRef =

      orb.resolve_initial_references("NameService");
                          rootContext = NamingContextHelper.narrow(objRef);
                    } catch (org.omg.CORBA.ORBPackage.InvalidName ex) {
                          System.err.println("Error,   no   se   encuentra   el
servicio de nombres");
                    }

             NamingContext centroNamingContext=null;

             try {

                    //  A  continuación  me  voy  a  meter  por  la  rama  del
CentroVirtual
                    NameComponent centroContext[] = new NameComponent[1];

                    centroContext[0] = new NameComponent("CentroVirtual", "");

                    try {
                          // Pruebo a ver si existe
                          centroNamingContext                                  =
NamingContextHelper.narrow(rootContext.resolve(centroContext));
                    }  catch  (org.omg.CosNaming.NamingContextPackage.NotFound
ex) {
                          // Si no existe error
                          System.err.println("Error,  no  hay  ningún  Centro
Virtual funcionando");
                    }


             } catch (org.omg.CORBA.UserException ex) {
                    System.err.println("Excepción  no  esperada,  finalización
anormal");
                    ex.printStackTrace();
             }
             return centroNamingContext;

             }
      }
```

ConstantesServidor.java

```
package Centro;

import java.sql.*;
import org.omg.CosNaming.*;

public class ConstantesServidor {
       static final String servidor = "192.168.105.1";
       static final String nombreBaseDatos = "centrovirtual";
       static final String usuario = "jros";
       static final String claveUsuario = "";
       static Connection baseDatos;
       static Statement consultas;
       static NamingContext productosNaming;

       static {
```

```
                try {
                        Class.forName("postgresql.Driver");
                        baseDatos = DriverManager.getConnection(
                                "jdbc:postgresql://" + servidor + "/" +
                                nombreBaseDatos, usuario, claveUsuario);
                        consultas = baseDatos.createStatement();
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
        }
}
```

IdentificaSocio.java

```
package Centro;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import com.sun.java.swing.*;
import java.net.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

//import com.sun.java.swing.UIManager;
public class IdentificaSocio extends JApplet {
  boolean isStandalone = false;
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JTextField jTextField1 = new JTextField();
  JPasswordField jPasswordField1 = new JPasswordField();
  JButton jButton1 = new JButton();
//Get a parameter value

  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }

  //Construct the applet

  public IdentificaSocio() {
  }
//Initialize the applet

  public void init() {
    try {
    jbInit();
    }
    catch (Exception e) {
    e.printStackTrace();
    }
  }
  //static {
  //  try {
  //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
  //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
  //                                         UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
  //  }
  //  catch (Exception e) {}
  //}
//Component initialization

  private void jbInit() throws Exception {
```

```
          jLabel1.setText("Identificador de socio:");
          jLabel1.setBounds(new Rectangle(56, 33, 132, 26));
          jLabel2.setText("Clave:");
          jLabel2.setBounds(new Rectangle(56, 103, 61, 28));
          jTextField1.setBounds(new Rectangle(86, 63, 161, 27));
          jPasswordField1.setBounds(new Rectangle(86, 145, 161, 27));
          jButton1.setText("Aceptar");
          jButton1.setBounds(new Rectangle(123, 212, 101, 33));
          jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton1_actionPerformed(e);
            }
          });
          this.getContentPane().setLayout(null);
          this.setSize(400,300);
          this.getContentPane().add(jLabel1, null);
          this.getContentPane().add(jLabel2, null);
          this.getContentPane().add(jTextField1, null);
          this.getContentPane().add(jPasswordField1, null);
          this.getContentPane().add(jButton1, null);
        }
      //Get Applet information

        public String getAppletInfo() {
          return "Applet Information";
        }
      //Get parameter info

        public String[][] getParameterInfo() {
          return null;
        }

        void jButton1_actionPerformed(ActionEvent e) {
          //Codigo para el botón Aceptar

             // Se debe comprobar la validez del socio, y en caso de error
             // indicarlo.

             // Si el socio es válido pasaremos a utilizar CGI para llevar
             // el socio y el carro de la compra actuales

             // En primer lugar se creará un nuevo objeto (CarroCompra) en el
servidor
             // que llevará el socio (o internauta en su caso) y los productos
             // comprados. El OID del objeto se pasará de página en página
             //            utilizando           un           URL           :
http://servidor/pagina.html?datoscomoenCGI
             // El CGI en perl creará una nueva página donde el applet tenga
             // como parámetro el OID del objeto (CarroCompra).

             FactoriaSocio factoria=null;

             try {
                    NameComponent factoriaName[] = new NameComponent[1];
                    factoriaName[0]=new NameComponent("FactoriaSocio","");
                    org.omg.CORBA.Object             factoriaObj             =
ConstantesCliente.obtenerNamingContextCentro(this).resolve(factoriaName);
                    factoria = FactoriaSocioHelper.narrow(factoriaObj);
             } catch (org.omg.CORBA.UserException ex) {
                    System.err.println("Excepción  no  esperada,  finalización
anormal");
                    ex.printStackTrace();
             }

             CarroCompra                        carro                        =
factoria.validarSocio(jTextField1.getText(),String.valueOf(jPasswordField1.get
Password()));
```

```
            try {
                  if (carro==null)
                        getAppletContext().showDocument(new
URL(getCodeBase().toString()+"ErrorSocio.html"));
                  else {
                        String args[] = {};
                        ORB orb = ORB.init(args, null);

                        getAppletContext().showDocument(new  URL("http://"  +
getCodeBase().getHost()+"/cgi-bin/SeleccionCentro.cgi?"                        +
orb.object_to_string(carro)));

                  }
            } catch (MalformedURLException ex) {
                  ex.printStackTrace();
            }
        }

      }
```

JcarroCompra.java

```
      package Centro;

      import java.awt.*;
      import java.awt.event.*;
      import java.applet.*;
      //import com.sun.java.swing.*;
      import javax.swing.*;
      import javax.swing.table.*;
      import org.omg.CosNaming.*;
      import org.omg.CORBA.ORB;
      import java.net.*;
      import Centro.CarroCompraPackage.*;


      //import com.sun.java.swing.UIManager;
      public class JCarroCompra extends JApplet {
        boolean isStandalone = false;
        String IOR;
      //Get a parameter value
        JTable tablaProductos = new JTable();
        JLabel jLabel1 = new JLabel();
            FactoriaComercio factoria=null;
            Producto productos[];
            int cantidades[];
            int cantidadesAnteriores[];
            String comercios[];
        JButton jButton1 = new JButton();
        JButton jButton2 = new JButton();
        JTextField textoBuscar = new JTextField();
        JButton jButton3 = new JButton();

        public String getParameter(String key, String def) {
          return isStandalone ? System.getProperty(key, def) :
            (getParameter(key) != null ? getParameter(key) : def);
        }

        //Construct the applet

        public JCarroCompra() {
        }
      //Initialize the applet

        public void init() {
          try { IOR = this.getParameter("IOR", ""); } catch (Exception e) {
e.printStackTrace(); }
```

```
          try {
          jbInit();
          }
          catch (Exception e) {
          e.printStackTrace();
          }
        }
        //static {
        //  try {
        //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
        //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
        //                                         UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
        //  }
        //  catch (Exception e) {}
        //}
      //Component initialization

        private void jbInit() throws Exception {
          this.getContentPane().setLayout(null);
          this.setSize(650,300);
          tablaProductos.setBounds(new Rectangle(34, 77, 456, 156));

            String args[] = {};
            ORB orb= ORB.init(args,null);

            org.omg.CORBA.Object objCarro = orb.string_to_object(IOR);
            CarroCompra carro = CarroCompraHelper.narrow(objCarro);

            ListaProductosHolder productosH = new ListaProductosHolder();
            ListaCantidadHolder cantidadesH = new ListaCantidadHolder();
            ListaComerciosHolder comerciosH = new ListaComerciosHolder();

            carro.listarProductos(productosH,cantidadesH,comerciosH);

            productos = productosH.value;
            cantidades = cantidadesH.value;
            comercios = comerciosH.value;

            cantidadesAnteriores = (int[]) cantidades.clone();

            ContenidoTablaProductos     contenidoTablaProductos     =     new
ContenidoTablaProductos();
            tablaProductos.setModel(contenidoTablaProductos);
          jLabel1.setText("Este es su carro actual:");
          jLabel1.setBounds(new Rectangle(25, 21, 228, 30));
          jButton1.setText("Aceptar");
          jButton1.setBounds(new Rectangle(77, 262, 81, 23));
          jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton1_actionPerformed(e);
            }
          });
          jButton2.setText("Cancelar");
          jButton2.setBounds(new Rectangle(210, 261, 93, 23));
          jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton2_actionPerformed(e);
            }
          });
          textoBuscar.setText("");
          textoBuscar.setBounds(new Rectangle(500, 84, 100, 22));
          jButton3.setText("Buscar");
          jButton3.setBounds(new Rectangle(509, 129, 81, 23));
          jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
              jButton3_actionPerformed(e);
            }
          });
          this.getContentPane().add(tablaProductos, null);
          this.getContentPane().add(jLabel1, null);
          this.getContentPane().add(jButton1, null);
          this.getContentPane().add(jButton2, null);
          this.getContentPane().add(textoBuscar, null);
          this.getContentPane().add(jButton3, null);

        }
      //Get Applet information

        public String getAppletInfo() {
          return "Applet Information";
        }
      //Get parameter info

        public String[][] getParameterInfo() {
          String pinfo[][] =
          {
            {"IOR", "String", ""},
          };
          return pinfo;
        }

        void jButton3_actionPerformed(ActionEvent e) {
          // Boton buscar
             for (int i=0; i<productos.length; i++) {
                    if
(productos[i].nombre().toUpperCase().equals(textoBuscar.getText().toUpperCase(
))) {
                           tablaProductos.setRowSelectionInterval(i+1,i+1);
                           break;
                    }
             }
        }

        void jButton1_actionPerformed(ActionEvent e) {
          // Boton Aceptar

             tablaProductos.editCellAt(0,4);

             String args[] = {};
             ORB orb= ORB.init(args,null);

             org.omg.CORBA.Object objCarro = orb.string_to_object(IOR);
             CarroCompra carro = CarroCompraHelper.narrow(objCarro);

             for (int i=0; i<cantidades.length; i++) {
                    if (cantidades[i]!=cantidadesAnteriores[i]);

      carro.cambiarCantidad(productos[i],comercios[i],cantidades[i]);
             }

             try {
                    getAppletContext().showDocument(new     URL("http://"     +
getCodeBase().getHost()+"/cgi-bin/SeleccionCentro.cgi?" + IOR));
             } catch (MalformedURLException ex) {}
        }

        void jButton2_actionPerformed(ActionEvent e) {
          // Boton Cancelar

             try {
                    getAppletContext().showDocument(new     URL("http://"     +
getCodeBase().getHost()+"/cgi-bin/SeleccionCentro.cgi?" + IOR));
             } catch (MalformedURLException ex) {}
```

```
	}

		class ContenidoTablaProductos extends AbstractTableModel {
			public int getRowCount() {
				return productos.length + 1;
			}

			public int getColumnCount() {
				return 6;
			}

			public Object getValueAt(int row, int column) {
				// Listar los elementos de la variable productos
				if (row==0) {
					switch (column) {
						case 0: return "Nombre";
						case 1: return "Precio";
						case 2: return "Características";
						case 3: return "Precio Socios";
						case 4: return "Cantidad";
						case 5: return "Comercio";
					}
				}
				else {
					switch (column) {
						case    0:    return    productos[row-
1].nombre();
						case             1:             return
String.valueOf(productos[row-1].precio());
						case    2:    return    productos[row-
1].caracteristicas();
						case             3:             return
String.valueOf(productos[row-1].precioOferta());
						case             4:             return
String.valueOf(cantidades[row-1]);
						case 5: return comercios[row-1];
					}
				}
				return "0";
			}


			public  void   setValueAt(Object  aValue,  int  rowIndex,  int
columnIndex) {
				if (columnIndex==4 && rowIndex>0) {
					try {
						cantidades[rowIndex-1]                =
Integer.parseInt((String)aValue);
					} catch (ClassCastException ex) {
						cantidades[rowIndex-1] = 0;
					} catch (NumberFormatException ex) {
						cantidades[rowIndex-1] = 0;
					}
				}
			}


			public boolean isCellEditable(int row, int column) {
				return column==4 && row>0;
			}

			public String  getColumnName(int columnIndex) {
				switch (columnIndex) {
					case 0: return "Nombre";
					case 1: return "Precio";
					case 2: return "Características";
					case 3: return "Precio Socios";
					case 4: return "Cantidad";
```

```
                                case 5: return "Comercio";
                            }
                            return "";
                        }

                }

        }
```

ProductosComercio.java

```
package Centro;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//import com.sun.java.swing.*;
import javax.swing.*;
import javax.swing.table.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.ORB;
import java.net.*;
import java.util.*;

//import com.sun.java.swing.UIManager;
public class ProductosComercio extends JApplet {
  boolean isStandalone = false;
  String IOR;
  String comercio;
  JTable tablaProductos = new JTable();
  JLabel jLabel1 = new JLabel();
//Get a parameter value
       FactoriaComercio factoria=null;
       Producto productos[];
       int cantidades[];
  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JTextField textoBuscar = new JTextField();
  JButton jButton3 = new JButton();

  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }

  //Construct the applet

  public ProductosComercio() {
  }
//Initialize the applet

  public void init() {
    try { IOR = this.getParameter("IOR", ""); } catch (Exception e) {
e.printStackTrace(); }
     try  {  comercio  =  this.getParameter("COMERCIO",  "");  }  catch
(Exception e) { e.printStackTrace(); }
     try {
     jbInit();
     }
     catch (Exception e) {
     e.printStackTrace();
     }
  }
  //static {
  //  try {
  //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
```

```
        //                                      //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
        //                                       UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
        //  }
        //  catch (Exception e) {}
        //}
      //Component initialization

        private void jbInit() throws Exception {
          this.getContentPane().setLayout(null);
          this.setSize(550,300);
          tablaProductos.setBounds(new Rectangle(34, 77, 380, 156));

            try {
                  NameComponent factoriaName[] = new NameComponent[1];
                  factoriaName[0]=new NameComponent("FactoriaComercio","");
                  org.omg.CORBA.Object             factoriaObj             =
ConstantesCliente.obtenerNamingContextCentro(this).resolve(factoriaName);
                  factoria = FactoriaComercioHelper.narrow(factoriaObj);
            } catch (org.omg.CORBA.UserException ex) {
                  System.err.println("Excepción  no  esperada,  finalización
anormal");
                  ex.printStackTrace();
            }

            productos = factoria.listarProductos(comercio);
            Arrays.sort(productos,new ComparadorProductos());
            cantidades = new int[productos.length];

            ContenidoTablaProductos     contenidoTablaProductos     =     new
ContenidoTablaProductos();
            tablaProductos.setModel(contenidoTablaProductos);
          jLabel1.setText("Seleccione los productos deseados:");
          jLabel1.setBounds(new Rectangle(25, 21, 228, 30));
          jButton1.setText("Aceptar");
          jButton1.setBounds(new Rectangle(77, 262, 81, 23));
          jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton1_actionPerformed(e);
            }
          });
          jButton2.setText("Cancelar");
          jButton2.setBounds(new Rectangle(210, 261, 93, 23));
          jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton2_actionPerformed(e);
            }
          });
          textoBuscar.setText("");
          textoBuscar.setBounds(new Rectangle(425, 84, 100, 22));
          jButton3.setText("Buscar");
          jButton3.setBounds(new Rectangle(441, 129, 81, 23));
          jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton3_actionPerformed(e);
            }
          });
          this.getContentPane().add(tablaProductos, null);
          this.getContentPane().add(jLabel1, null);
          this.getContentPane().add(jButton1, null);
          this.getContentPane().add(jButton2, null);
          this.getContentPane().add(textoBuscar, null);
          this.getContentPane().add(jButton3, null);
        }
      //Get Applet information

        public String getAppletInfo() {
```

```
          return "Applet Information";
        }
      //Get parameter info

        public String[][] getParameterInfo() {
          String pinfo[][] =
          {
            {"IOR", "String", ""},
            {"COMERCIO", "String", ""},
          };
          return pinfo;
        }

        void jButton3_actionPerformed(ActionEvent e) {
          // Boton buscar
             for (int i=0; i<productos.length; i++) {
                    if
(productos[i].nombre().toUpperCase().equals(textoBuscar.getText().toUpperCase(
))) {
                           tablaProductos.setRowSelectionInterval(i+1,i+1);
                           break;
                    }
             }
        }

        void jButton1_actionPerformed(ActionEvent e) {
          // Boton Aceptar

             tablaProductos.editCellAt(0,4);

             String args[] = {};
             ORB orb= ORB.init(args,null);

             org.omg.CORBA.Object objCarro = orb.string_to_object(IOR);
             CarroCompra carro = CarroCompraHelper.narrow(objCarro);

             for (int i=0; i<cantidades.length; i++) {
                    if (cantidades[i]>0)

      carro.anadirProducto(productos[i],cantidades[i],comercio);
             }

             try {
                    getAppletContext().showDocument(new     URL("http://"     +
getCodeBase().getHost()+"/cgi-bin/SeleccionCentro.cgi?" + IOR));
             } catch (MalformedURLException ex) {}
        }

        void jButton2_actionPerformed(ActionEvent e) {
          // Boton Cancelar

             try {
                    getAppletContext().showDocument(new     URL("http://"     +
getCodeBase().getHost()+"/cgi-bin/SeleccionCentro.cgi?" + IOR));
             } catch (MalformedURLException ex) {}
        }

             class ContenidoTablaProductos extends AbstractTableModel {
                    public int getRowCount() {
                           return productos.length + 1;
                    }

                    public int getColumnCount() {
                           return 5;
                    }

                    public Object getValueAt(int row, int column) {
                           // Listar los elementos de la variable productos
```

```
                        if (row==0) {
                                switch (column) {
                                        case 0: return "Nombre";
                                        case 1: return "Precio";
                                        case 2: return "Características";
                                        case 3: return "Precio Socios";
                                        case 4: return "Cantidad";
                                }
                        }
                        else {
                                switch (column) {
                                        case    0:    return    productos[row-
1].nombre();
                                        case             1:             return
String.valueOf(productos[row-1].precio());
                                        case    2:    return    productos[row-
1].caracteristicas();
                                        case             3:             return
String.valueOf(productos[row-1].precioOferta());
                                        case             4:             return
String.valueOf(cantidades[row-1]);
                                }
                        }
                        return "0";
                }


                public  void   setValueAt(Object  aValue,  int  rowIndex,  int
columnIndex) {
                        if (columnIndex==4 && rowIndex>0) {
                                try {
                                        cantidades[rowIndex-1]                =
Integer.parseInt((String)aValue);
                                } catch (ClassCastException ex) {
                                        cantidades[rowIndex-1] = 0;
                                } catch (NumberFormatException ex) {
                                        cantidades[rowIndex-1] = 0;
                                }
                        }
                }


                public boolean isCellEditable(int row, int column) {
                        return column==4 && row>0;
                }

                public String  getColumnName(int columnIndex) {
                        switch (columnIndex) {
                                case 0: return "Nombre";
                                case 1: return "Precio";
                                case 2: return "Características";
                                case 3: return "Precio Socios";
                                case 4: return "Cantidad";
                        }
                        return "";
                }

        }

}
```

SeleccionCentro.java

```
package Centro;

import java.awt.*;
import java.awt.event.*;
```

```
      import java.applet.*;
      import javax.swing.*;
      import org.omg.CosNaming.*;
      import java.net.*;

      //import com.sun.java.swing.UIManager;
      public class SeleccionCentro extends JApplet {
        boolean isStandalone = false;
        String carroActual;
        JLabel jLabel1 = new JLabel();
        JComboBox listaCentros = new JComboBox();
        JButton jButton1 = new JButton();
        JButton jButton2 = new JButton();
        JButton jButton3 = new JButton();

            String IOR;
            FactoriaComercio factoria=null;
      //Get a parameter value

        public String getParameter(String key, String def) {
          return isStandalone ? System.getProperty(key, def) :
            (getParameter(key) != null ? getParameter(key) : def);
        }

        //Construct the applet

        public SeleccionCentro() {
        }
      //Initialize the applet

        public void init() {
          try  {  carroActual  =  this.getParameter("CarroCompra",  "null");  }
catch (Exception e) { e.printStackTrace(); }
          try {
          jbInit();
          }
          catch (Exception e) {
          e.printStackTrace();
          }

             IOR=getParameter("IOR");

             // Inicializo la lista de centros

             try {
                    NameComponent factoriaName[] = new NameComponent[1];
                    factoriaName[0]=new NameComponent("FactoriaComercio","");
                    org.omg.CORBA.Object             factoriaObj             =
ConstantesCliente.obtenerNamingContextCentro(this).resolve(factoriaName);
                    factoria = FactoriaComercioHelper.narrow(factoriaObj);
             } catch (org.omg.CORBA.UserException ex) {
                    System.err.println("Excepción  no  esperada,  finalización
anormal");
                    ex.printStackTrace();
             }

             String comercios[] = {};

             comercios = factoria.listarComercios();

             for (int i=0; i<comercios.length; i++)
                    listaCentros.addItem(comercios[i]);
        }
        //static {
        //  try {
        //                                        //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
```

```
        //                                      //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
        //                                       UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
        //  }
        //  catch (Exception e) {}
        //}
      //Component initialization

        private void jbInit() throws Exception {
          jLabel1.setText("Seleccione un centro a visitar:");
          jLabel1.setBounds(new Rectangle(45, 37, 188, 22));
          listaCentros.setBounds(new Rectangle(84, 69, 123, 22));
          jButton1.setText("Visitar");
          jButton1.setBounds(new Rectangle(115, 131, 71, 23));
          jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton1_actionPerformed(e);
            }
          });
          jButton2.setText("Realizar pedido");
          jButton2.setBounds(new Rectangle(25, 251, 132, 23));
          jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton2_actionPerformed(e);
            }
          });
          jButton3.setText("Carro");
          jButton3.setBounds(new Rectangle(312, 12, 81, 23));
          jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
              jButton3_actionPerformed(e);
            }
          });
          this.getContentPane().setLayout(null);
          this.setSize(400,300);
          this.getContentPane().add(jLabel1, null);
          this.getContentPane().add(listaCentros, null);
          this.getContentPane().add(jButton1, null);
          this.getContentPane().add(jButton2, null);
          this.getContentPane().add(jButton3, null);
        }
      //Get Applet information

        public String getAppletInfo() {
          return "Applet Information";
        }
      //Get parameter info

        public String[][] getParameterInfo() {
          String pinfo[][] =
          {
            {"CarroCompra", "String", ""},
          };
          return pinfo;
        }

        void jButton3_actionPerformed(ActionEvent e) {
            // Pulsado el botón del carro de la compra
             try {
                    getAppletContext().showDocument(new     URL("http://"     +
getCodeBase().getHost()+"/cgi-bin/CarroCompra.cgi?" + IOR));
             } catch (MalformedURLException ex) {
             }
        }

        void jButton1_actionPerformed(ActionEvent e) {
            // Pulsado el botón de visitar
```

```
            String                              dirWeb                              =
factoria.direccionWeb((String)listaCentros.getSelectedItem());
            try {
                  getAppletContext().showDocument(new  URL(dirWeb  +  "?"  +
IOR));
            } catch (MalformedURLException ex) {
            }
        }

        void jButton2_actionPerformed(ActionEvent e) {
            // Pulsado el botón para realizar pedido
        }
      }
```

ServidorCentro.java

```
      package Centro;

      import org.omg.CORBA.*;
      import org.omg.CosNaming.*;
      import java.io.*;

      public class ServidorCentro {

            public static void main (String args[]) {
                  // Creo un nuevo ORB
                  ORB orb = ORB.init(args, null);

                  // Voy a incluir la factoría de socios en el árbol de
nombres.

                  // En primer lugar obtengo el contexto raíz
                  NamingContext rootContext=null;
                  try {
                        org.omg.CORBA.Object objRef =

      orb.resolve_initial_references("NameService");
                        rootContext = NamingContextHelper.narrow(objRef);
                  } catch (org.omg.CORBA.ORBPackage.InvalidName ex) {
                        System.err.println("Error,   no   se   encuentra   el
servicio de nombres");
                        System.exit(-1);
                  }

            try {

                  //  A  continuación  me  voy  a  meter  por  la  rama  del
CentroVirtual
                  NameComponent centroContext[] = new NameComponent[1];

                  centroContext[0] = new NameComponent("CentroVirtual", "");

                  NamingContext servidores;

                  try {
                        // Pruebo a ver si existe
                        servidores                                          =
NamingContextHelper.narrow(rootContext.resolve(centroContext));
                  }  catch  (org.omg.CosNaming.NamingContextPackage.NotFound
ex) {
                        // Si no existe la creo
                        servidores                                          =
rootContext.bind_new_context(centroContext);
                  }

                  // Creo la rama para los productos
                  centroContext[0] = new NameComponent("Productos", "");
```

```
                    try {
                          // Pruebo a ver si existe
                          ConstantesServidor.productosNaming                  =
NamingContextHelper.narrow(rootContext.resolve(centroContext));
                    }  catch  (org.omg.CosNaming.NamingContextPackage.NotFound
ex) {
                          // Si no existe la creo
                          ConstantesServidor.productosNaming                  =
rootContext.bind_new_context(centroContext);
                    }

                    // Creo una factoría de socios
                    FactoriaSocioImpl factSocio = new FactoriaSocioImpl();
                    orb.connect(factSocio);

                    // Y asocio la factoría con un nombre.
                    centroContext[0] = new NameComponent("FactoriaSocio", "");

                    servidores.rebind(centroContext, factSocio);

                    // Creo una factoría de comercios
                    FactoriaComercioImpl       factComercio       =       new
FactoriaComercioImpl();
                    orb.connect(factComercio);

                    // Y asocio la factoría con un nombre.
                    centroContext[0]  =  new  NameComponent("FactoriaComercio",
"");

                    servidores.rebind(centroContext, factComercio);


                    System.out.println("Listo");

                    // Espero a que se pulse enter para terminar la ejecución
del servidor
                    try {
                          BufferedReader teclado;
                          teclado       =       new       BufferedReader(new
InputStreamReader(System.in));
                          String linea=teclado.readLine();
                    } catch (IOException ex) {}

                    servidores.unbind(centroContext);

             } catch (org.omg.CORBA.UserException ex) {
                    System.err.println("Excepción  no  esperada,  finalización
anormal");
                    ex.printStackTrace();
             }

             }
      }
```

B.2 Centro comercial virtual en C++

CarroCompraImpl.h

```
#ifndef CARROCOMPRAIMPL_H
#define CARROCOMPRAIMPL_H

#include <OB/CORBA.h>
#include "CentroComercial_skel.h"
```

```
class Centro_CarroCompra_Impl : public Centro_CarroCompra_skel {

private:
       Centro_Internauta_var refInternautavar;
       Centro_ListaProductosCantidad_var refProductovar;

public:
       Centro_CarroCompra_Impl();
         Centro_Internauta_ptr refInternauta();
         void refInternauta(Centro_Internauta_ptr arg);
         Centro_ListaProductosCantidad* refProducto();
         void refProducto(const Centro_ListaProductosCantidad& arg);
         void    anadirProducto(Centro_Producto_ptr    prod,    CORBA_Long
cantidad, const char *nombreComercio);
          void        listarProductos(Centro_ListaProductos*&        prod,
ListaCantidad*& cantidad, ListaComercios*& comercios);
          void   cambiarCantidad(Centro_Producto_ptr   prod,   const   char
*comercio, CORBA_Long cantidad);
};

#endif
```

CarroCompraImpl.cpp

```
#include "CarroCompraImpl.h"

Centro_CarroCompra_Impl::Centro_CarroCompra_Impl() {
   refProductovar = new Centro_ListaProductosCantidad;
}

Centro_Internauta_ptr Centro_CarroCompra_Impl::refInternauta() {
   return Centro_Internauta::_duplicate(refInternautavar);
}

void Centro_CarroCompra_Impl::refInternauta(Centro_Internauta_ptr arg) {
   refInternautavar = Centro_Internauta::_duplicate(arg);
}

Centro_ListaProductosCantidad* Centro_CarroCompra_Impl::refProducto() {
   return refProductovar;
}

void                          Centro_CarroCompra_Impl::refProducto(const
Centro_ListaProductosCantidad& arg) {
   Centro_ListaProductosCantidad *refProductovar2;
   (*refProductovar2) = arg;
   refProductovar = refProductovar2;
}

void  Centro_CarroCompra_Impl::anadirProducto(Centro_Producto_ptr  prod,
CORBA_Long cantidad,
      const char *nombreComercio) {

   int i;

   for (i=0; i<refProductovar->length(); i++) {
      if (refProductovar[i].prod->idProducto()==prod->idProducto()
      && !strcmp(refProductovar[i].nombreComercio,nombreComercio)) {
         refProductovar[i].cantidad+=cantidad;
         return;
      }
   }

   Centro_ProductosCantidad_var nuevo = new Centro_ProductosCantidad;

   nuevo->prod = Centro_Producto::_duplicate(prod);
```

```
         nuevo->cantidad = cantidad;
         nuevo->nombreComercio = CORBA_string_dup(nombreComercio);

         refProductovar->insert(nuevo);

      }

      void    Centro_CarroCompra_Impl::listarProductos(Centro_ListaProductos*&
prod, ListaCantidad*& cantidad,
         ListaComercios*& comercios) {

         int i;
         Centro_ListaProductos *lProductos = new Centro_ListaProductos;
         ListaCantidad *lCantidad = new ListaCantidad;
         ListaComercios *lComercios = new ListaComercios;

         for (i=0; i<refProductovar->length(); i++) {
            lProductos-
>insert(Centro_Producto::_duplicate(refProductovar[i].prod));
            lCantidad->insert(refProductovar[i].cantidad);
            lComercios-
>insert(CORBA_string_dup(refProductovar[i].nombreComercio));
         }

         prod = lProductos;
         cantidad = lCantidad;
         comercios = lComercios;
      }

      void  Centro_CarroCompra_Impl::cambiarCantidad(Centro_Producto_ptr  prod,
const char *comercio, CORBA_Long cantidad) {
         int i;

         for (i=0; i<refProductovar->length(); i++) {
            if (refProductovar[i].prod->idProducto()==prod->idProducto()
            && !strcmp(refProductovar[i].nombreComercio,comercio)) {
               refProductovar[i].cantidad=cantidad;
               return;
            }
         }
      }
```

InternautaImpl.h

```
      #ifndef INTERNAUTAIMPL_H
      #define INTERNAUTAIMPL_H

      #include <OB/CORBA.h>
      #include "CentroComercial_skel.h"

      class Centro_Internauta_Impl : public Centro_Internauta_skel {

         CORBA_String_var  nombrevar,  apellido1var,  apellido2var,  telefonovar,
correoElectronicovar, numeroCuentavar, direccionvar;

      public:

         char* nombre();

         void nombre(const char *arg);

         char *apellido1();

         void apellido1(const char *arg);

         char *apellido2();
```

```
   void apellido2(const char *arg);

   char *telefono();

   void telefono(const char *arg);

   char *correoElectronico();

   void correoElectronico(const char *arg);

   char *numeroCuenta();

   void numeroCuenta(const char *arg);

   char *direccion();

   void direccion(const char *arg);

};

#endif
```

InternautaImpl.cpp

```
#include "InternautaImpl.h"

char *Centro_Internauta_Impl::nombre() {
      return CORBA_string_dup(nombrevar);
};

void Centro_Internauta_Impl::nombre(const char *arg) {
      nombrevar=CORBA_string_dup(arg);
}

char *Centro_Internauta_Impl::apellido1() {
      return CORBA_string_dup(apellido1var);
}

void Centro_Internauta_Impl::apellido1(const char *arg) {
      apellido1var=CORBA_string_dup(arg);
}

char *Centro_Internauta_Impl::apellido2() {
      return CORBA_string_dup(apellido2var);
}

void Centro_Internauta_Impl::apellido2(const char *arg) {
      apellido2var=CORBA_string_dup(arg);
}

char *Centro_Internauta_Impl::telefono() {
      return CORBA_string_dup(telefonovar);
}

void Centro_Internauta_Impl::telefono(const char *arg) {
      telefonovar=CORBA_string_dup(arg);
}

char *Centro_Internauta_Impl::correoElectronico() {
      return CORBA_string_dup(correoElectronicovar);
}

void Centro_Internauta_Impl::correoElectronico(const char *arg) {
      correoElectronicovar=CORBA_string_dup(arg);
}

char *Centro_Internauta_Impl::numeroCuenta() {
```

```
        return CORBA_string_dup(numeroCuentavar);
    }

    void Centro_Internauta_Impl::numeroCuenta(const char *arg) {
        numeroCuentavar=CORBA_string_dup(arg);
    }

    char *Centro_Internauta_Impl::direccion() {
        return CORBA_string_dup(direccionvar);
    }

    void Centro_Internauta_Impl::direccion(const char *arg) {
        direccionvar = CORBA_string_dup(arg);
    }
```

SocioImpl.h

```
    #ifndef SOCIOIMPL_H
    #define SOCIOIMPL_H

    #include <OB/CORBA.h>
    #include "CentroComercial_skel.h"
    #include "InternautaImpl.h"

    class Centro_Socio_Impl : public Centro_Socio_skel, public
Centro_Internauta_Impl
    {

       CORBA_String_var idSociovar, clavevar;

    public:
       char *idSocio();
       void idSocio(const char *arg);
       char *clave();
       void clave(const char *arg);
    };

    #endif
```

SocioImpl.cpp

```
    #include "SocioImpl.h"

    char *Centro_Socio_Impl::idSocio() {
        return CORBA_string_dup(idSociovar);
    }

    void Centro_Socio_Impl::idSocio(const char *arg) {
        idSociovar=CORBA_string_dup(arg);
    }

    char *Centro_Socio_Impl::clave() {
        return CORBA_string_dup(clavevar);
    }

    void Centro_Socio_Impl::clave(const char *arg) {
        clavevar=CORBA_string_dup(arg);
    }
```

ProductoImpl.h

```
    #ifndef PRODUCTOIMPL_H
    #define PRODUCTOIMPL_H

    #include <OB/CORBA.h>
```

```
#include "CentroComercial_skel.h"


class Centro_Producto_Impl : public Centro_Producto_skel {


   CORBA_String_var nombrevar, caracteristicasvar;

   CORBA_Long preciovar, precioOfertavar, idProductovar;

public:
   Centro_Producto_Impl(CORBA_Long  idProducto,  const  char  *nombre,
CORBA_Long precio,
      const char *caracteristicas, CORBA_Long precioOferta);

   char *nombre();
   void nombre(const char *arg);
   char *caracteristicas();
   void caracteristicas(const char *arg);
   CORBA_Long precio();
   void precio(CORBA_Long arg);
   CORBA_Long precioOferta();
   void precioOferta(CORBA_Long arg);
   CORBA_Long idProducto();
   void idProducto(CORBA_Long arg);
};

#endif
```

ProductoImpl.cpp

```
#include "ProductoImpl.h"

Centro_Producto_Impl::Centro_Producto_Impl(CORBA_Long  idProducto,  const
char* nombre, CORBA_Long precio,
      const char* caracteristicas, CORBA_Long precioOferta) {

   idProductovar = idProducto;
   nombrevar = nombre;
   preciovar = precio;
   caracteristicasvar = caracteristicas;
   precioOfertavar = precioOferta;
}

char *Centro_Producto_Impl::nombre() {
   return CORBA_string_dup(nombrevar);
}

void Centro_Producto_Impl::nombre(const char *arg) {
   nombrevar = CORBA_string_dup(arg);
}

char *Centro_Producto_Impl::caracteristicas() {
   return CORBA_string_dup(caracteristicasvar);
}

void Centro_Producto_Impl::caracteristicas(const char *arg) {
   caracteristicasvar = CORBA_string_dup(arg);
}

CORBA_Long Centro_Producto_Impl::precio() {
   return preciovar;
}

void Centro_Producto_Impl::precio(CORBA_Long arg) {
   preciovar = arg;
}
```

```
CORBA_Long Centro_Producto_Impl::precioOferta() {
   return precioOfertavar;
}

void Centro_Producto_Impl::precioOferta(CORBA_Long arg) {
   precioOfertavar = arg;
}

CORBA_Long Centro_Producto_Impl::idProducto() {
   return idProductovar;
}

void Centro_Producto_Impl::idProducto(CORBA_Long arg) {
   idProductovar = arg;
}
```

FactoriaSocioImpl.h

```
#ifndef FACTORIASOCIOIMPL_H
#define FACTORIASOCIOIMPL_H

#include <OB/CORBA.h>
#include "CentroComercial_skel.h"

class Centro_FactoriaSocio_Impl : public Centro_FactoriaSocio_skel
{

   Centro_CarroCompra_ptr  validarSocio(const  char  *id,  const  char
*clave);


   CORBA_Long altaSocio(const char *idSocio, const char *clave, const
char *nombre, const char *apellido1,
      const  char  *apellido2,  const  char  *telefono,  const  char
*correoElectronico, const char *numeroCuenta,
      const char *direccion);
};

#endif
```

FactoriaSocioImpl.cpp

```
#include "FactoriaSocioImpl.h"
#include <libpq++.H>
#include <string>
#include "ConstantesServidor.h"
#include "CarroCompraImpl.h"
#include "SocioImpl.h"
#include <libpq++.H>

Centro_CarroCompra_ptr     Centro_FactoriaSocio_Impl::validarSocio(const
char *id, const char *clave)
{

   CORBA_String_var  idS  =  CORBA_string_dup("SELECT  clave  FROM  socio
WHERE idSocio='");
   idS += CORBA_string_dup(id);
   idS += CORBA_string_dup("'");
   PgDatabase                    *consultas=                    new
PgDatabase(ConstantesServidor::nombreconsultas.data());
   if ( consultas->ConnectionBad() ) {
      cout << "No se ha podido establecer la conexion..." << endl
         << "Mensaje de error devuelto: " << consultas->ErrorMessage()
<< endl;
      return 0;
```

```
         }

         ExecStatusType resultado = consultas->Exec(idS);

         // Consulta realizada

         //  Lo  primero  se  comprueba  que  hay  alguna  fila  en  la  tabla
correspondiente

         // a este socio -> que esta dado de alta

         if (resultado!=PGRES_TUPLES_OK || consultas->Tuples()==0) {
            // No existe tal socio

            return NULL;
         }

         // Si el socio existe compruebo que la clave es correcta

         string claveVerdadera = consultas->GetValue(0,"clave");

         if (!(claveVerdadera==clave)) {

             //System.out.println("Clave erronea, introducida: " + clave + "
devería ser: " + claveVerdadera);

             // La clave no es correcta

             return NULL;

         }

         // Tras la validacion creo un carro de la compra para este socio

         Centro_CarroCompra_var carro = new Centro_CarroCompra_Impl;

      //   ConstantesServidor::orb->connect(carro);

         // Creo el socio que estara asociado al carro

         Centro_Socio_var socio = new Centro_Socio_Impl;

         socio->idSocio(id);

         socio->clave(clave);

         // Asocio el socio al carro

         carro->refInternauta(Centro_Socio::_duplicate(socio));

         // Devuelvo el carro de la compra

         return Centro_CarroCompra::_duplicate(carro);

      };

      CORBA_Long  Centro_FactoriaSocio_Impl::altaSocio(const  char  *idSocio,
const char *clave,
         const char *nombre, const char *apellido1, const char *apellido2,
         const  char  *telefono,  const  char  *correoElectronico,  const  char
*numeroCuenta,
         const char *direccion)
      {
         string idSocioS = idSocio;
         string claveS = clave;
         string nombreS = nombre;
         string apellido1S = apellido1;
         string apellido2S = apellido2;
```

```
         string telefonoS = telefono;
         string correoElectronicoS = correoElectronico;
         string numeroCuentaS = numeroCuenta;
         string direccionS = direccion;

         PgDatabase                        *consultas=                        new
PgDatabase(ConstantesServidor::nombreconsultas.data());
         if ( consultas->ConnectionBad() ) {
            cout << "No se ha podido establecer la conexion..." << endl
               << "Mensaje de error devuelto: " << consultas->ErrorMessage()
<< endl;
            return 0;
         }


         return consultas->ExecCommandOk(
            ("INSERT INTO Socio VALUES ('" + idSocioS + "','" + claveS +
"','" +
              nombreS + "','" + apellido1S + "','" + apellido2S + "','" +
              telefonoS + "','" + correoElectronicoS + "','" + numeroCuentaS
              + "','" + direccionS + "');").data());

      };
```

FactoriaComercioImpl.h

```
      #ifndef FACTORIACOMERCIOIMPL_H
      #define FACTORIACOMERCIOIMPL_H

      #include <OB/CORBA.h>
      #include "CentroComercial.h"
      #include "CentroComercial_skel.h"

      class Centro_FactoriaComercio_Impl : public Centro_FactoriaComercio_skel
      {
      public:
            ListaComercios* listarComercios();
              char* direccionWeb(const char* nombre);
            Centro_ListaProductos* listarProductos(const char* Comercio);
      };

      #endif
```

FactoriaComercioImpl.cpp

```
      #include "FactoriaComercioImpl.h"
      #include <libpq++.H>
      #include "ConstantesServidor.h"
      #include "CentroComercial.h"
      #include "ProductoImpl.h"
      #include <stdlib.h>

      Centro_FactoriaComercio::ListaComercios
*Centro_FactoriaComercio_Impl::listarComercios()
      {
         ListaComercios *comercios = new ListaComercios;
         int i;

         CORBA_String_var   sql   =   CORBA_string_dup("SELECT   nombre   FROM
comercio;");

         PgDatabase                        *consultas=                        new
PgDatabase(ConstantesServidor::nombreconsultas.data());
         if ( consultas->ConnectionBad() ) {
            cout << "No se ha podido establecer la conexion..." << endl
               << "Mensaje de error devuelto: " << consultas->ErrorMessage()
<< endl;
```

```
            return 0;
         }

         // Comienzo realizando la consulta de los comercios existentes
         ExecStatusType resultado = consultas->Exec(sql);

         if (resultado!=PGRES_TUPLES_OK)
            return comercios;

            // A continuacion tomo el nombre de todos los comercios

         comercios->length(consultas->Tuples());

         for (i=0; i<consultas->Tuples(); i++)
             (*comercios)[i]=CORBA_string_dup(consultas-
>GetValue(i,"nombre"));

         return comercios;
      }

      char* Centro_FactoriaComercio_Impl::direccionWeb(const char* nombre) {

         CORBA_String_var  sql  =  CORBA_string_dup("SELECT  direccionWeb  FROM
comercio WHERE nombre='");
         sql += CORBA_string_dup(nombre);
         sql += CORBA_string_dup("';");

         PgDatabase                        *consultas=                        new
PgDatabase(ConstantesServidor::nombreconsultas.data());
         if ( consultas->ConnectionBad() ) {
            cout << "No se ha podido establecer la conexion..." << endl
               << "Mensaje de error devuelto: " << consultas->ErrorMessage()
<< endl;
            return 0;
         }


         ExecStatusType resultado = consultas->Exec(sql);

         if (resultado!=PGRES_TUPLES_OK)
            return NULL;

         return CORBA_string_dup(consultas->GetValue(0,"direccionWeb"));
      }

      Centro_ListaProductos*
Centro_FactoriaComercio_Impl::listarProductos(const char* Comercio) {
         int i;
         Centro_ListaProductos *productos = new Centro_ListaProductos;

         CORBA_String_var ComercioS = CORBA_string_dup("SELECT * FROM Producto
WHERE  idproducto  IN  (SELECT  idproducto  FROM  comercio_producto  WHERE
nombre_comercio = '");
         ComercioS += CORBA_string_dup(Comercio);
         ComercioS += CORBA_string_dup("');");

         PgDatabase                        *consultas=                        new
PgDatabase(ConstantesServidor::nombreconsultas.data());
         if ( consultas->ConnectionBad() ) {
            cout << "No se ha podido establecer la conexion..." << endl
               << "Mensaje de error devuelto: " << consultas->ErrorMessage()
<< endl;
            return 0;
         }

         ExecStatusType resultado = consultas->Exec(ComercioS);

         if (resultado!=PGRES_TUPLES_OK)
```

```
      return productos;

   productos->length(consultas->Tuples());

   for (i=0; i<consultas->Tuples(); i++) {
      Centro_Producto_var nuevoProducto = new Centro_Producto_Impl(
         strtol(consultas->GetValue(i,"idproducto"),0,10),
         consultas->GetValue(i,"nombre"),
         strtol(consultas->GetValue(i,"precio"),0,10),
         consultas->GetValue(i,"caracteristicas"),
         strtol(consultas->GetValue(i,"preciooferta"),0,10));

       (*productos)[i]=Centro_Producto::_duplicate(nuevoProducto);
   }

   return productos;

}
```

ConstantesServidor.h

```
#ifndef CONSTANTESSERVIDOR_H
#define CONSTANTESSERVIDOR_H

#include <libpq++.H>
#include <OB/CORBA.h>
#include <OB/CosNaming.h>
#include <string.h>

class ConstantesServidor {
public:
       static string servidor;

       static string nombreBaseDatos;

       static string usuario;

       static string claveUsuario;

       static string nombreconsultas;

        static CORBA_ORB_var orb;

        static CORBA_BOA_var boa;

        static void inicializar(int arg, char *narg[]);


};

#endif
```

ConstantesServidor.cpp

```
#include "ConstantesServidor.h"
#include <iostream.h>

string ConstantesServidor::servidor = "192.168.105.1";

string ConstantesServidor::nombreBaseDatos = "centrovirtual";

string ConstantesServidor::usuario = "jros";

string ConstantesServidor::claveUsuario = "123456";

string ConstantesServidor::nombreconsultas;
```

```
CORBA_ORB_var ConstantesServidor::orb;

CORBA_BOA_var ConstantesServidor::boa;

void ConstantesServidor::inicializar(int narg, char *arg[])
{
      nombreconsultas = "dbname=" + nombreBaseDatos + " user=" +
usuario
            + " password=" + claveUsuario + "host=" + servidor;

      orb = CORBA_ORB_init(narg,arg);
      boa = orb->BOA_init(narg,arg);
}
```

ServidorCentro.cpp

```
#include <OB/CORBA.h>
#include <OB/CosNaming.h>
#include "CentroComercial.h"
#include "ConstantesServidor.h"
#include "FactoriaSocioImpl.h"
#include "FactoriaComercioImpl.h"
#include <iostream.h>
#include <signal.h>

void finalizar(int);

int main (int narg, char *arg[])
{

   ConstantesServidor::inicializar(narg,arg);

   // Uso el servicio de nombres
   CORBA_Object_var obj;

   try
   {
      obj          =          ConstantesServidor::orb          ->
resolve_initial_references("NameService");
   }
   catch(const CORBA_ORB::InvalidName&)
   {
      cerr << arg[0] << ": no puede resolver 'NameService'" << endl;
      return 1;
   }

   if(CORBA_is_nil(obj))
   {
      cerr << arg[0] << ": 'NameService' es una referencia a nil" <<
endl;
      return 1;
   }

   CosNaming_NamingContext_var              rootContext              =
CosNaming_NamingContext::_narrow(obj);

   if(CORBA_is_nil(rootContext))
   {
      cerr << arg[0]
        << ": 'NameService' no es un servidor de nombres"
        << endl;
      return 1;
   }

   try
   {
      CosNaming_Name centroContext;
```

```
            centroContext.length(1);
            centroContext[0].id = CORBA_string_dup("CentroVirtual");
            centroContext[0].kind = CORBA_string_dup("");

            CosNaming_NamingContext_var servidores;

            try {
               // Pruebo a ver si existe
               servidores    =    CosNaming_NamingContext::_narrow(rootContext-
>resolve(centroContext));

            } catch (CosNaming_NamingContext::NotFound ex) {
               // Si no existe la creo
               servidores = rootContext->bind_new_context(centroContext);
            }


            // Creo una factoria de socios

            Centro_FactoriaSocio_var          factSocio          =          new
Centro_FactoriaSocio_Impl;
            ConstantesServidor::orb->connect(factSocio);

            // Y asocio la factoria con un nombre.

            CosNaming_Name factoriaSocioContext;
            factoriaSocioContext.length(1);
            factoriaSocioContext[0].id = CORBA_string_dup("FactoriaSocio");
            factoriaSocioContext[0].kind = CORBA_string_dup("");
            servidores->rebind(factoriaSocioContext, factSocio);

            Centro_FactoriaComercio_var        factComercio        =        new
Centro_FactoriaComercio_Impl;
            ConstantesServidor::orb->connect(factComercio);

            CosNaming_Name factoriaComercioContext;
            factoriaComercioContext.length(1);
            factoriaComercioContext[0].id                                      =
CORBA_string_dup("FactoriaComercio");
            factoriaComercioContext[0].kind = CORBA_string_dup("");
            servidores->rebind(factoriaComercioContext, factComercio);


            cout  <<  "Servidor  del  centro  virtual  listo.  Pulse  CTRL+Z  para
finalizar" << endl;
            signal(SIGTSTP,finalizar);

            ConstantesServidor::boa-
>impl_is_ready(CORBA_ImplementationDef::_nil());

            servidores->unbind(factoriaSocioContext);
            rootContext->unbind(centroContext);

         }
         catch (const CosNaming_NamingContext::NotFound& ex)
         {
            cerr << arg[0] << ": Producida la excepcion 'NotFound' (";
            switch(ex.why)
            {
              case CosNaming_NamingContext::missing_node:
                                    cerr << "nodo no encontrado";
                                 break;

              case CosNaming_NamingContext::not_context:
                                    cerr << "no contexto";
                                 break;

              case CosNaming_NamingContext::not_object:
```

```
                                        cerr << "no objeto";
                                    break;
            }
            cerr << ")" << endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::CannotProceed&)
         {
            cerr << arg[0] << ": Producida la excepcion 'CannotProceed'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::InvalidName&)
         {
            cerr << arg[0] << ": Producida la excepcion 'InvalidName'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::AlreadyBound&)
         {
            cerr << arg[0] << ": Producida la excepcion 'AlreadyBound'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::NotEmpty&)
         {
            cerr << arg[0] << ": Producida la excepcion 'NotEmpty'" << endl;
            return 1;
         }

         return 0;
      }

      void finalizar(int a) {
         ConstantesServidor::boa-
>deactivate_impl(CORBA_ImplementationDef::_nil());
      }
```

ConstantesCliente.h

```
#ifndef CONSTANTESCLIENTE_H
#define CONSTANTESCLIENTE_H

#include <OB/CORBA.h>
#include <OB/CosNaming.h>

class ConstantesCliente
{
public:
   static CosNaming_NamingContext_var centroNamingContext;
   static CORBA_ORB_var orb;
   static CORBA_BOA_var boa;


   static int inicializar(int narg, char *arg[]);
};

#endif
```

ConstantesCliente.cpp

```
#include "ConstantesCliente.h"

CosNaming_NamingContext_var ConstantesCliente::centroNamingContext;
CORBA_ORB_var ConstantesCliente::orb;
CORBA_BOA_var ConstantesCliente::boa;
```

```
      int ConstantesCliente::inicializar(int narg, char *arg[])
      {
         orb = CORBA_ORB_init(narg,arg);
         boa = orb->BOA_init(narg,arg);

         // En primer lugar obtengo el contexto raiz

         CORBA_Object_var obj;

         try
         {
            obj = orb -> resolve_initial_references("NameService");
         }
         catch(const CORBA_ORB::InvalidName&)
         {
            cerr << arg[0] << ": no puede resolver 'NameService'" << endl;
            return 1;
         }

         if(CORBA_is_nil(obj))
         {
            cerr << arg[0] << ": 'NameService' es una referencia a nil" <<
endl;
            return 1;
         }

         CosNaming_NamingContext_var               rootContext               =
CosNaming_NamingContext::_narrow(obj);

         if(CORBA_is_nil(rootContext))
         {
            cerr << arg[0]
              << ": 'NameService' no es un servidor de nombres"
              << endl;
            return 1;
         }

         try {
            CosNaming_Name centroContext;
            centroContext.length(1);
            centroContext[0].id = CORBA_string_dup("CentroVirtual");
            centroContext[0].kind = CORBA_string_dup("");

            CosNaming_NamingContext_var servidores;

            try {
               // Pruebo a ver si existe
               centroNamingContext                                              =
CosNaming_NamingContext::_narrow(rootContext->resolve(centroContext));

            } catch (CosNaming_NamingContext::NotFound ex) {

               // Si no existe error
               cerr << "Error, no hay ningun Centro Virtual funcionando" <<
endl;
               return 1;
            }

            return 0;

         } catch (const CosNaming_NamingContext::NotFound& ex) {
            cerr << arg[0] << ": Producida la excepcion 'NotFound' (";
            switch(ex.why)
            {
              case CosNaming_NamingContext::missing_node:
                                       cerr << "nodo no encontrado";
                                    break;
```

```
            case CosNaming_NamingContext::not_context:
                                    cerr << "no contexto";
                                 break;

            case CosNaming_NamingContext::not_object:
                                    cerr << "no objeto";
                                 break;
            }
            cerr << ")" << endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::CannotProceed&)
         {
            cerr << arg[0] << ": Producida la excepcion 'CannotProceed'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::InvalidName&)
         {
            cerr << arg[0] << ": Producida la excepcion 'InvalidName'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::AlreadyBound&)
         {
            cerr << arg[0] << ": Producida la excepcion 'AlreadyBound'" <<
endl;
            return 1;
         }
         catch(const CosNaming_NamingContext::NotEmpty&)
         {
            cerr << arg[0] << ": Producida la excepcion 'NotEmpty'" << endl;
            return 1;
         }

         return 0;

      }
```

ClienteCentro.h

```
#ifndef CLIENTECENTRO_H
#define CLIENTECENTRO_H

#include <qt/qpushbutton.h>
#include <qt/qlabel.h>
#include <qt/qlineedit.h>
#include <qt/qcombobox.h>
#include <qt/qdialog.h>
#include <qt/qlistview.h>
#include <qt/qobject.h>
#include <OB/CORBA.h>
#include "CentroComercial.h"

class ventanaIdentificaSocio : public QDialog {
Q_OBJECT
public:
   ventanaIdentificaSocio(QWidget *padre = 0, const char *nombre=0);

private:
   QLabel *lblNombre;
   QLabel *lblClave;
   QLineEdit *txtNombre;
   QLineEdit *txtClave;
   QPushButton *btnAceptar;

private slots:
```

```
            void btnAceptarPulsado();
         };

         class ventanaSeleccionCentro : public QDialog {
         Q_OBJECT
         public:
            ventanaSeleccionCentro(QWidget *padre = 0, const char *nombre=0);

         private:
            QLabel *lbl1SC;
            QComboBox *cmbCentros;
            QPushButton *btnVisitar;
            QPushButton *btnRealizarPedido;
            QPushButton *btnCarro;

         private slots:
            void btnVisitarPulsado();
            void btnCarroPulsado();
         };

         class ventanaProductosComercio : public QDialog {
         Q_OBJECT
         public:
            ventanaProductosComercio(const char *comercio, QWidget *padre = 0,
const char *nombre=0);

         private:
            CORBA_String_var Comercio;
            QListViewItem *elementoSeleccionado;
            Centro_ListaProductos_var listaProd;

            QLabel *lbl1PC;
            QListView *lsvtablaProductos;
            QPushButton *btnAceptar;
            QPushButton *btnCancelar;
            QLineEdit *txtCantidad;

         private slots:
            void btnAceptarPulsado();
            void btnCancelarPulsado();
            void nuevaSeleccion(QListViewItem *);
         };

         class ventanaCarroCompra : public QDialog {
         Q_OBJECT
         public:
            ventanaCarroCompra(QWidget *padre = 0, const char *nombre=0);

         private:
            QListViewItem *elementoSeleccionado;
            Centro_ListaProductos_var listaProd;
            Centro_CarroCompra::ListaCantidad_var listaCant;
            Centro_CarroCompra::ListaComercios_var listaComercios;

            QLabel *lbl1CC;
            QListView *lsvtablaProductos;
            QPushButton *btnAceptar;
            QPushButton *btnCancelar;
            QLineEdit *txtCantidad;

         private slots:
            void btnAceptarPulsado();
            void btnCancelarPulsado();
            void nuevaSeleccion(QListViewItem *);
         };

         #endif
```

ClienteCentro.cpp

```
#include "ClienteCentro.h"
#include <qt/qapplication.h>
#include "ConstantesCliente.h"
#include <OB/CosNaming.h>
#include <qt/qmessagebox.h>
#include <stdio.h>
#include <stdlib.h>
#include <qt/qvalidator.h>

Centro_CarroCompra_var carroActual = NULL;

// Ventana identifica socio

ventanaIdentificaSocio::ventanaIdentificaSocio(QWidget *padre = 0, const
char *nombre=0)
   : QDialog(padre,nombre,TRUE)
{
   setMinimumSize(400,300);

   lblNombre = new QLabel("Identificador de socio:", this, "lblNombre");
   lblClave = new QLabel("Clave:", this, "lblClave");
   txtNombre = new QLineEdit(this, "txtNombre");
   txtClave = new QLineEdit(this, "txtClave");
   btnAceptar = new QPushButton("Aceptar", this, "btnAceptar");

   lblNombre->setGeometry(56,33,132,26);
   lblClave->setGeometry(56,103,61,28);
   txtNombre->setGeometry(86,63,161,27);
   txtClave->setGeometry(86,145,161,27);
   btnAceptar->setGeometry(123,212,101,33);

   txtClave->setEchoMode(QLineEdit::NoEcho);

   connect(btnAceptar,             SIGNAL(clicked()),             this,
SLOT(btnAceptarPulsado()));
}

void ventanaIdentificaSocio::btnAceptarPulsado() {
   Centro_FactoriaSocio_var factoria;
   try {
      CosNaming_Name factoriaName;
      factoriaName.length(1);
      factoriaName[0].id = CORBA_string_dup("FactoriaSocio");
      factoriaName[0].kind = CORBA_string_dup("");

      CORBA_Object_var                  factoriaObj                  =
ConstantesCliente::centroNamingContext->resolve(factoriaName);
      factoria = Centro_FactoriaSocio::_narrow(factoriaObj);

   } catch (CORBA_UserException ex) {
      cerr << "Excepcion no esperada" << endl;
   }

   Centro_CarroCompra_ptr   carro   =   factoria->validarSocio(txtNombre-
>text(),txtClave->text());

   txtNombre->setText("");
   txtClave->setText("");

   if (carro==NULL)
      QMessageBox::critical(this,    "ERROR",    "Socio    no    valido",
QMessageBox::Ok, 0);

   else {
      carroActual = carro;
      ventanaSeleccionCentro vSelCen;
      vSelCen.resize(400,300);
```

```
            this->hide();
            vSelCen.exec();
            this->done(0);
         }

      }


      // Ventana Seleccion de Centro

      ventanaSeleccionCentro::ventanaSeleccionCentro(QWidget *padre = 0, const
char *nombre=0)
         : QDialog(padre,nombre,TRUE)
      {
         int i;

         setMinimumSize(400,300);

         lbl1SC = new QLabel("Seleccione un centro a visitar:", this,
"lbl1SC");
         cmbCentros = new QComboBox(this,"cmbCentros");
         btnVisitar = new QPushButton("Visitar", this, "btnVisitar");
         btnRealizarPedido = new QPushButton("Realizar Pedido", this,
"btnRealizarPedido");
         btnCarro = new QPushButton("Carro", this, "btnCarro");

         lbl1SC->setGeometry(45, 37, 188, 22);
         cmbCentros->setGeometry(84, 69, 123, 22);
         btnVisitar->setGeometry(115, 131, 71, 23);
         btnRealizarPedido->setGeometry(25, 251, 132, 23);
         btnCarro->setGeometry(312, 12, 81, 23);

         Centro_FactoriaComercio_var factoria;
         try {
            CosNaming_Name factoriaName;
            factoriaName.length(1);
            factoriaName[0].id = CORBA_string_dup("FactoriaComercio");
            factoriaName[0].kind = CORBA_string_dup("");

            CORBA_Object_var                    factoriaObj                    =
ConstantesCliente::centroNamingContext->resolve(factoriaName);
            factoria = Centro_FactoriaComercio::_narrow(factoriaObj);

         } catch (CORBA_UserException ex) {
            cerr << "Excepcion no esperada" << endl;
         }

         Centro_FactoriaComercio::ListaComercios_var listaCom = factoria-
>listarComercios();

         for (i=0; i< listaCom->length(); i++)
            cmbCentros->insertItem(listaCom[i]);

         connect(btnVisitar,              SIGNAL(clicked()),              this,
SLOT(btnVisitarPulsado()));
         connect(btnCarro, SIGNAL(clicked()), this, SLOT(btnCarroPulsado()));
      }

      void ventanaSeleccionCentro::btnVisitarPulsado() {
         ventanaProductosComercio vProdCom(cmbCentros->currentText());
         vProdCom.resize(550,300);
         vProdCom.exec();

      }
      void ventanaSeleccionCentro::btnCarroPulsado() {
         ventanaCarroCompra vCarroCom;
         vCarroCom.resize(550,300);
         vCarroCom.exec();
```

```
      }

      // Ventana Productos de comercio

      ventanaProductosComercio::ventanaProductosComercio(const char *comercio,
QWidget *padre = 0, const char *nombre=0)
         : QDialog(padre,nombre,TRUE)
      {
         int i;

         Comercio = comercio;

         setMinimumSize(550,300);

         btnAceptar = new QPushButton("Aceptar", this, "btnAceptarPC");
         btnCancelar = new QPushButton("Cancelar", this, "btnCancelarPC");
         lsvtablaProductos = new QListView(this, "lsvtablaProductosPC");
         txtCantidad = new QLineEdit(this, "txtCantidad");
         lbl1PC = new QLabel("Cantidad: ", this, "lbl2PC");

         btnAceptar->setGeometry(77, 262, 81, 23);
         btnCancelar->setGeometry(210, 261, 93, 23);
         lsvtablaProductos->setGeometry(34, 77, 380, 156);
         lbl1PC->setGeometry(100,27,100,30);
         txtCantidad->setGeometry(230,27,160,27);

         txtCantidad->setValidator(new
QIntValidator(0,1000,this,"ValidaCantidad"));

         lsvtablaProductos->addColumn("Nombre");
         lsvtablaProductos->addColumn("Precio");
         lsvtablaProductos->addColumn("Caracteristicas");
         lsvtablaProductos->addColumn("Precio Socios");
         lsvtablaProductos->addColumn("Cantidad");

         lsvtablaProductos->setSorting(0);

         Centro_FactoriaComercio_var factoria;
         try {
            CosNaming_Name factoriaName;
            factoriaName.length(1);
            factoriaName[0].id = CORBA_string_dup("FactoriaComercio");
            factoriaName[0].kind = CORBA_string_dup("");

            CORBA_Object_var                    factoriaObj                    =
ConstantesCliente::centroNamingContext->resolve(factoriaName);
            factoria = Centro_FactoriaComercio::_narrow(factoriaObj);

         } catch (CORBA_UserException ex) {
            cerr << "Excepcion no esperada" << endl;
         }

         listaProd = factoria->listarProductos(CORBA_string_dup(Comercio));

         for (i=0; i< listaProd->length(); i++) {
            char precio[50], preciooferta[50], pos[50];
            sprintf(precio,"%ld",listaProd[i]->precio());
            sprintf(preciooferta,"%ld",listaProd[i]->precioOferta());
            sprintf(pos,"%ld",i);
            new QListViewItem(lsvtablaProductos,
               listaProd[i]->nombre(), precio,
               listaProd[i]->caracteristicas(), preciooferta, "0" , pos);
         }

         elementoSeleccionado = NULL;

         connect(btnAceptar,             SIGNAL(clicked()),             this,
SLOT(btnAceptarPulsado()));
```

```
         connect(btnCancelar,             SIGNAL(clicked()),             this,
SLOT(btnCancelarPulsado()));
         connect(lsvtablaProductos, SIGNAL(selectionChanged(QListViewItem *)),
this, SLOT(nuevaSeleccion(QListViewItem *)));
      }

      void ventanaProductosComercio::btnAceptarPulsado() {
         int i;

         if (elementoSeleccionado != NULL) {
            elementoSeleccionado->setText(4,txtCantidad->text());
         }

         QListViewItem *iter = lsvtablaProductos->firstChild();

         for (i=0; i<lsvtablaProductos->childCount(); i++) {
            CORBA_Long cant = strtol(iter->text(4),0,10);
            int pos = strtol(iter->text(5),0,10);

            if (cant>0)
               carroActual->anadirProducto(listaProd[pos],cant,Comercio);


            iter = iter->nextSibling();
         }

         this->done(0);
      }

      void ventanaProductosComercio::btnCancelarPulsado() {
         this->done(0);
      }

      void ventanaProductosComercio::nuevaSeleccion(QListViewItem *nuevoSel) {
         if (elementoSeleccionado != NULL) {
            elementoSeleccionado->setText(4,txtCantidad->text());
         }

         txtCantidad->setText(nuevoSel->text(4));
         elementoSeleccionado = nuevoSel;
      }


      // Ventana Carro de la compra

      ventanaCarroCompra::ventanaCarroCompra(QWidget  *padre  =  0,  const  char
*nombre=0)
         : QDialog(padre,nombre,TRUE)
      {
         int i;

         setMinimumSize(550,300);

         btnAceptar = new QPushButton("Aceptar", this, "btnAceptarCC");
         btnCancelar = new QPushButton("Cancelar", this, "btnCancelarCC");
         lsvtablaProductos = new QListView(this, "lsvtablaProductosCC");
         txtCantidad = new QLineEdit(this, "txtCantidadCC");
         lbl1CC = new QLabel("Cantidad: ", this, "lbl2CC");

         btnAceptar->setGeometry(77, 262, 81, 23);
         btnCancelar->setGeometry(210, 261, 93, 23);
         lsvtablaProductos->setGeometry(34, 77, 456, 156);
         lbl1CC->setGeometry(100,27,100,30);
         txtCantidad->setGeometry(230,27,160,27);

         txtCantidad->setValidator(new
QIntValidator(0,1000,this,"ValidaCantidadCC"));
```

```
         lsvtablaProductos->addColumn("Nombre");
         lsvtablaProductos->addColumn("Precio");
         lsvtablaProductos->addColumn("Caracteristicas");
         lsvtablaProductos->addColumn("Precio Socios");
         lsvtablaProductos->addColumn("Cantidad");
         lsvtablaProductos->addColumn("Comercio");

         lsvtablaProductos->setSorting(0);

         carroActual->listarProductos(listaProd,listaCant,listaComercios);

         for (i=0; i< listaProd->length(); i++) {
            char precio[50], preciooferta[50], cantidad[50], pos[50];
            sprintf(precio,"%ld",listaProd[i]->precio());
            sprintf(preciooferta,"%ld",listaProd[i]->precioOferta());
            sprintf(cantidad,"%ld",listaCant[i]);
            sprintf(pos,"%ld",i);
            new QListViewItem(lsvtablaProductos,
               listaProd[i]->nombre(), precio,
               listaProd[i]->caracteristicas(), preciooferta, cantidad,
               listaComercios[i], pos );
         }

         elementoSeleccionado = NULL;

         connect(btnAceptar,             SIGNAL(clicked()),             this,
SLOT(btnAceptarPulsado()));
         connect(btnCancelar,             SIGNAL(clicked()),             this,
SLOT(btnCancelarPulsado()));
         connect(lsvtablaProductos, SIGNAL(selectionChanged(QListViewItem *)),
this, SLOT(nuevaSeleccion(QListViewItem *)));
      }

      void ventanaCarroCompra::btnAceptarPulsado() {
         int i;

         if (elementoSeleccionado != NULL) {
            elementoSeleccionado->setText(4,txtCantidad->text());
         }

         QListViewItem *iter = lsvtablaProductos->firstChild();

         for (i=0; i<lsvtablaProductos->childCount(); i++) {
            CORBA_Long cant = strtol(iter->text(4),0,10);
            int pos = strtol(iter->text(6),0,10);

            if (cant!=listaCant[pos])
               carroActual-
>cambiarCantidad(listaProd[pos],listaComercios[pos],cant);


            iter = iter->nextSibling();
         }

         this->done(0);
      }

      void ventanaCarroCompra::btnCancelarPulsado() {
         this->done(0);
      }

      void ventanaCarroCompra::nuevaSeleccion(QListViewItem *nuevoSel) {
         if (elementoSeleccionado != NULL) {
            elementoSeleccionado->setText(4,txtCantidad->text());
         }

         txtCantidad->setText(nuevoSel->text(4));
         elementoSeleccionado = nuevoSel;
```

```
}

// Programa principal

int main(int narg, char *arg[]) {
   if (ConstantesCliente::inicializar(narg,arg)==1)
      return 1;

   QApplication centro(narg,arg);

   ventanaIdentificaSocio vIdenSoc;
   vIdenSoc.resize(400,300);
   centro.setMainWidget(&vIdenSoc);

   return vIdenSoc.exec();
}
```

Makefile para el centro virtual

```
CPPFLAGS = -I. -I/usr/local/pgsql/include

all: ServidorCentro ClienteCentro

ServidorCentro:         CentroComercial.o         CentroComercial_skel.o
ConstantesServidor.o    SocioImpl.o    CarroCompraImpl.o    FactoriaSocioImpl.o
ServidorCentro.o FactoriaComercioImpl.o ProductoImpl.o InternautaImpl.o
        g++ CentroComercial.o CentroComercial_skel.o ConstantesServidor.o
SocioImpl.o     CarroCompraImpl.o     FactoriaSocioImpl.o     ServidorCentro.o
FactoriaComercioImpl.o ProductoImpl.o InternautaImpl.o -o ServidorCentro -L
/usr/local/pgsql/lib -lOB -lCosNaming -lpq++ -lpq -lcrypt


ClienteCentro:  CentroComercial.o  ConstantesCliente.o  ClienteCentro.o
moc_ClienteCentro.o
        g++    CentroComercial.o    ConstantesCliente.o    ClienteCentro.o
moc_ClienteCentro.o -o ClienteCentro -lOB -lCosNaming -lqt

moc_ClienteCentro.o: ClienteCentro.h
        moc ClienteCentro.h -o moc_ClienteCentro.cpp
        g++    -I    /usr/include/qt    -c    moc_ClienteCentro.cpp    -o
moc_ClienteCentro.o
```

Apéndice C. Código fuente del compilador para acceso a BD

En este apéndice se incluirá tanto el código para la generación del compilador, como el código de la biblioteca que se ha utilizado en la documentación para realizar la explicación.

C.1 Código del compilador

idltodb.l

```
        #include "interfaz.h"
        #include <stl.h>
        #include "idltobd.tab.h"
        #include <string.h>
        #include <iostream.h>
        extern int linea;

digito [0-9]
```

```
letra [a-zA-Z]

%%

[ \t]+       ;
[\n]         linea++;
";"          return ';';
"{"          return '{';
"}"          return '}';
":"          return ':';
","          return ',';
"::"         return DOSDOSPUNTOS;

module       return MODULE;
interface    return INTERFACE;
attribute    return ATTRIBUTE;
readonly     return READONLY;
float        return FLOAT;
double       return DOUBLE;
long         return LONG;
short        return SHORT;
unsigned     return UNSIGNED;
char         return CHAR;
wchar        return WCHAR;
boolean            return BOOLEAN;
Object       return OBJECT;
string       return STRING;
wstring            return WSTRING;

{letra}({letra}|{digito})*{
                           yylval.cadena = new char[strlen(yytext)+1];
                           strcpy(yylval.cadena,yytext);
                           return IDENTIFIER;}

.            cout << "ERROR Lexico en linea " << linea << endl;

%%
```

idltodb.y

```
%{
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include "atributo.h"
#include "interfaz.h"
#include "tipo_attr.h"

int linea=1;
ofstream salidaSQL,salidaIDL;
char *nombreIDL;
list<interfaz *> interfaces;
char *strcopia(char *);
char *stranade(char *, char* );
void stranadedelante(char *&dest, char *orig);
char *strfinal(char *);
bool tienepunto(char *);
extern  void  generarClases(list<interfaz*>  *linter,  ofstream  &SQL,
ofstream &IDL, char *nombreIDL);

extern yyparse();
extern yylex();
extern FILE *yyin;
yyerror(char *s);
%}

%union
```

```
	{
		char* cadena;
		list<char *>* lcadena;
		atributo* atrib;
		list<atributo *>* latrib;
		bool boolean;
		tipo_attr tipoattr;
		interfaz* interf;
		list<interfaz*>* linterf;
	};

	%token MODULE, INTERFACE, ATTRIBUTE, READONLY, FLOAT, DOUBLE, LONG,
SHORT
	%token UNSIGNED, CHAR, WCHAR, BOOLEAN, OBJECT, STRING, WSTRING,
DOSDOSPUNTOS

	%token <cadena>	IDENTIFIER

	%type <atrib>	simple_declarator
	%type <latrib>		multiple_simple_declarator
	%type <cadena>		scoped_name
	%type <boolean>		readonly
	%type <tipoattr>	param_type_spec
	%type <tipoattr>	base_type_spec
	%type <tipoattr>	integer_type
	%type <tipoattr>	signed_int
	%type <tipoattr>	unsigned_int
	%type <tipoattr>	floating_pt_type
	%type <latrib>		attr_dcl
	%type <latrib>		export
	%type <latrib>		interface_body
	%type <interf>		interface_header
	%type <interf>		forward_dcl
	%type <interf>		interface_dcl
	%type <interf>		interface
	%type <linterf>		definition
	%type <linterf>		multiple_definition
	%type <linterf>		specification
	%type <linterf>		module
	%type <lcadena>		inheritance_spec
	%type <lcadena>		other_classes

	%%

	specification:	multiple_definition
		{

		// Convierte los nombres de las interfaces padres en referencias
a estas
		bool encontradoforward = false;
		list<interfaz *>::iterator it = $1->begin();
		while ((*it)!=0) {
			if ((*it)->nombrespadres!=0) {
				(*it)->padres = new list<interfaz *>;
				list<char *>::iterator itp = (*it)->nombrespadres-
>begin();
				while ((*itp)!=0) {
					encontradoforward = false;
					bool errorpadre=true;
					list<interfaz *>::iterator it2 = $1->begin();
					while ( (!encontradoforward && !(it2==it)) ||
(encontradoforward && (*it2)!=0) ) {
						if (tienepunto(*itp)) {
							if	(!strcmp(stranade((*it2)-
>modulo,(*it2)->nombre),(*itp))) {
								if ((*it2)->atributos==0)
									encontradoforward =
true;
```

```
                                                else {
                                                      (*it)->padres-
>push_back((*it2));
                                                      errorpadre=false;
                                                      break;
                                                }
                                          }
                                    }
                                    else {
                                          if              (!strcmp((*it2)-
>modulo,(*it)->modulo) && !strcmp((*it2)->nombre,(*itp))) {
                                                if ((*it2)->atributos==0)
                                                      encontradoforward  =
true;
                                                else {
                                                      (*it)->padres-
>push_back((*it2));
                                                      errorpadre=false;
                                                      break;
                                                }
                                          }
                                    }

                                    it2++;
                              }
                              if (errorpadre) {
                                    cerr << "La clase base '" << (*itp) <<
"' no existe" << endl;
                                    return -1;
                              }
                              itp++;
                        }
                  }
                  it++;
            }

            // Convierte los nombres de los tipos interfaces de los atributos
en referencias a estas
            encontradoforward = false;
            it = $1->begin();
            while ((*it)!=0) {
                  if ((*it)->atributos!=0) {
                        list<atributo  *>::iterator  ita  =  (*it)->atributos-
>begin();
                        while ((*ita)!=0) {
                           if ((*ita)->tipo.tipo==tipo_attr::tinterfaz) {
                               encontradoforward = false;
                               bool erroratrib=true;
                               list<interfaz *>::iterator it2 = $1->begin();
                               while ( (!encontradoforward && !(it2==it)) ||
(encontradoforward && (*it2)!=0) ) {
                                    if                  (tienepunto((*ita)-
>tipo.nombreinterfaz)) {
                                          if      (!strcmp(stranade((*it2)-
>modulo,(*it2)->nombre),(*ita)->tipo.nombreinterfaz)) {
                                                if ((*it2)->atributos==0)
                                                      encontradoforward  =
true;
                                                else {
                                                      (*ita)-
>tipo.inter=(*it2);
                                                      erroratrib=false;
                                                      break;
                                                }
                                          }
                                    }
                                    else {
```

```
                                                if              (!strcmp((*it2)-
>modulo,(*it)->modulo)  &&  !strcmp((*it2)->nombre,(*ita)->tipo.nombreinterfaz))
{
                                                      if ((*it2)->atributos==0)
                                                            encontradoforward  =
true;
                                                      else {
                                                            (*ita)-
>tipo.inter=(*it2);
                                                            erroratrib=false;
                                                            break;
                                                      }
                                                }
                                          }

                                          it2++;
                                    }
                                    if (erroratrib) {
                                          cerr   <<   "El   tipo   ’"   <<   (*ita)-
>tipo.nombreinterfaz << "’ del atributo ’" <<
                                                (*ita)->nombre   <<   "’   en   la
interfaz ’" << (*it)->nombre << "’ no existe" << endl;
                                          return -1;
                                    }
                              }
                              ita++;

                        }
                  }
                  it++;
            }

            // Elimina las interfaces definidas como forward
            it = $1->begin();
            while ((*it)!=0) {
                  if ((*it)->atributos==0)
                        it = $1->erase(it);
                  else it++;
            }

      /*    it = $1->begin();
            while ((*it)!=0) {
                  cout << (*it)->modulo << "." << (*it)->nombre << endl;
                  it++;
            }*/

            generarClases($1,salidaSQL,salidaIDL,nombreIDL);

            }
      ;

      multiple_definition:      definition
            { $$ = $1; }
                        | multiple_definition definition
            { $$ = new list<interfaz *>;
            list<interfaz *>::iterator it = $1->begin();
            while ((*it)!=0) {
                  $$->push_back(*it);
                  it++;
            }
            it = $2->begin();
            while ((*it)!=0) {
                  $$->push_back(*it);
                  it++;
            }

            }
      ;
```

```
	definition:  interface ';'
		{ $$ = new list<interfaz*>;
		$$->push_back($1); }
			| module ';'
		{ $$ = $1; }
	;

	module:			MODULE IDENTIFIER '{' multiple_definition '}'
		{ list<interfaz *>::iterator it = $4->begin();
		while ((*it)!=0) {

			if ((*it)->atributos==0) {
				// Se ha declarado como fordward
				list<interfaz *>::iterator it2 = it;
				it2++;
				while ((*it2)!=0) {
					if  (!strcmp((*it)->nombre,(*it2)->nombre)  &&
(*it2)->atributos!=0) break;
					it2++;
				}
				if ((*it2)==0) {
					cerr << "Clase Fordward '" << (*it)->nombre
<< "' no definida" << endl;
					return -1;
				}
			}
			else {
				// No es forward, hay que ver si se ha declarado dos
veces
				list<interfaz *>::iterator it2 = it;
				it2++;
				while ((*it2)!=0) {
					if (!strcmp((*it)->nombre,(*it2)->nombre)) {
						if ((*it2)->atributos==0)
							it2 = $4->erase(it2);
						else {
							cerr  <<  "Clase  '"  <<  (*it)-
>nombre << "' definida dos veces" << endl;
							return -1;
						}
					}
					it2++;
				}
			}
			stranadedelante((*it)->modulo,$2);
			it++;
		}
		$$ = $4;
		}

	;

	interface:   interface_dcl
		{$$ = $1;}
			| forward_dcl
		{$$ = $1;}
	;

	interface_dcl:		interface_header '{' interface_body '}'
		{ $1->atributos = $3;
		$$ = $1; }
	;

	forward_dcl: INTERFACE IDENTIFIER
		{ $$ = new interfaz($2); }
	;
```

```
	interface_header:   INTERFACE IDENTIFIER
		{ $$ = new interfaz($2); }
				| INTERFACE IDENTIFIER inheritance_spec
		{ $$ = new interfaz($2);
		$$->nombrespadres=$3; }
	;

	interface_body:     interface_body export
		{$$ = new list<atributo *>;
		list<atributo *>::iterator it = $1->begin();
		while ((*it)!=0) {
			$$->push_back(*it);
			it++;
		}
		it = $2->begin();
		while ((*it)!=0) {
			$$->push_back(*it);
			it++;
		}

		}

			|
		{$$ = new list<atributo *>; }
	;

	export:             attr_dcl ';'
		{ $$ = $1; }
	;

	inheritance_spec:   ':' scoped_name other_classes
		{ $3->push_back($2);
		$$ = $3; }
	;

	other_classes:      ',' scoped_name other_classes
		{ $3->push_back($2);
		$$ = $3; }
			|
		{ $$ = new list<char*>; }
	;


	scoped_name: IDENTIFIER
		{ $$ = strcopia($1); }
			| DOSDOSPUNTOS IDENTIFIER
		{ $$ = stranade("",$2);}
			| scoped_name DOSDOSPUNTOS IDENTIFIER
		{ $$ = stranade($1,$3); }
	;

	attr_dcl:    readonly   ATTRIBUTE   param_type_spec   simple_declarator
multiple_simple_declarator
		{$5->push_back($4);
		list<atributo *>::iterator it = $5->begin();
		while ((*it)!=0) {
			(*it)->tipo=$3;
			(*it)->readonly=$1;
			it++;
		}
		$$ = $5;
		}
	;

	multiple_simple_declarator:      ','                   simple_declarator
multiple_simple_declarator
		{$3->push_back($2);
		$$ = $3}
```

```
                              |
        {$$ = new list<atributo *>; }
;

readonly:    READONLY
        {$$ = true;}
             |
        {$$ = false;}
;

simple_declarator:  IDENTIFIER
        { $$ = new atributo($1); }
;

param_type_spec:    base_type_spec
        {$$ = $1;}
                    | string_type
        {$$.tipo = tipo_attr::tstring;}
                    | wide_string_type
        {$$.tipo = tipo_attr::twstring;}
                    | scoped_name
        {$$.tipo = tipo_attr::tinterfaz;
        $$.nombreinterfaz = strcopia($1);
        }
;

base_type_spec:     floating_pt_type
        {$$ = $1;}
              | integer_type
        {$$ = $1;}
              | char_type
        {$$.tipo = tipo_attr::tchar;}
              | wide_char_type
        {$$.tipo = tipo_attr::twchar;}
              | boolean_type
        {$$.tipo = tipo_attr::tbool;}
              | object_type
        {$$.tipo = tipo_attr::tobject;}
;

floating_pt_type:   FLOAT
        {$$.tipo = tipo_attr::tfloat;}
                    | DOUBLE
        {$$.tipo = tipo_attr::tdouble;}
                    | LONG DOUBLE
        {$$.tipo = tipo_attr::tlongdouble;}
;

integer_type:signed_int
        {$$ = $1;}
              | unsigned_int
        {$$ = $1;}
;

signed_int:  signed_short_int
        {$$.tipo = tipo_attr::tshort;}
              | signed_long_int
        {$$.tipo = tipo_attr::tlong;}
              | signed_longlong_int
        {$$.tipo = tipo_attr::tlonglong;}
;

signed_short_int:   SHORT
;

signed_long_int:    LONG
;
```

```
signed_longlong_int:      LONG LONG
;

unsigned_int:unsigned_short_int
       {$$.tipo = tipo_attr::tunsignedshort;}
              | unsigned_long_int
       {$$.tipo = tipo_attr::tunsignedlong;}
              | unsigned_longlong_int
       {$$.tipo = tipo_attr::tunsignedlonglong;}
;

unsigned_short_int: UNSIGNED SHORT
;

unsigned_long_int:  UNSIGNED LONG
;

unsigned_longlong_int:    UNSIGNED LONG LONG
;

char_type:   CHAR
;

wide_char_type:     WCHAR
;

boolean_type:BOOLEAN
;

object_type: OBJECT
;

string_type: STRING
;

wide_string_type:   WSTRING
;

%%

yyerror(char *s)
{
       cerr << s << " en linea: " << linea << '\n';
}

main()
{
       char nombre[80];

       printf("\nIntroduzca el nombre del fichero IDL de entrada: ");
       do {
              cin >> nombre;
              if (!(yyin=fopen(nombre,"r")))
                     cout << "Nombre incorrecto, ¿nombre?: ";
       } while (!yyin);

       nombreIDL = new char[strlen(nombre)+1];
       strcpy(nombreIDL,nombre);

       printf("\nIntroduzca el nombre del fichero SQL de salida: ");
       cin >> nombre;
       salidaSQL.open(nombre);

       printf("\nIntroduzca el nombre del fichero IDL de salida: ");
       cin >> nombre;
       salidaIDL.open(nombre);
```

```
	int ret;

	ret=yyparse();

	cout << "FIN DEL ANALISIS LEXICO Y SINTACTICO\n";
	salidaSQL.close();
	salidaIDL.close();

	return ret;
}

// Crea una copia del parametro y la devuelve
char *strcopia(char *arg) {
	char *ret = new char[strlen(arg)+1];
	strcpy(ret,arg);
	return ret;
}

// Anade los dos nombres introduciendo un punto en medio
char *stranade(char *dest, char* orig) {
	char *ret = new char[strlen(dest) + 1 + strlen(orig) + 1];
	strcpy(ret,dest);
	strcat(ret,".");
	strcat(ret,orig);
	return ret;
}

// Anade delante del destino el origen y un punto
void stranadedelante(char *&dest, char *orig) {
	if (dest[0]==0) {
		delete [] dest;
		char *ret = new char[strlen(orig) + 1];
		strcpy(ret,orig);
		dest = ret;
		return;
	}
	char *ret = new char[strlen(dest) + 1 + strlen(orig) + 1];
	strcpy(ret,orig);
	strcat(ret,".");
	strcat(ret,dest);
	delete [] dest;
	dest=ret;
}

// Devuelve la ultima parte dividiendo la cadena por los puntos
char *strfinal(char *arg) {
	int i;
	int principio=0, longi=0;
	for (i=0; i<strlen(arg); i++) {
		if (arg[i]=='.') {
			principio=i+1;
			longi=0;
		}
		else longi++;
	}
	char *ret = new char[longi+1];
	strcpy(ret,&arg[principio]);
	return ret;
}

// Comprueba si hay un punto en la cadena
bool tienepunto(char *arg) {
	int i;
	for (i=0; i<strlen(arg); i++) {
		if (arg[i]=='.') return true;
	}
	return false;
```

```
}
```

interfaz.h

```
#ifndef INTERFAZ_H
#define INTERFAZ_H

#include "atributo.h"
#include <stl.h>

class interfaz {
public:
   interfaz(char       *nom)       {nombre=new       char[strlen(nom)];
strcpy(nombre,nom);
                          modulo = new char[1]; strcpy(modulo,""); atributos =
0;
                              padres=0; nombrespadres=0;}
   char* nombre;
   char* modulo;
   list<interfaz *>* padres;
   list<char *>* nombrespadres;
   list<atributo *>* atributos;
};

#endif
```

atributo.h

```
#ifndef ATRIBUTO_H
#define ATRIBUTO_H

#include <string.h>
#include "tipo_attr.h"

class atributo {
public:
   atributo(char       *nom)       {nombre=new       char[strlen(nom)];
strcpy(nombre,nom);}
   char* nombre;
   tipo_attr tipo;
   bool readonly;
};


#endif
```

tipo_attr.h

```
#ifndef TIPO_ATTR_H
#define TIPO_ATTR_H

class interfaz;

struct tipo_attr {
   enum     {tfloat,     tdouble,     tlongdouble,     tshort,     tlong,
tlonglong,tunsignedshort, tunsignedlong,
      tunsignedlonglong,   tchar,   twchar,   tbool,   tobject,   tstring,
twstring, tinterfaz} tipo;

   char *nombreinterfaz;
   interfaz *inter;

};

#endif
```

generador.cpp

```
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include "interfaz.h"
#include <string.h>

void generarSQL(interfaz *inter, ofstream &SQL);
void generarIDL(interfaz *inter, ofstream &IDL);
void generarImpl(interfaz *inter);
void generarFactoryImpl(interfaz *inter);
void generarOA(interfaz *inter);
void generarIteratorImpl(interfaz *inter);
void generarSafe(interfaz *inter);
void generarServidor(list<interfaz*> *linter);
char *nombretabla(char *modulo, char *nombre, char *intermedio);
void imprimeTipoSQL(atributo* atr, ofstream &SQL);
void imprimeTipoIDL(atributo* atr, ofstream &IDL);
void imprimeTipoJava(atributo* atr, ofstream &SAL);
void imprimeObjetoJava(atributo* atr, ofstream &SAL);
void imprimeObjetoJava(tipo_attr tipo, ofstream &SAL);
void imprimeObtenerDatos(atributo* atr, ofstream &SAL,int i, char
*resultset);
void imprimeCastingJava(atributo* atr, ofstream &SAL, int nclave);
void imprimeNuloSQL(atributo* atr, ofstream &SAL);
void imprimeNuloJava(atributo* atr, ofstream &SAL);
void imprimeClave(interfaz *inter,ofstream &IMPL);
void imprimeGeneraClave(interfaz *inter,ofstream &IMPL);
void imprimeGeneraSQL(interfaz *inter,ofstream &IMPL);
void imprimeGeneraValoresInsertSQL(interfaz *inter,ofstream &IMPL);
void imprimeGeneraValoresNuevoObjeto(interfaz *inter,ofstream &IMPL);
void tratarherencia(interfaz *inter);


void generarClases(list<interfaz*> *linter, ofstream &SQL, ofstream
&IDL, char *nombreIDL) {
   list<interfaz *>::iterator it = linter->begin();
   while ((*it)!=0) {
      SQL << "DROP TABLE " << nombretabla((*it)->modulo,(*it)-
>nombre,"_") << ";" << endl;
      it++;
   }

   SQL << endl;

   IDL << "#include \"" << nombreIDL << "\"" << endl;

   it = linter->begin();
   while ((*it)!=0) {
      tratarherencia(*it);
      generarSQL((*it),SQL);
      generarIDL((*it),IDL);
      generarImpl(*it);
      generarFactoryImpl(*it);
      generarOA(*it);
      generarIteratorImpl(*it);
      generarSafe(*it);
      it++;
   }
   generarServidor(linter);

}

void tratarherencia(interfaz *inter) {
   if (inter->nombrespadres!=0) {
      list<interfaz *>::iterator it = inter->padres->begin();

      while ((*it)!=0) {
```

```
               list<atributo *>::iterator it2 = (*it)->atributos->begin();
               while ((*it2)!=0) {
                  inter->atributos->push_back(*it2);
                  it2++;
               }
               it++;
            }
         }
      }

      // Sustituye los '.' por intermedio y une el modulo al nombre con otro
intermedio
      char *nombretabla(char *modulo, char *nombre, char *intermedio) {
         char  *ret  =  new  char[strlen(modulo)  +  strlen(intermedio)  +
strlen(nombre) + 1];
         int i;

         if (modulo[0]!=0) {
            char *aux = new char[strlen(modulo) + 1];
            int principiomod=0;
            strcpy(aux,modulo);
            ret[0]=0;
            for (i=0; i<=strlen(modulo); i++) {
               if (modulo[i]=='.' || modulo[i]==0) {
                  aux[i]=0;
                  strcat(ret,&aux[principiomod]);
                  strcat(ret,intermedio);
                  principiomod=i+1;
               }
            }
            strcat(ret,nombre);

            delete [] aux;
         }
         else strcpy(ret,nombre);

         return ret;
      }

      void generarSQL(interfaz *inter, ofstream &SQL) {
         cout << "Generando la sentencia SQL para la clase " << inter->nombre
<< endl;
         SQL << endl;
         SQL  <<  "CREATE  TABLE  "  <<  nombretabla(inter->modulo,inter-
>nombre,"_") << " (" << endl;

         list<atributo *>::iterator it = inter->atributos->begin();
         while ((*it)!=0) {
            SQL << "\t" << (*it)->nombre;

            // Tipo del atributo
            imprimeTipoSQL((*it),SQL);

            SQL << "," << endl;
            it++;
         }

         SQL  <<  "\tCONSTRAINT  pk_"  <<  nombretabla(inter->modulo,inter-
>nombre,"_") << " PRIMARY KEY (";
         it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               // Es un atributo primario
               if (primeratr)
                  primeratr=false;
               else
                  SQL << ", ";
```

```
               SQL << (*it)->nombre;
            }
            it++;
         }

         SQL << ")" << endl << "\t);" << endl;

      }

      void imprimeTipoSQL(atributo* atr, ofstream &SQL) {
            switch (atr->tipo.tipo) {
               case tipo_attr::tfloat:
                  SQL << " float4";
                  break;
               case tipo_attr::tdouble:
               case tipo_attr::tlongdouble:
                  SQL << " float8";
                  break;
               case tipo_attr::tshort:
               case tipo_attr::tunsignedshort:
                  SQL << " int2";
                  break;
               case tipo_attr::tlong:
               case tipo_attr::tunsignedlong:
                  SQL << " int4";
                  break;
               case tipo_attr::tlonglong:
               case tipo_attr::tunsignedlonglong:
                  SQL << " int8";
                  break;
               case tipo_attr::tchar:
               case tipo_attr::twchar:
                  SQL << " char";
                  break;
               case tipo_attr::tbool:
                  SQL << " bool";
                  break;
               case tipo_attr::tobject:
                  cerr << "No se admite aun el tipo object" << endl;
                  exit(-1);
                  break;
               case tipo_attr::tstring:
               case tipo_attr::twstring:
                  SQL << " text";
                  break;
               case tipo_attr::tinterfaz:
                  if (atr->readonly) {
                     cerr << "No se admite como clave primaria una atributo
que no sea de un tipo basico" << endl;
                     exit(-1);
                  }
                  bool primeraclave=true;

                  list<atributo *>::iterator it = atr->tipo.inter->atributos-
>begin();
                  while ((*it)!=0) {
                     if ((*it)->readonly) {
                        if (primeraclave)
                           primeraclave=false;
                        else
                           SQL << "," << endl << "\t" << atr->nombre;

                        SQL << "_" << (*it)->nombre;
                        imprimeTipoSQL((*it),SQL);
                        SQL << " CONSTRAINT fk_" << atr->nombre << "_" <<
(*it)->nombre <<
```

```
                        "  REFERENCES  "  <<  nombretabla(atr->tipo.inter-
>modulo,atr->tipo.inter->nombre,"_")
                        << " (" << (*it)->nombre << ")";
                     }
                     it++;
                  }
                  break;
            }

      }

      void imprimeTipoIDL(atributo* atr, ofstream &IDL) {
            switch (atr->tipo.tipo) {
               case tipo_attr::tfloat:
                  IDL << " float";
                  break;
               case tipo_attr::tdouble:
                  IDL << " double";
                  break;
               case tipo_attr::tlongdouble:
                  IDL << " long double";
                  break;
               case tipo_attr::tshort:
                  IDL << " short";
                  break;
               case tipo_attr::tunsignedshort:
                  IDL << " unsigned short";
                  break;
               case tipo_attr::tlong:
                  IDL << " long";
                  break;
               case tipo_attr::tunsignedlong:
                  IDL << " unsigned long";
                  break;
               case tipo_attr::tlonglong:
                  IDL << " long long";
                  break;
               case tipo_attr::tunsignedlonglong:
                  IDL << " unsigned long long";
                  break;
               case tipo_attr::tchar:
                  IDL << " char";
                  break;
               case tipo_attr::twchar:
                  IDL << " wchar";
                  break;
               case tipo_attr::tbool:
                  IDL << " boolean";
                  break;
               case tipo_attr::tobject:
                  cerr << "No se admite aun el tipo object" << endl;
                  exit(-1);
                  break;
               case tipo_attr::tstring:
                  IDL << " string";
                  break;
               case tipo_attr::twstring:
                  IDL << " wstring";
                  break;
            }
      }

      void generarIDL(interfaz *inter, ofstream &IDL) {
         cout << "Generando interfaces IDL para la clase " << inter->nombre <<
endl;
         int nmodulos = 0;
         int i;
```

```
         if (inter->modulo[0]!=0) {
            char *aux = new char[strlen(inter->modulo)+1];
            int j;
            for (i=0, j=0; i<=strlen(inter->modulo); i++, j++) {
               if (inter->modulo[i]==’.’ || inter->modulo[i]==0) {
                  aux[j]=0;
                  IDL << "module " << aux << " {" << endl;
                  j=-1;
                  nmodulos++;
               }
               else
                  aux[j]=inter->modulo[i];
            }
            delete [] aux;
         }

         // Se genera la interfaz nombreIterator
         IDL << "interface " << inter->nombre << "Iterator {" << endl;

         IDL << "\t" << nombretabla(inter->modulo,inter->nombre,"::") << "
next();" << endl;

         IDL << "};" << endl;


         // Se genera la interfaz nombreFactory
         IDL << "interface " << inter->nombre << "Factory {" << endl;

         IDL << "\t" << nombretabla(inter->modulo,inter->nombre,"::") << "
getObject(";

         list<atributo *>::iterator it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               // Es un atributo primario
               if (primeratr)
                  primeratr=false;
               else
                  IDL << ", ";

               IDL << "in";
               imprimeTipoIDL((*it),IDL);
               IDL << " " << (*it)->nombre;
            }
            it++;
         }

         IDL << ");" << endl;

         IDL << "\tvoid remove(";

         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               // Es un atributo primario
               if (primeratr)
                  primeratr=false;
               else
                  IDL << ", ";

               IDL << "in";
               imprimeTipoIDL((*it),IDL);
               IDL << " " << (*it)->nombre;
            }
            it++;
         }
```

```
         IDL << ");" << endl;

         IDL << "\tboolean exists(";

         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               // Es un atributo primario
               if (primeratr)
                  primeratr=false;
               else
                  IDL << ", ";

               IDL << "in";
               imprimeTipoIDL((*it),IDL);
               IDL << " " << (*it)->nombre;
            }
            it++;
         }

         IDL << ");" << endl;

         IDL << "\t" << inter->nombre << "Iterator list(in string condition);"
<< endl;

         IDL << "};" << endl;

         for (i=0; i<nmodulos; i++) {
            IDL << "};" << endl;
         }

      }

      void generarImpl(interfaz *inter) {
         cout << "Generando Impl para la clase " << inter->nombre << endl;
         char *nombreClase = new char[strlen(inter->nombre) + strlen("Impl") +
1];

         strcpy(nombreClase,inter->nombre);
         strcat(nombreClase,"Impl");

         char *nombreFichero = new char[strlen(nombreClase) + strlen(".java")
+ 1];

         strcpy(nombreFichero,nombreClase);
         strcat(nombreFichero,".java");

         ofstream IMPL;
         IMPL.open(nombreFichero);

         delete [] nombreFichero;

         if (inter->modulo[0]!=0)
            IMPL << "package " << inter->modulo << ";" << endl;

         IMPL << "public class " << nombreClase << " extends _" << nombreClase
<< "Base" << endl;
         IMPL << "\timplements dbcollection.ChangeSensible {" << endl;

         list<atributo *>::iterator it = inter->atributos->begin();
         while ((*it)!=0) {
            IMPL << "\tprivate";
            imprimeTipoJava((*it),IMPL);
            IMPL << " " << (*it)->nombre << ";" << endl;
            it++;
         }
```

```
         IMPL << "\tprivate boolean changed;" << endl;
         IMPL << "\tpublic boolean changed() { return changed; }" << endl;

         IMPL << "\tpublic " << nombreClase << "(";
         it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if (primeratr)
               primeratr=false;
            else
               IMPL << ",";

            imprimeTipoJava((*it),IMPL);
            IMPL << " " << (*it)->nombre;
            it++;
         }

         IMPL << ") {" << endl;
         IMPL << "\t\tchanged = false;" << endl;

         it = inter->atributos->begin();
         while ((*it)!=0) {
            IMPL << "\t\tthis." << (*it)->nombre << " = " << (*it)->nombre <<
";" << endl;
            it++;
         }

         IMPL << "\t}" << endl << endl;

         it = inter->atributos->begin();
         while ((*it)!=0) {
            IMPL << "\tpublic";
            imprimeTipoJava((*it),IMPL);
            IMPL << " " << (*it)->nombre << "() { return " << (*it)->nombre <<
"; }" << endl;

            if (!(*it)->readonly) {
               IMPL << "\tpublic void " << (*it)->nombre << "(";
               imprimeTipoJava((*it),IMPL);
               IMPL << " valor) {" << endl;
               IMPL << "\t\tchanged = true;" << endl;
               IMPL << "\t\t" << (*it)->nombre << " = valor;" << endl;
               IMPL << "\t}" << endl;
            }

            IMPL << endl;

            it++;
         }



         IMPL << "}";

         IMPL.close();

         delete [] nombreClase;
      }


      void imprimeTipoJava(atributo* atr, ofstream &SAL) {
            switch (atr->tipo.tipo) {
               case tipo_attr::tfloat:
                  SAL << " float";
                  break;
               case tipo_attr::tdouble:
               case tipo_attr::tlongdouble:
```

```
                SAL << " double";
                break;
            case tipo_attr::tshort:
            case tipo_attr::tunsignedshort:
                SAL << " short";
                break;
            case tipo_attr::tlong:
            case tipo_attr::tunsignedlong:
                SAL << " int";
                break;
            case tipo_attr::tlonglong:
            case tipo_attr::tunsignedlonglong:
                SAL << " long";
                break;
            case tipo_attr::tchar:
            case tipo_attr::twchar:
                SAL << " char";
                break;
            case tipo_attr::tbool:
                SAL << " boolean";
                break;
            case tipo_attr::tobject:
                cerr << "No se admite aun el tipo object" << endl;
                exit(-1);
                break;
            case tipo_attr::tstring:
            case tipo_attr::twstring:
                SAL << " String";
                break;
            case tipo_attr::tinterfaz:
                SAL << " " << atr->tipo.nombreinterfaz;
                break;
        }
    }

    void imprimeObjetoJava(atributo* atr, ofstream &SAL) {
          switch (atr->tipo.tipo) {
            case tipo_attr::tfloat:
                SAL << " new Float(" << atr->nombre << ")";
                break;
            case tipo_attr::tdouble:
            case tipo_attr::tlongdouble:
                SAL << " new Double(" << atr->nombre << ")";
                break;
            case tipo_attr::tshort:
            case tipo_attr::tunsignedshort:
                SAL << " new Short(" << atr->nombre << ")";
                break;
            case tipo_attr::tlong:
            case tipo_attr::tunsignedlong:
                SAL << " new Integer(" << atr->nombre << ")";
                break;
            case tipo_attr::tlonglong:
            case tipo_attr::tunsignedlonglong:
                SAL << " new Long(" << atr->nombre << ")";
                break;
            case tipo_attr::tchar:
            case tipo_attr::twchar:
                SAL << " new Character(" << atr->nombre << ")";
                break;
            case tipo_attr::tbool:
                SAL << " new Boolean(" << atr->nombre << ")";
                break;
            case tipo_attr::tobject:
                cerr << "No se admite aun el tipo object" << endl;
                exit(-1);
                break;
            case tipo_attr::tstring:
```

```
            case tipo_attr::twstring:
               SAL << " " << atr->nombre;
               break;
         }
   }


   void imprimeObjetoJava(tipo_attr tipo, ofstream &SAL) {
         switch (tipo.tipo) {
            case tipo_attr::tfloat:
               SAL << " new Float(" ;
               break;
            case tipo_attr::tdouble:
            case tipo_attr::tlongdouble:
               SAL << " new Double(" ;
               break;
            case tipo_attr::tshort:
            case tipo_attr::tunsignedshort:
               SAL << " new Short(" ;
               break;
            case tipo_attr::tlong:
            case tipo_attr::tunsignedlong:
               SAL << " new Integer(" ;
               break;
            case tipo_attr::tlonglong:
            case tipo_attr::tunsignedlonglong:
               SAL << " new Long(" ;
               break;
            case tipo_attr::tchar:
            case tipo_attr::twchar:
               SAL << " new Character(" ;
               break;
            case tipo_attr::tbool:
               SAL << " new Boolean(" ;
               break;
            case tipo_attr::tobject:
               cerr << "No se admite aun el tipo object" << endl;
               exit(-1);
               break;
            case tipo_attr::tstring:
            case tipo_attr::twstring:
               SAL << "( " ;
               break;
         }
   }

   void imprimeNuloSQL(atributo* atr, ofstream &SAL) {
         switch (atr->tipo.tipo) {
            case tipo_attr::tfloat:
            case tipo_attr::tdouble:
            case tipo_attr::tlongdouble:
               SAL << "0.0";
               break;
            case tipo_attr::tshort:
            case tipo_attr::tunsignedshort:
            case tipo_attr::tlong:
            case tipo_attr::tunsignedlong:
            case tipo_attr::tlonglong:
            case tipo_attr::tunsignedlonglong:
               SAL << "0";
               break;
            case tipo_attr::tbool:
               SAL << "false";
               break;
            case tipo_attr::tobject:
               cerr << "No se admite aun el tipo object" << endl;
               exit(-1);
               break;
```

```
               case tipo_attr::tchar:
               case tipo_attr::twchar:
               case tipo_attr::tstring:
               case tipo_attr::twstring:
                  break;
               case tipo_attr::tinterfaz:
                  bool primeraclave=true;

                  list<atributo *>::iterator it = atr->tipo.inter->atributos-
>begin();
                  while ((*it)!=0) {
                     if ((*it)->readonly) {
                        if (primeraclave)
                           primeraclave=false;
                        else
                           SAL << "' , '";

                        imprimeNuloSQL((*it),SAL);
                     }
                     it++;
                  }
                  break;

            }
      }


      void generarFactoryImpl(interfaz *inter) {
         cout << "Generando FactoryImpl para la clase " << inter->nombre <<
endl;

         char    *nombreClase    =    new    char[strlen(inter->nombre)    +
strlen("FactoryImpl") + 1];

         strcpy(nombreClase,inter->nombre);
         strcat(nombreClase,"FactoryImpl");

         char *nombreFichero = new char[strlen(nombreClase) + strlen(".java")
+ 1];

         strcpy(nombreFichero,nombreClase);
         strcat(nombreFichero,".java");

         ofstream IMPL;
         IMPL.open(nombreFichero);

         delete [] nombreFichero;

         if (inter->modulo[0]!=0)
            IMPL << "package " << inter->modulo << ";" << endl;

         IMPL << "import java.util.Vector;" << endl << endl;
         IMPL << "public class " << nombreClase << " extends _" << nombreClase
<< "Base {" << endl;
         IMPL << "\tprivate org.omg.CORBA.ORB orb;" << endl;
         IMPL << "\tprivate " << inter->nombre << "OA oa;" << endl;
         IMPL << "\tprivate dbcollection.AbstractFactory af;" << endl;

         IMPL << endl << "\tpublic " << nombreClase << "(org.omg.CORBA.ORB
orb) {" << endl;
         IMPL << "\t\tsuper();" << endl;
         IMPL << "\t\tthis.orb = orb;" << endl;
         IMPL << "\t\taf = new dbcollection.AbstractFactory(orb, oa = new " <<
inter->nombre << "OA());" << endl;
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic " << inter->nombre << " getObject(";
```

```
         imprimeClave(inter,IMPL);

         IMPL << ") {" << endl;

         IMPL << "\t\tVector key = new Vector();" << endl;

         imprimeGeneraClave(inter,IMPL);
         IMPL << "\t\treturn (" << inter->nombre;
         IMPL << ")(af.getObject(key));" << endl;

         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic void remove(";

         imprimeClave(inter,IMPL);

         IMPL << ") {" << endl;

         IMPL << "\t\tVector key = new Vector();" << endl;

         imprimeGeneraClave(inter,IMPL);
         IMPL << "\t\taf.remove(key);" << endl;

         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic boolean exists(";

         imprimeClave(inter,IMPL);

         IMPL << ") {" << endl;

         IMPL << "\t\tVector key = new Vector();" << endl;

         imprimeGeneraClave(inter,IMPL);
         IMPL << "\t\treturn af.exists(key);" << endl;

         IMPL << "\t}" << endl << endl;


         IMPL << "\tpublic " << inter->nombre << "Iterator list(String cond)
{" << endl;

         IMPL    <<    "\t\treturn    new    "    <<    inter->nombre    <<
"IteratorImpl(af,oa,cond,this,orb);" << endl;

         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic void finalize() {" << endl;
         IMPL << "\t\taf.finalize();" << endl;
         IMPL << "\t}" << endl;


         IMPL << "}";

         IMPL.close();

         delete [] nombreClase;

      }

      void imprimeClave(interfaz *inter,ofstream &IMPL) {
         list<atributo *>::iterator it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               if (primeratr)
                  primeratr=false;
               else
```

```
                  IMPL << ",";

               imprimeTipoJava((*it),IMPL);
               IMPL << " " << (*it)->nombre;
            }
            it++;
         }
      }


      void imprimeGeneraClave(interfaz *inter,ofstream &IMPL) {
         list<atributo *>::iterator it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
               IMPL << "\t\tkey.add(";
               imprimeObjetoJava((*it),IMPL);
               IMPL << ");" << endl;
            }
            it++;
         }
      }


      void generarOA(interfaz *inter) {
         int i;
         cout << "Generando OA para la clase " << inter->nombre << endl;

         char *nombreClase = new char[strlen(inter->nombre) + strlen("OA") +
1];

         strcpy(nombreClase,inter->nombre);
         strcat(nombreClase,"OA");

         char *nombreFichero = new char[strlen(nombreClase) + strlen(".java")
+ 1];

         strcpy(nombreFichero,nombreClase);
         strcat(nombreFichero,".java");

         ofstream IMPL;
         IMPL.open(nombreFichero);

         delete [] nombreFichero;

         if (inter->modulo[0]!=0)
            IMPL << "package " << inter->modulo << ";" << endl;

         IMPL << "import java.util.Vector;" << endl;
         IMPL << "import java.util.Enumeration;" << endl;
         IMPL << "import java.sql.*;" << endl << endl;
         IMPL   <<   "public   class   "   <<   nombreClase   <<   "   implements
dbcollection.dbObjectAdaptor {" << endl;

         IMPL << "\tprivate Statement localSt;" << endl;
         IMPL << "\tprivate Statement pSt;" << endl;
         IMPL << "\tprivate ResultSet pRs;" << endl;
         IMPL << "\tprivate boolean hasMore = false;" << endl << endl;

         IMPL << "\tpublic " << nombreClase << "() {" << endl;
         IMPL << "\t\ttry {" << endl;
         IMPL                 <<                "\t\t\tlocalSt                =
dbcollection.servidorBD.baseDatos.createStatement();" << endl;
         IMPL                  <<                  "\t\t\tpSt                 =
dbcollection.servidorBD.baseDatos.createStatement();" << endl;
         IMPL << "\t\t} catch (SQLException ex) {" << endl;
         IMPL  <<  "\t\t\tthrow  new  RuntimeException(\"Fallo  en  la  base  de
datos\");" << endl;
         IMPL << "\t\t}" << endl;
```

```
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic Object retrieve(Vector key) {" << endl;
         IMPL << "\t\ttry {" << endl;
         IMPL << "\t\t\tResultSet rs = localSt.executeQuery(\"SELECT * FROM ";
         IMPL << nombretabla(inter->modulo,inter->nombre,"_") << " WHERE ";
         imprimeGeneraSQL(inter,IMPL);
         IMPL << "\");" << endl;
         IMPL << "\t\t\tif (!rs.next()) {" << endl;

         list<atributo *>::iterator it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
               IMPL << "\t\t\t\tif (";
               if     ((*it)->tipo.tipo==tipo_attr::tstring     ||     (*it)-
>tipo.tipo==tipo_attr::twstring) {
                  IMPL << "!((String)key.get(0)).equals(\"\")) {" << endl;
               }
               else {
                  imprimeCastingJava((*it),IMPL,0);
                  IMPL << " != ";
                  imprimeNuloJava((*it),IMPL);
                  IMPL << ") {" << endl;
               }
               break;
            }
            it++;
         }

         IMPL << "\t\t\t\t\tlocalSt.executeUpdate(\"INSERT INTO ";
         IMPL << nombretabla(inter->modulo,inter->nombre,"_") << " VALUES (";
         imprimeGeneraValoresInsertSQL(inter,IMPL);
         IMPL << ")\");" << endl;
         IMPL << "\t\t\t\t\treturn new " << inter->nombre << "Impl(";
         imprimeGeneraValoresNuevoObjeto(inter,IMPL);
         IMPL << ");" << endl;
         IMPL << "\t\t\t\t} else" << endl;
         IMPL << "\t\t\t\t\treturn null;" << endl;
         IMPL << "\t\t\t} else {" << endl;
         i=1;
         it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if ((*it)->tipo.tipo==tipo_attr::tinterfaz) {

               if (primeratr) {
                  primeratr=false;
                  IMPL << "\t\t\t\tVector key2 = new Vector();" << endl;
               }
               else
                  IMPL << "\t\t\t\tkey2 = new Vector();" << endl;

               int j=i;
               list<atributo *>::iterator it2 = (*it)->tipo.inter->atributos-
>begin();
               while ((*it2)!=0) {
                  if ((*it2)->readonly) {
                     IMPL << "\t\t\t\tkey2.add(";
                     imprimeObjetoJava((*it2)->tipo, IMPL);
                     imprimeObtenerDatos((*it2),IMPL,i,"rs");
                     IMPL << "));" << endl;
                     i++;
                  }
                  it2++;
               }
               IMPL << "\t\t\t\t" << (*it)->tipo.nombreinterfaz << " p" << j
<< " = (";
               IMPL << (*it)->tipo.nombreinterfaz <<")(new ";
```

```
               IMPL << (*it)->tipo.nombreinterfaz << "OA()).retrieve(key2);"
<< endl;
            }
            i++;
            it++;
         }


         IMPL << "\t\t\t\treturn new " << inter->nombre << "Impl(";
         i=1;
         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if (primeratr)
               primeratr=false;
            else
               IMPL << ", ";

            if ((*it)->tipo.tipo==tipo_attr::tinterfaz) {
               IMPL << "p" << i;
               list<atributo *>::iterator it2 = (*it)->tipo.inter->atributos-
>begin();
               while ((*it2)!=0) {
                  if ((*it2)->readonly)
                     i++;
                  it2++;
               }
               i--;
            }
            else
               imprimeObtenerDatos((*it),IMPL,i,"rs");

            i++;
            it++;
         }
         IMPL << ");" << endl;
         IMPL << "\t\t\t}" << endl;
         IMPL << "\t\t} catch (SQLException ex) {" << endl;
         IMPL << "\t\t\treturn null;" << endl;
         IMPL << "\t\t}" << endl;
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic void store(Object o) {" << endl;
         // Si no hay atributos que no sean primarios no hay que actualizar
nada
         it = inter->atributos->begin();
         while ((*it)!=0) {
            if (!(*it)->readonly) {
               break;
            }
            it++;
         }
        if ((*it)!=0) {
         IMPL << "\t\t" << inter->nombre << " obj = (" << inter->nombre << ")
o;" << endl;
         IMPL << "\t\ttry {" << endl;
         IMPL << "\t\t\tlocalSt.executeUpdate(\"UPDATE ";
         IMPL << nombretabla(inter->modulo,inter->nombre,"_") << " SET ";
         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if (!(*it)->readonly) {
               if ((*it)->tipo.tipo==tipo_attr::tinterfaz) {
                  list<atributo   *>::iterator   it2   =   (*it)->tipo.inter-
>atributos->begin();
                  while ((*it2)!=0) {
                     if ((*it2)->readonly) {
                        if (primeratr)
```

```
                        primeratr=false;
                     else
                        IMPL << ", ";
                     IMPL << (*it)->nombre << "_" << (*it2)->nombre << " =
'\" + ((obj.";
                     IMPL << (*it)->nombre << "()==null) ? \"\" : ";
                     imprimeObjetoJava((*it2)->tipo,IMPL);
                     IMPL << "obj.";
                     IMPL  << (*it)->nombre << "()." << (*it2)->nombre <<
"()).toString()) + \"'\";
                  }
                  it2++;
               }
            }
            else {
               if (primeratr)
                  primeratr=false;
               else
                  IMPL << ", ";
               IMPL << (*it)->nombre << " = '\" + obj." << (*it)->nombre <<
"() + \"'\";
            }
         }
         it++;
      }
      IMPL << " WHERE ";
      it = inter->atributos->begin();
      primeratr=true;
      while ((*it)!=0) {
         if ((*it)->readonly) {
            if (primeratr)
               primeratr=false;
            else
               IMPL << " AND ";

            IMPL << (*it)->nombre << " = '\" + obj." << (*it)->nombre <<
"() + \"'\";
         }
         it++;
      }
      IMPL << "\");" << endl;
      IMPL << "\t\t} catch (SQLException ex) {" << endl;
      IMPL << "\t\t\treturn;" << endl;
      IMPL << "\t\t}" << endl;
     }
      IMPL << "\t}" << endl << endl;

      IMPL << "\tpublic void remove(Vector key) {" << endl;
      IMPL << "\t\ttry {" << endl;
      IMPL << "\t\t\tlocalSt.executeUpdate(\"DELETE FROM ";
      IMPL << nombretabla(inter->modulo,inter->nombre,"_") << " WHERE ";
      imprimeGeneraSQL(inter,IMPL);
      IMPL << "\");" << endl;
      IMPL << "\t\t} catch (SQLException ex) {" << endl;
      IMPL << "\t\t\treturn;" << endl;
      IMPL << "\t\t}" << endl;
      IMPL << "\t}" << endl << endl;

      IMPL << "\tpublic boolean exists(Vector key) {" << endl;
      IMPL << "\t\ttry {" << endl;
      IMPL << "\t\t\tResultSet rs = localSt.executeQuery(\"SELECT * FROM ";
      IMPL << nombretabla(inter->modulo,inter->nombre,"_") << " WHERE ";
      imprimeGeneraSQL(inter,IMPL);
      IMPL << "\");" << endl;
      IMPL << "\t\t\treturn rs.next();" << endl;
      IMPL << "\t\t} catch (SQLException ex) {" << endl;
      IMPL << "\t\t\treturn false;" << endl;
      IMPL << "\t\t}" << endl;
```

```
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic boolean hasMoreElements() {" << endl;
         IMPL << "\t\treturn hasMore;" << endl;
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic Enumeration list(String cond) {" << endl;
         IMPL << "\t\ttry {" << endl;
         IMPL << "\t\t\tString queryStr = \"SELECT * FROM ";
         IMPL  <<  nombretabla(inter->modulo,inter->nombre,"_")  <<  "\";"  <<
endl;
         IMPL << "\t\t\tif (!cond.equals(\"\")) {" << endl;
         IMPL << "\t\t\t\tqueryStr = queryStr + \" WHERE \" + cond;" << endl;
         IMPL << "\t\t\t}" << endl;
         IMPL << "\t\t\tpRs = pSt.executeQuery(queryStr);" << endl;
         IMPL << "\t\t\thasMore = pRs.next();" << endl;
         IMPL << "\t\t\treturn (Enumeration) this;" << endl;
         IMPL << "\t\t} catch (SQLException ex) {" << endl;
         IMPL  <<  "\t\t\tthrow  new  RuntimeException(\"Error  en  la  Base  de
Datos\");" << endl;
         IMPL << "\t\t}" << endl;
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic Object nextElement() {" << endl;
         IMPL      <<      "\t\tif      (!hasMore)      {      throw      new
java.util.NoSuchElementException(); }" << endl;
         IMPL << "\t\ttry {" << endl;
         i=1;
         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if ((*it)->tipo.tipo==tipo_attr::tinterfaz) {
               if (primeratr) {
                  primeratr=false;
                  IMPL << "\t\t\tVector key2 = new Vector();" << endl;
               }
               else
                  IMPL << "\t\t\tkey2 = new Vector();" << endl;
               int j=i;
               list<atributo *>::iterator it2 = (*it)->tipo.inter->atributos-
>begin();
               while ((*it2)!=0) {
                  if ((*it2)->readonly) {
                     IMPL << "\t\t\tkey2.add(";
                     imprimeObjetoJava((*it2)->tipo, IMPL);
                     imprimeObtenerDatos((*it2),IMPL,i,"pRs");
                     IMPL << "));" << endl;
                     i++;
                  }
                  it2++;
               }

               IMPL << "\t\t\t" << (*it)->tipo.nombreinterfaz << " p" << j <<
" = (";
               IMPL << (*it)->tipo.nombreinterfaz <<")(new ";
               IMPL  <<  (*it)->tipo.nombreinterfaz  <<  "OA()).retrieve(key2);"
<< endl;
            }
            i++;
            it++;
         }

         IMPL << "\t\t\tObject tmpObj = new " << inter->nombre << "Impl(";
         i=1;
         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if (primeratr)
```

```
               primeratr=false;
            else
               IMPL << ", ";

            if ((*it)->tipo.tipo==tipo_attr::tinterfaz) {
               IMPL << "p" << i;
               list<atributo *>::iterator it2 = (*it)->tipo.inter->atributos-
>begin();
               while ((*it2)!=0) {
                  if ((*it2)->readonly)
                     i++;
                  it2++;
               }
               i--;

            }
            else
               imprimeObtenerDatos((*it),IMPL,i,"pRs");

            i++;
            it++;
         }
         IMPL << ");" << endl;
         IMPL << "\t\t\thasMore = pRs.next();" << endl;
         IMPL << "\t\t\treturn tmpObj;" << endl;
         IMPL << "\t\t} catch (SQLException ex) {" << endl;
         IMPL  <<  "\t\t\tthrow  new  RuntimeException(\"Error  en  la  Base  de
Datos\");" << endl;
         IMPL << "\t\t}" << endl;
         IMPL << "\t}" << endl << endl;


         IMPL << "}";

         IMPL.close();

         delete [] nombreClase;

      }

      void imprimeGeneraSQL(interfaz *inter,ofstream &IMPL) {
         int nclave = 0;
         list<atributo *>::iterator it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               if (primeratr)
                  primeratr=false;
               else
                  IMPL << " AND ";

               IMPL  <<  (*it)->nombre  <<  "  =  '\"  +  key.get("  <<  nclave  <<
").toString() + \"'";
               nclave++;
            }
            it++;
         }
      }

      void imprimeGeneraValoresInsertSQL(interfaz *inter,ofstream &IMPL) {
         int nclave = 0;
         list<atributo *>::iterator it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if (primeratr)
               primeratr=false;
            else
               IMPL << " , ";
```

```
      if ((*it)->readonly) {
         IMPL << "'\" + key.get(" << nclave << ").toString() + \"'";
         nclave++;
      }
      else {
         IMPL << "'";
         imprimeNuloSQL((*it),IMPL);
         IMPL << "'";
      }
      it++;
   }
}

void imprimeGeneraValoresNuevoObjeto(interfaz *inter,ofstream &IMPL) {
   int nclave = 0;
   list<atributo *>::iterator it = inter->atributos->begin();
   bool primeratr=true;
   while ((*it)!=0) {
      if (primeratr)
         primeratr=false;
      else
         IMPL << " , ";
      if ((*it)->readonly) {
         imprimeCastingJava((*it),IMPL,nclave);
         nclave++;
      }
      else {
         imprimeNuloJava((*it),IMPL);
      }
      it++;
   }
}


void imprimeNuloJava(atributo* atr, ofstream &SAL) {
      switch (atr->tipo.tipo) {
         case tipo_attr::tfloat:
            SAL << "(float)0.0";
            break;
         case tipo_attr::tdouble:
         case tipo_attr::tlongdouble:
            SAL << "0.0";
            break;
         case tipo_attr::tshort:
         case tipo_attr::tunsignedshort:
            SAL << "(short)0";
            break;
         case tipo_attr::tlong:
         case tipo_attr::tunsignedlong:
         case tipo_attr::tlonglong:
         case tipo_attr::tunsignedlonglong:
            SAL << "0";
            break;
         case tipo_attr::tchar:
         case tipo_attr::twchar:
            SAL << "(char)0";
            break;
         case tipo_attr::tbool:
            SAL << "false";
            break;
         case tipo_attr::tobject:
            cerr << "No se admite aun el tipo object" << endl;
            exit(-1);
            break;
         case tipo_attr::tstring:
         case tipo_attr::twstring:
            SAL << "\"\"";
            break;
```

```
               case tipo_attr::tinterfaz:
                  SAL << "null";
                  break;

            }
      }


      void imprimeCastingJava(atributo* atr, ofstream &SAL, int nclave) {
            switch (atr->tipo.tipo) {
               case tipo_attr::tfloat:
                  SAL << "((Float)(key.get(" << nclave << "))).floatValue()";
                  break;
               case tipo_attr::tdouble:
               case tipo_attr::tlongdouble:
                  SAL     <<     "((Double)(key.get("     <<     nclave     <<
"))).doubleValue()";
                  break;
               case tipo_attr::tshort:
               case tipo_attr::tunsignedshort:
                  SAL << "((Short)(key.get(" << nclave << "))).shortValue()";
                  break;
               case tipo_attr::tlong:
               case tipo_attr::tunsignedlong:
                  SAL << "((Integer)(key.get(" << nclave << "))).intValue()";
                  break;
               case tipo_attr::tlonglong:
               case tipo_attr::tunsignedlonglong:
                  SAL << "((Long)(key.get(" << nclave << "))).longValue()";
                  break;
               case tipo_attr::tchar:
               case tipo_attr::twchar:
                  SAL    <<    "((Character)(key.get("    <<    nclave    <<
"))).charValue()";
                  break;
               case tipo_attr::tbool:
                  SAL     <<     "((Boolean)(key.get("     <<     nclave     <<
"))).booleanValue()";
                  break;
               case tipo_attr::tobject:
                  cerr << "No se admite aun el tipo object" << endl;
                  exit(-1);
                  break;
               case tipo_attr::tstring:
               case tipo_attr::twstring:
                  SAL << "(String)(key.get(" << nclave << "))";
                  break;
               case tipo_attr::tinterfaz:
                  SAL << "(" << atr->tipo.nombreinterfaz << ")";
                  break;
            }
      }


      void  imprimeObtenerDatos(atributo*  atr,  ofstream  &SAL,  int  i,  char
*resultset){
            switch (atr->tipo.tipo) {
               case tipo_attr::tfloat:
                  SAL << resultset << ".getFloat(" << i << ")";
                  break;
               case tipo_attr::tdouble:
               case tipo_attr::tlongdouble:
                  SAL << resultset << ".getDouble(" << i << ")";
                  break;
               case tipo_attr::tshort:
               case tipo_attr::tunsignedshort:
                  SAL << resultset << ".getShort(" << i << ")";
                  break;
```

```
               case tipo_attr::tlong:
               case tipo_attr::tunsignedlong:
                  SAL << resultset << ".getInt(" << i << ")";
                  break;
               case tipo_attr::tlonglong:
               case tipo_attr::tunsignedlonglong:
                  SAL << resultset << ".getLong(" << i << ")";
                  break;
               case tipo_attr::tchar:
               case tipo_attr::twchar:
                  SAL << resultset << ".getString(" << i << ").charAt(0)";
                  break;
               case tipo_attr::tbool:
                  SAL << resultset << ".getBoolean(" << i << ")";
                  break;
               case tipo_attr::tobject:
                  cerr << "No se admite aun el tipo object" << endl;
                  exit(-1);
                  break;
               case tipo_attr::tstring:
               case tipo_attr::twstring:
                  SAL << resultset << ".getString(" << i << ")";
                  break;
               case tipo_attr::tinterfaz:
                  SAL << "null";
                  break;
            }
      }

      void generarIteratorImpl(interfaz *inter) {

         cout << "Generando IteratorImpl para la clase " << inter->nombre <<
endl;

         char    *nombreClase    =    new    char[strlen(inter->nombre)    +
strlen("IteratorImpl") + 1];

         strcpy(nombreClase,inter->nombre);
         strcat(nombreClase,"IteratorImpl");

         char *nombreFichero = new char[strlen(nombreClase) + strlen(".java")
+ 1];

         strcpy(nombreFichero,nombreClase);
         strcat(nombreFichero,".java");

         ofstream IMPL;
         IMPL.open(nombreFichero);

         delete [] nombreFichero;

         if (inter->modulo[0]!=0)
            IMPL << "package " << inter->modulo << ";" << endl;

         IMPL << "import dbcollection.AbstractFactory;" << endl;
         IMPL << "import java.util.Vector;" << endl;
         IMPL << "import java.util.Enumeration;" << endl << endl;
         IMPL << "public class " << nombreClase << " extends _" << nombreClase
<< "Base {" << endl;
         IMPL << "\tprivate String cond;" << endl;
         IMPL << "\tprivate AbstractFactory af;" << endl;
         IMPL << "\tprivate Enumeration e;" << endl;
         IMPL << "\tprivate " << inter->nombre << "Factory fac;" << endl;
         IMPL << "\tprivate org.omg.CORBA.ORB orb;" << endl << endl;

         IMPL << "\tpublic " << nombreClase << "(AbstractFactory af, " <<
inter->nombre << "OA oa, ";
```

```
         IMPL  <<  "String  cond,  "  <<  inter->nombre  <<  "Factory  fac,
org.omg.CORBA.ORB orb) {" << endl;
         IMPL << "\t\te = oa.list(cond);" << endl;
         IMPL << "\t\tthis.af = af;" << endl;
         IMPL << "\t\tthis.cond = cond;" << endl;
         IMPL << "\t\tthis.fac = fac;" << endl;
         IMPL << "\t\tthis.orb = orb;" << endl;
         IMPL << "\t}" << endl << endl;

         IMPL << "\tpublic " << inter->nombre << " next() {" << endl;
         IMPL << "\t\tif (!e.hasMoreElements()) { orb.disconnect(this); return
null; }" << endl;
         IMPL << "\t\tObject tmpObj = e.nextElement();" << endl;
         list<atributo *>::iterator it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
               IMPL << "\t\t";
               imprimeTipoJava((*it),IMPL);
               IMPL  <<  "  "  <<  (*it)->nombre  <<  "  =  ((" <<  inter->nombre  <<
")tmpObj)." << (*it)->nombre;
               IMPL << "();" << endl;;
            }
            it++;
         }
         IMPL << "\t\tVector key = new Vector();" << endl;
         imprimeGeneraClave(inter,IMPL);
         IMPL << "\t\treturn (" << inter->nombre;
         IMPL << ")(af.cacheObject(key,tmpObj));" << endl;
         IMPL << "\t}" << endl;

         IMPL << "}";

         IMPL.close();

         delete [] nombreClase;

      }

      void generarSafe(interfaz *inter) {
         cout << "Generando Safe para la clase " << inter->nombre << endl;

         char *nombreClase = new char[strlen(inter->nombre) + strlen("Safe") +
1];

         strcpy(nombreClase,"Safe");
         strcat(nombreClase,inter->nombre);

         char  *nombreFichero  =  new  char[strlen(nombreClase)  +  strlen(".java")
+ 1];

         strcpy(nombreFichero,nombreClase);
         strcat(nombreFichero,".java");

         ofstream IMPL;
         IMPL.open(nombreFichero);

         delete [] nombreFichero;

         if (inter->modulo[0]!=0)
            IMPL << "package " << inter->modulo << ";" << endl;


         IMPL << "public class " << nombreClase <<" {" << endl;
         IMPL << "\tpublic " << inter->nombre << " ref;" << endl;
         IMPL << "\tprivate " << inter->nombre << "Factory factory;" << endl;
         list<atributo *>::iterator it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
```

```
               IMPL << "\tprivate";
               imprimeTipoJava((*it),IMPL);
               IMPL << " " << (*it)->nombre << ";" << endl;
            }
            it++;
         }
         IMPL << endl;

         IMPL << "\tpublic " << nombreClase << "(" << inter->nombre <<
"Factory factory,";
         it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if (primeratr)
               primeratr=false;
            else
               IMPL << ",";

            imprimeTipoJava((*it),IMPL);
            IMPL << " " << (*it)->nombre;
            it++;
         }
         IMPL << ") {" << endl;
         IMPL << "\t\tthis.factory = factory;" << endl;
         it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
               IMPL << "\t\tthis." << (*it)->nombre << " = " << (*it)->nombre
<< ";" << endl;
            }
            it++;
         }
         IMPL << "\t\tref = factory.getObject(";
         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               if (primeratr)
                  primeratr=false;
               else
                  IMPL << ", ";

               IMPL << (*it)->nombre;
            }
            it++;
         }
         IMPL << ");" << endl;
         it = inter->atributos->begin();
         while ((*it)!=0) {
            if (!(*it)->readonly) {
               IMPL << "\t\t" << (*it)->nombre << "(" << (*it)->nombre << ");"
<< endl;
            }
            it++;
         }
         IMPL << "\t}" << endl << endl;

         // Si no hay atributos que no sean primarios no hay que generar el
segundo constructor
         it = inter->atributos->begin();
         while ((*it)!=0) {
            if (!(*it)->readonly) {
               break;
            }
            it++;
         }
        if ((*it)!=0) {
```

```
         IMPL << "\tpublic " << nombreClase << "(" << inter->nombre << "Factory factory,";
         it = inter->atributos->begin();
         bool primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               if (primeratr)
                  primeratr=false;
               else
                  IMPL << ",";

               imprimeTipoJava((*it),IMPL);
               IMPL << " " << (*it)->nombre;
            }
            it++;
         }
         IMPL << ") {" << endl;
         IMPL << "\t\tthis.factory = factory;" << endl;
         it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
               IMPL << "\t\tthis." << (*it)->nombre << " = " << (*it)->nombre << ";" << endl;
            }
            it++;
         }
         IMPL << "\t\tref = factory.getObject(";
         it = inter->atributos->begin();
         primeratr=true;
         while ((*it)!=0) {
            if ((*it)->readonly) {
               if (primeratr)
                  primeratr=false;
               else
                  IMPL << ", ";

               IMPL << (*it)->nombre;
            }
            it++;
         }
         IMPL << ");" << endl;
         IMPL << "\t}" << endl << endl;
        }

         IMPL << "\tpublic " << nombreClase << "(" << inter->nombre << "Factory factory, ";
         IMPL << inter->nombre << " ref) {" << endl;
         IMPL << "\t\tthis.factory = factory;" << endl;
         IMPL << "\t\tthis.ref = ref;" << endl;
         it = inter->atributos->begin();
         while ((*it)!=0) {
            if ((*it)->readonly) {
               IMPL << "\t\tthis." << (*it)->nombre << " = " << (*it)->nombre << "();" << endl;
            }
            it++;
         }
         IMPL << "\t}" << endl << endl;

         it = inter->atributos->begin();
         while ((*it)!=0) {
            IMPL << "\tpublic";
            imprimeTipoJava((*it),IMPL);
            IMPL << " " << (*it)->nombre << "() {" << endl;
            IMPL << "\t\twhile (true) {" << endl;
            IMPL << "\t\t\ttry {" << endl;
            IMPL << "\t\t\t\treturn ref." << (*it)->nombre << "();" << endl;
```

```
            IMPL << "\t\t\t} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {" <<
endl;
            IMPL << "\t\t\t\tref = factory.getObject(";
            list<atributo *>::iterator it2 = inter->atributos->begin();
            primeratr=true;
            while ((*it2)!=0) {
               if ((*it2)->readonly) {
                  if (primeratr)
                     primeratr=false;
                  else
                     IMPL << ", ";

                  IMPL << (*it2)->nombre;
               }
               it2++;
            }
            IMPL << ");" << endl;
            IMPL << "\t\t\t}" << endl;
            IMPL << "\t\t}" << endl;
            IMPL << "\t}" << endl << endl;

            if (!(*it)->readonly) {
               IMPL << "\tpublic void " << (*it)->nombre << "(";
               imprimeTipoJava((*it),IMPL);
               IMPL << " value) {" << endl;
               IMPL << "\t\twhile (true) {" << endl;
               IMPL << "\t\t\ttry {" << endl;
               IMPL << "\t\t\t\tref." << (*it)->nombre << "(value);" << endl;
               IMPL << "\t\t\t\treturn;" << endl;
               IMPL << "\t\t\t} catch (org.omg.CORBA.OBJECT_NOT_EXIST ex) {"
<< endl;
               IMPL << "\t\t\t\tref = factory.getObject(";
               it2 = inter->atributos->begin();
               primeratr=true;
               while ((*it2)!=0) {
                  if ((*it2)->readonly) {
                     if (primeratr)
                        primeratr=false;
                     else
                        IMPL << ", ";

                     IMPL << (*it2)->nombre;
                  }
                  it2++;
               }
               IMPL << ");" << endl;
               IMPL << "\t\t\t\t" << (*it)->nombre << "(value);" << endl;
               IMPL << "\t\t\t}" << endl;
               IMPL << "\t\t}" << endl;
               IMPL << "\t}" << endl << endl;
            }
            it++;
         }



         IMPL << "}";

         IMPL.close();

         delete [] nombreClase;


      }


      void generarServidor(list<interfaz*> *linter) {
         int i;
```

```
          cout << "Generando Servidor" << endl;

          char *nombreClase = new char[strlen("Servidor") + 1];

          strcpy(nombreClase,"Servidor");

          char *nombreFichero = new char[strlen(nombreClase) + strlen(".java")
+ 1];

          strcpy(nombreFichero,nombreClase);
          strcat(nombreFichero,".java");

          ofstream IMPL;
          IMPL.open(nombreFichero);

          delete [] nombreFichero;

          list<interfaz *>::iterator it = linter->begin();
          interfaz *inter1 = *it;

          if (inter1->modulo[0]!=0)
             IMPL << "package " << inter1->modulo << ";" << endl;

          IMPL << "import org.omg.CORBA.*;" << endl;
          IMPL << "import java.io.*;" << endl;
          IMPL << "import org.omg.CosNaming.*;" << endl << endl;

          IMPL << "public class Servidor {" << endl;
          IMPL << "\tpublic static void main(String args[]) {" << endl;
          IMPL << "\t\tORB orb = ORB.init(args,null);" << endl;
          IMPL << "\t\tNamingContext rootContext=null;" << endl;
          IMPL << "\t\ttry {" << endl;
          IMPL       <<       "\t\t\torg.omg.CORBA.Object       objRef       =
orb.resolve_initial_references(\"NameService\");" << endl;
          IMPL << "\t\t\trootContext = NamingContextHelper.narrow(objRef);" <<
endl;
          IMPL << "\t\t} catch (org.omg.CORBA.ORBPackage.InvalidName ex) {" <<
endl;
          IMPL  <<  "\t\t\tSystem.err.println(\"Error,  no  se  encuentra  el
servicio de nombres\");" << endl;
          IMPL << "\t\t\tSystem.exit(-1);" << endl;
          IMPL << "\t\t}" << endl;
          IMPL << "\t\ttry {" << endl;
          IMPL << "\t\t\tNameComponent modContext[] = new NameComponent[1];" <<
endl;
          IMPL << "\t\t\tNamingContext servidores = rootContext;" << endl;
          int principiomod=0;
          char *copiamod = new char[strlen(inter1->modulo)];
          strcpy (copiamod,inter1->modulo);
          for (i=0; i<=strlen(inter1->modulo); i++) {
             if (inter1->modulo[i]=='.' || inter1->modulo[i]==0) {
                copiamod[i]=0;

                IMPL  <<  "\t\t\tmodContext[0]  =  new  NameComponent(\""  <<
&copiamod[principiomod] << "\", \"\");" << endl;
                IMPL << "\t\t\ttry {" << endl;
                IMPL            <<            "\t\t\t\tservidores            =
NamingContextHelper.narrow(servidores.resolve(modContext));" << endl;
                IMPL              <<              "\t\t\t}              catch
(org.omg.CosNaming.NamingContextPackage.NotFound ex) {" << endl;
                IMPL            <<            "\t\t\t\tservidores            =
servidores.bind_new_context(modContext);" << endl;
                IMPL << "\t\t\t}" << endl;

                principiomod=i+1;
             }
          }
```

```
         while ((*it)!=0) {
            IMPL << "\t\t\t" << (*it)->nombre << "FactoryImpl fact" << (*it)-
>nombre << " = new ";
            IMPL << (*it)->nombre << "FactoryImpl(orb);" << endl;
            IMPL << "\t\t\torb.connect(fact" << (*it)->nombre << ");" << endl;
            IMPL  <<  "\t\t\tmodContext[0]  =  new  NameComponent(\""  <<  (*it)-
>nombre;
            IMPL << "FactoryImpl\", \"\");" << endl;
            IMPL  <<  "\t\t\tservidores.rebind(modContext,  fact"  <<  (*it)-
>nombre << ");" << endl;
            it++;
         }

         IMPL  <<  "\t\t\tSystem.out.println(\"Servidor   a  la  escucha\");"  <<
endl;
         IMPL  <<  "\t\t\tSystem.out.println(\"Pulse  ENTER  para  finalizar\");"
<< endl;
         IMPL << "\t\t\ttry {" << endl;
         IMPL << "\t\t\t\tBufferedReader teclado;" << endl;
         IMPL     <<     "\t\t\t\tteclado     =     new     BufferedReader(new
InputStreamReader(System.in));" << endl;
         IMPL << "\t\t\t\tString linea=teclado.readLine();" << endl;
         IMPL << "\t\t\t} catch (IOException ex) {}" << endl;
         it = linter->begin();
         while ((*it)!=0) {
            IMPL << "\t\t\tfact" << (*it)->nombre << ".finalize();" << endl;
            it++;
         }
         IMPL << "\t\t} catch (org.omg.CORBA.UserException ex) {" << endl;
         IMPL    <<    "\t\t\tSystem.err.println(\"Excepcion    no    esperada,
finalizacion anormal\");" << endl;
         IMPL << "\t\t\tex.printStackTrace();" << endl;
         IMPL << "\t\t}" << endl;
         IMPL << "\t}" << endl;

         IMPL << "}";

         IMPL.close();

         delete [] nombreClase;

      }
```

Makefile para el compilador

```
all: idltobd

idltobd.tab.c: idltobd.y interfaz.h atributo.h tipo_attr.h
	bison -d idltobd.y

lex.yy.c: idltobd.l idltobd.y interfaz.h atributo.h tipo_attr.h
	flex idltobd.l

lex.yy.o: lex.yy.c
	g++ -c lex.yy.c -o lex.yy.o

idltobd.tab.o: idltobd.tab.c
	g++ -c idltobd.tab.c -o idltobd.tab.o

idltobd: idltobd.tab.o lex.yy.o generador.o
	g++ lex.yy.o idltobd.tab.o generador.o -o idltobd –lfl
```

C.2 Código de la biblioteca

biblioteca.ddl

```
module javi {
module biblioteca {
       interface autor;

       interface libro {
              readonly attribute string nombre;
              attribute autor autorLibro;
              attribute long ano;
       };

       interface editorial {
              readonly attribute string nombre;
              attribute string direccion;
       };

       interface autor {
              readonly attribute string nombre;
              readonly attribute string apellidos;
              attribute boolean vivo;
              attribute short edad;
       };

};
};
```

Cliente.java

```
package javi.biblioteca;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.io.*;

public class Cliente {

   public static autorFactory autorFac=null;
   public static editorialFactory editFac=null;
   public static libroFactory libroFac=null;

   public static void main(String args[]) {
     try {
      ORB orb = ORB.init(args,null);

      org.omg.CORBA.Object                    obj                    =
orb.resolve_initial_references("NameService");

      NamingContext root_context = NamingContextHelper.narrow(obj);

      if (root_context==null) {
        System.out.println("Error: Servicio de nombres no encontrado");
      }

      NameComponent nombre[] = new NameComponent[1];
      nombre[0]=new NameComponent("javi","");
      NamingContext servidores;

      try {
        servidores                                                    =
NamingContextHelper.narrow(root_context.resolve(nombre));
      } catch (org.omg.CosNaming.NamingContextPackage.NotFound ex) {
        System.out.println("Error: Lance el servidor de la biblioteca");
```

```
              return;
            }

            nombre[0]=new NameComponent("biblioteca","");

            try {
              servidores                                                    =
NamingContextHelper.narrow(servidores.resolve(nombre));
            } catch (org.omg.CosNaming.NamingContextPackage.NotFound ex) {
              System.out.println("Error: Lance el servidor de la biblioteca");
              return;
            }

            org.omg.CORBA.Object editFacObj;

            try {
                nombre[0]=new NameComponent("editorialFactoryImpl","");
                editFacObj=servidores.resolve(nombre);
                editFac=editorialFactoryHelper.narrow(editFacObj);
            } catch (Exception ex) {
                 System.out.println("Error:   Lance   el   servidor   de   la
biblioteca");
            }

            org.omg.CORBA.Object autorFacObj;
            try {
                nombre[0]=new NameComponent("autorFactoryImpl","");
                autorFacObj=servidores.resolve(nombre);
                autorFac=autorFactoryHelper.narrow(autorFacObj);
            } catch (Exception ex) {
                 System.out.println("Error:   Lance   el   servidor   de   la
biblioteca");
            }

            org.omg.CORBA.Object libroFacObj;

            try {
                nombre[0]=new NameComponent("libroFactoryImpl","");
                libroFacObj=servidores.resolve(nombre);
                libroFac=libroFactoryHelper.narrow(libroFacObj);
            } catch (Exception ex) {
                 System.out.println("Error:   Lance   el   servidor   de   la
biblioteca");
            }

            BufferedReader      teclado      =      new      BufferedReader(new
InputStreamReader(System.in));;

            while (true) {
               System.out.print("Comando (h ayuda): ");
               String linea = null;
               try {
                  linea = teclado.readLine();
               } catch (IOException ex) {
                  continue;
               }
               if (linea==null) continue;

               if (linea.equalsIgnoreCase("h")) {
                  System.out.println("Comando\taccion");
                  System.out.println("nl\tnuevo libro");
                  System.out.println("ll\tlistar libros");
                  System.out.println("ol\tobtener libro");
                  System.out.println("ne\tnueva editorial");
                  System.out.println("le\tlistar editoriales");
                  System.out.println("oe\tobtener editorial");
                  System.out.println("na\tnuevo autor");
                  System.out.println("la\tlistar autores");
```

```
                    System.out.println("oa\tobtener autor");
                    System.out.println("q\tquitar");
                    System.out.println();
                    continue;
                 }
                 if (linea.equalsIgnoreCase("q")) break;

                 if (linea.equalsIgnoreCase("nl")) {
                    System.out.print("Introduzca el nombre del libro: ");
                    while (true) {
                       try {
                          linea = teclado.readLine();
                          break;
                       } catch (IOException ex) {
                          continue;
                       }
                    }
                    if (!libroFac.exists(linea)) {
                       libroFac.getObject(linea);
                    }
                    else
                       System.out.println("El libro ya existe");

                    continue;
                 }

                 if (linea.equalsIgnoreCase("ll")) {
                    libroIterator it = libroFac.list("");

System.out.println("Nombre\tnombreAutor\tapellidosautor\tano");
                    Safelibro l;
                    libro auxl;
                    while ((auxl=it.next()) != null) {
                       l=new Safelibro(libroFac,auxl);
                       System.out.print(l.nombre());
                       System.out.print("\t");
                       if (l.autorLibro()!=null) {
                          System.out.print(l.autorLibro().nombre());
                          System.out.print("\t");
                          System.out.print(l.autorLibro().apellidos());
                          System.out.print("\t");
                       }
                       else
                          System.out.print("\t\t");
                       System.out.print(l.ano());
                       System.out.println();
                    }
                    continue;
                 }

                 if (linea.equalsIgnoreCase("ol")) {
                    System.out.print("Introduzca el nombre del libro: ");
                    while (true) {
                       try {
                          linea = teclado.readLine();
                          break;
                       } catch (IOException ex) {
                          continue;
                       }
                    }
                    if (libroFac.exists(linea)) {
                       tratarLibro(new Safelibro(libroFac,linea));
                    }
                    else
                       System.out.println("El libro no existe");

                    continue;
                 }
```

```
if (linea.equalsIgnoreCase("ne")) {
   System.out.print("Introduzca el nombre de la editorial: ");
   while (true) {
      try {
         linea = teclado.readLine();
         break;
      } catch (IOException ex) {
         continue;
      }
   }
   if (!editFac.exists(linea)) {
      editFac.getObject(linea);
   }
   else
      System.out.println("La editorial ya existe");

   continue;
}

if (linea.equalsIgnoreCase("le")) {
   editorialIterator it = editFac.list("");
   System.out.println("Nombre\tdireccion");
   Safeeditorial e;
   editorial auxe;
   while ((auxe=it.next()) != null) {
      e=new Safeeditorial(editFac,auxe);
      System.out.print(e.nombre());
      System.out.print("\t");
      System.out.print(e.direccion());
      System.out.println();
   }
   continue;
}

if (linea.equalsIgnoreCase("oe")) {
   System.out.print("Introduzca el nombre de la editorial: ");
   while (true) {
      try {
         linea = teclado.readLine();
         break;
      } catch (IOException ex) {
         continue;
      }
   }
   if (editFac.exists(linea)) {
      tratarEditorial(new Safeeditorial(editFac,linea));
   }
   else
      System.out.println("La editorial no existe");

   continue;
}

if (linea.equalsIgnoreCase("na")) {
   System.out.print("Introduzca el nombre del autor: ");
   while (true) {
      try {
         linea = teclado.readLine();
         break;
      } catch (IOException ex) {
         continue;
      }
   }
   System.out.print("Introduzca los apellidos del autor: ");
   String linea2;
   while (true) {
      try {
```

```
                    linea2 = teclado.readLine();
                    break;
                } catch (IOException ex) {
                    continue;
                }
            }

            if (!autorFac.exists(linea,linea2)) {
                autorFac.getObject(linea,linea2);
            }
            else
                System.out.println("El autor ya existe");

            continue;
        }

        if (linea.equalsIgnoreCase("la")) {
            autorIterator it = autorFac.list("");
            System.out.println("Nombre\tapellidos\tvivo\tedad");
            Safeautor a;
            autor auxa;
            while ((auxa=it.next()) != null) {
                a=new Safeautor(autorFac,auxa);
                System.out.print(a.nombre());
                System.out.print("\t");
                System.out.print(a.apellidos());
                System.out.print("\t");
                System.out.print(a.vivo());
                System.out.print("\t");
                System.out.print(a.edad());
                System.out.println();
            }
            continue;
        }

        if (linea.equalsIgnoreCase("oa")) {
            System.out.print("Introduzca el nombre del autor: ");
            while (true) {
                try {
                    linea = teclado.readLine();
                    break;
                } catch (IOException ex) {
                    continue;
                }
            }
            System.out.print("Introduzca los apellidos del autor: ");
            String linea2;
            while (true) {
                try {
                    linea2 = teclado.readLine();
                    break;
                } catch (IOException ex) {
                    continue;
                }
            }

            if (autorFac.exists(linea,linea2)) {
                tratarAutor(new Safeautor(autorFac,linea,linea2));
            }
            else
                System.out.println("La editorial no existe");

            continue;
        }
     }

    }catch (Exception ex) {
        ex.printStackTrace();
```

```
            }


          }

          public static void tratarLibro(Safelibro l) {
             BufferedReader      teclado      =      new      BufferedReader(new
InputStreamReader(System.in));;
             while (true) {
                System.out.print("Comando para el libro (h ayuda): ");
                String linea = null;
                try {
                   linea = teclado.readLine();
                } catch (IOException ex) {
                   continue;
                }
                if (linea==null) continue;

                if (linea.equalsIgnoreCase("h")) {
                   System.out.println("Comando\taccion");
                   System.out.println("d\tdatos");
                   System.out.println("q\tvolver");
                   System.out.println("ano\tmodificar el ano");
                   System.out.println("autor\tmodificar el autor");
                   System.out.println("b\tborrar");
                   continue;
                }

                if (linea.equalsIgnoreCase("q")) return;

                if (linea.equalsIgnoreCase("d")) {

System.out.println("Nombre\tnombreAutor\tapellidosautor\tano");
                   System.out.print(l.nombre());
                   System.out.print("\t");
                   if (l.autorLibro()!=null) {
                      System.out.print(l.autorLibro().nombre());
                      System.out.print("\t");
                      System.out.print(l.autorLibro().apellidos());
                      System.out.print("\t");
                   }
                   else
                      System.out.print("\t\t");
                   System.out.print(l.ano());
                   System.out.println();
                   continue;
                }

                if (linea.equalsIgnoreCase("ano")) {
                   System.out.print("Introduzca el ano: ");
                   while (true) {
                      try {
                         linea = teclado.readLine();
                         break;
                      } catch (IOException ex) {
                         continue;
                      }
                   }
                   l.ano(Integer.parseInt(linea));
                   System.out.println("Ano modificado");
                   continue;
                }

                if (linea.equalsIgnoreCase("autor")) {
                   System.out.print("Introduzca el nombre del autor: ");
                   while (true) {
                      try {
                         linea = teclado.readLine();
```

```
                        break;
                     } catch (IOException ex) {
                        continue;
                     }
                  }
                  System.out.print("Introduzca los apellidos del autor: ");
                  String linea2;
                  while (true) {
                     try {
                        linea2 = teclado.readLine();
                        break;
                     } catch (IOException ex) {
                        continue;
                     }
                  }
                  if (autorFac.exists(linea,linea2)) {
                     l.autorLibro(autorFac.getObject(linea,linea2));
                     System.out.println("Autor modificado");
                  }
                  else
                     System.out.println("El autor no existe");
               }

               if (linea.equalsIgnoreCase("b")) {

                  libroFac.remove(l.nombre());
                  System.out.println("libro Borrado");
                  break;
               }
            }
         }

         public static void tratarEditorial(Safeeditorial e) {
            BufferedReader      teclado      =      new      BufferedReader(new
InputStreamReader(System.in));;
            while (true) {
               System.out.print("Comando para la editorial (h ayuda): ");
               String linea = null;
               try {
                  linea = teclado.readLine();
               } catch (IOException ex) {
                  continue;
               }
               if (linea==null) continue;

               if (linea.equalsIgnoreCase("h")) {
                  System.out.println("Comando\taccion");
                  System.out.println("d\tdatos");
                  System.out.println("q\tvolver");
                  System.out.println("direccion\tmodificar la direccion");
                  System.out.println("b\tborrar");
                  continue;
               }

               if (linea.equalsIgnoreCase("q")) return;

               if (linea.equalsIgnoreCase("d")) {
                  System.out.println("Nombre\tdireccion");
                  System.out.print(e.nombre());
                  System.out.print("\t");
                  System.out.print(e.direccion());
                  System.out.println();
                  continue;
               }

               if (linea.equalsIgnoreCase("direccion")) {
                  System.out.print("Introduzca la direccion: ");
                  while (true) {
```

```
                    try {
                       linea = teclado.readLine();
                       break;
                    } catch (IOException ex) {
                       continue;
                    }
                 }
                 e.direccion(linea);
                 System.out.println("Direccion modificada");
                 continue;
              }

              if (linea.equalsIgnoreCase("b")) {

                 editFac.remove(e.nombre());
                 System.out.println("editorial Borrada");
                 break;
              }
           }
        }

        public static void tratarAutor(Safeautor a) {
           BufferedReader      teclado      =      new      BufferedReader(new
InputStreamReader(System.in));;
           while (true) {
              System.out.print("Comando para el autor (h ayuda): ");
              String linea = null;
              try {
                 linea = teclado.readLine();
              } catch (IOException ex) {
                 continue;
              }
              if (linea==null) continue;

              if (linea.equalsIgnoreCase("h")) {
                 System.out.println("Comando\taccion");
                 System.out.println("d\tdatos");
                 System.out.println("q\tvolver");
                 System.out.println("vivo\tmodificar la condicion de vivo");
                 System.out.println("edad\tmodificar la edad");
                 System.out.println("b\tborrar");
                 continue;
              }

              if (linea.equalsIgnoreCase("q")) return;

              if (linea.equalsIgnoreCase("d")) {
                 System.out.println("Nombre\tapellidos\tvivo\tedad");
                 System.out.print(a.nombre());
                 System.out.print("\t");
                 System.out.print(a.apellidos());
                 System.out.print("\t");
                 System.out.print(a.vivo());
                 System.out.print("\t");
                 System.out.print(a.edad());
                 System.out.println();
                 continue;
              }

              if (linea.equalsIgnoreCase("edad")) {
                 System.out.print("Introduzca la edad: ");
                 while (true) {
                    try {
                       linea = teclado.readLine();
                       break;
                    } catch (IOException ex) {
                       continue;
                    }
```

```
                }
                a.edad((short)Integer.parseInt(linea));
                System.out.println("Edad modificada");
                continue;
            }

            if (linea.equalsIgnoreCase("vivo")) {
                a.vivo(!a.vivo());
                System.out.println("Condicion de vivo modificada");
                continue;
            }

            if (linea.equalsIgnoreCase("b")) {
                autorFac.remove(a.nombre(),a.apellidos());
                System.out.println("autor Borrado");
                break;
            }
        }
    }


}
```

BIBLIOGRAFÍA

[APA98] Apache. *Apache Web Server Documentation,* 1998. http://www.apache.org

[BJM94] P. Bohnhoff, R. Janssen, and R. Martín. *Fundamentos Cliente/Servidor*. IBM, 1994

[DR95] Charles Donnelly, Richard Stallman. *Bison, version 1.25*. 1995

[GHJ+94] Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA. 1994

[JAW99] Jamie Jaworski. *Java 1.2*. Prentice Hall. 1999

[KAM97] Klaus Bergner, Andreas Raush and Marc Sihling. *Using UML for Modeling a Distributed Java Application*. 1997

[MUE96] J. Muelver. *Creación de Páginas Web con Perl*. Anaya Multimedia, S.A., 1996.

[MZ95] T.J. Mowbray and R. Zahavi. *The Essential CORBA*. John Wiley & Sons, Inc., 1995.

[OH97] R. Orfali and D.Harkey. *Client/Server Programming with JAVA and CORBA*. John Wiley & Sons, Inc., 1997.

[OHE96] R. Orfali, D.Harkey, and J. Edwards. *The Essential Distributed Objects Survival Guide*. John Wiley & Sons, Inc., 1996.

[OMG97] Object Management Group. *Common Object Request Broker Architecture and Specification, revision 2.1*, August 1997.

[OMG99] Object Management Group. *CORBA Main Site*, 1999. http://www.omg.org

[ORA99] Oracle. *Documentación HTML de Oracle 8*. 1999

[PAX95] Vern Paxson. *Flex, versión 2.5*. 1995

[POS98a] The PostgreSQL Global Development Group. *The PostgreSQL Administrator's Guide*. 1998

[POS98b] The PostgreSQL Global Development Group. *The PostgreSQL Programmer's Guide*. 1998

[PWGB98] D. Pedrick, J. Weedon, J. Goldberg, and E. Bleinfield. *Programming with VisiBroker*. John Wiley & Sons, Inc., 1998

[ROS97] J. Rosenberger. *Teach Yourself CORBA in 14 days*. Macmillan Computer Publishing, 1997.

[SEV98] Diego Sevilla Ruiz. *Aplicaciones Distribuidas en Internet/Intranets: de los Sockets a los Objetos Distribuidos*. Proyecto fin de carrera 1998.

[SUN99] SUN Microsystems. *The JDK 1.2 Documentation*, 1999. http://java.sun.com/products/jdk1.2/docs

[VD98] A. Voguel and K. Duddy. *JAVA Programming with CORBA*. John Wiley & Sons, Inc., 1998.

[VIS97a] Visigenic Software, Inc. *Visigenic's VisiBroker for Java Installation and Administration Guide, version 3.1*, November 1997.

[VIS97b] Visigenic Software, Inc. *Visigenic's Visibroker for Java Programmer's Guide, version 3.0*, September 1997.