Desarrollo en Java EE de una Librería virtual

Fernando Costas Comesaña

ITIS

Consultor: Javier Ferro García

16 de enero de 2007

Este trabajo de fin de carrera consiste en el desarrollo, implementación y despliegue de una aplicación utilizando la tecnología JavaEE. Esta tecnología Java permite la creación de aplicaciones basadas en la Web que funcionan a través de Internet de una forma sencilla. Las aplicaciones JavaEE están compuestas por diferentes componentes que se ensamblan dentro de la aplicación. Los principales componentes son: Las aplicaciones cliente que se encargan de implementar el modelo de negocio que se ejecuta en el lado del cliente, los componentes Java Servlet que son los componentes Web que se ejecutan en el lado del servidor y permiten interactuar con los usuarios y los Enterprise JavaBeans que son componentes de negocio y se ejecutan en el servidor y permiten acceder por ejemplo a base de datos. Todos estos componentes de despliegan en el entorno de producción, donde se ejecutan por un servidor de aplicaciones.

En concreto he desarrollado, implementado y he realizado del despliegue de una aplicación para la venta de libros a través de la Web. Esta pequeña librería virtual que he creado consta de dos partes bien diferenciadas, una que es el apartado de ventas, orientada a cualquier usuario de Internet, y otra que es la parte de administración para la gestión de la librería.

Índice

1. Justificación y Objetivos..... 4

2. Planificación..... 5
2.1 Proceso de Ingeniería seguido.....5
2.2. Planificación con hitos y temporización..... 5
2.3. Diagrama de Grantt de la planificación..... 6

3. Análisis de requisitos..... 7
3.1. Recogida y documentación de requisitos.....7
3.1.1 Consideraciones previas..... 7
3.1.2 Información inicial..... 7
3.2. Modelo del dominio y del negocio.....7
3.3. Modelo de casos de uso.....8
3.4. Requisitos de la interfaz de usuario..... 11
3.5. Especificación de las clases de análisis.....11
3.6. Análisis de la interfaz de usuario..... 13

4. Diseño.....18
4.1. El diseño arquitectónico.....18
4.2. El Diseño de los casos de uso, Modelo de Negocio.....19
4.2.1 Identificación de las relaciones entre clases.....21
4.3. Revisión del diagrama estático de diseño..... 23
4.3.1 Diseño del modelo de negocio..... 23
4.3.2 Diseño de los servlets. Interfaz de usuario.....24
4.4. Diseño de la persistencia.....24
4.4.1 Diagrama entidad-relación..... 25
4.4.2 Tablas base de datos..... 25

5. Programación.....27
5.1. Codificación de los componentes.....27
5.2. Descripción de las clases.....27
5.3. Creación de la base de datos..... 30
5.3.1 Creación de las tablas:.....30

6. Pruebas y Despliegue.....33
6.1. Juego de pruebas..... 33
6.2. Documentación de las pruebas realizadas.....35
6.3. Despliegue de la aplicación.....38

7. Conclusiones.....40

8. Bibliografía.....41

1. Justificación y Objetivos

El objetivo que busco al realizar este trabajo de fin de carrera es el de aprender a utilizar una tecnología, JavaEE, que creo que en este momento es fundamental para poder desarrollar aplicaciones pensadas en su uso en la Web o lo que es lo mismo en Internet.

El aprender la tecnología JavaEE creo que no se centra sólo en aprender a programar los diferentes componentes de la aplicaciones (Servlets, EJB, etc.) si no que también permite aprender a utilizar otro software como los servidores de aplicaciones, que son imprescindibles para el funcionamiento del software creado.

También quiero comentar que aunque los componentes Web de JavaEE utilizados para comunicarse con los usuarios pueden ser Servlets con páginas en HTML utilizando formularios o las mas elegantes páginas JSP, yo me he decido a utilizar formularios en HTML debido principalmente ya conocía este código y esto me a ayudado a desarrollar la aplicación de una forma más rápida que si tuviera que aprender a programar páginas JSP. Esta decisión la he tomado por que debido a que me ha costado mucho esfuerzo aprender la tecnología JavaEE, ya que la desconocía por completo antes de realizar este trabajo de fin de carrera, y me ha quedado muy poco tiempo para la implementación de la aplicación.

Por último me gustaría comentar que la principal motivación que me ha impulsado a realizar un trabajo de fin de carrera en el área de JavaEE es la posibilidad de utilizar un lenguaje de programación orientado a objetos multiplataforma, como Java, para realizar aplicaciones cliente/servidor utilizando una gran variedad de tecnologías de Internet. Además un aspecto que para mi es muy importante es que todas estas tecnologías se encuentran implementadas en software libre y de código abierto.

2. Planificación

2.1 Proceso de Ingeniería seguido.

El proceso de ingeniería que he utilizado a sido el clásico en cascada pero con partes de proceso interactivo. En cada etapa he ido obteniendo los documentos base para la etapa siguiente, pero a su vez he tenido que ir corrigiendo los errores de la etapa anterior. Esto lo achaco fundamentalmente, al desconocimiento que tenía de la tecnología aplicada 'JavaEE' al principio del trabajo. Esto me hecho incurrir en una serie de errores de diseño que he tenido que ir corrigiendo en etapas sucesivas.

2.2. Planificación con hitos y temporización.

La planificación del TFC viene marcada de forma indudable por las fechas de entrega de las distintas PEC y la fecha de entrega final. Por lo que toda la planificación gira alrededor de esas fechas.

Análisis previo.

Definir los objetivos del TFC.

Desarrollar el plan de trabajo

Entrega del plan de trabajo → PEC1 (29/09/06)

Búsqueda e instalación de las herramientas necesarias para el desarrollo del proyecto.

Análisis de requisitos.

Recogida y documentación de requisitos.

Modelo de casos de uso.

Documentación de requisitos de la interfaz de usuario.

Especificación de las clases.

Análisis de la interfaz de usuario.

Diseño.

Diseño de los casos de uso.

Revisión del diagrama estático de diseño.

Diseño de la persistencia.

Diseño de la interfaz gráfica de usuario.

Documentación del "Análisis de requisitos y diseño".

Entrega del análisis de requisitos y diseño → PEC2 (30/10/06)

Programación.

Codificación de los componentes.

Creación de la base de datos.

Pruebas.

Preparación de un juego de pruebas.

Revisión de la codificación.

Documentación de las pruebas.

Documentación de la implementación.

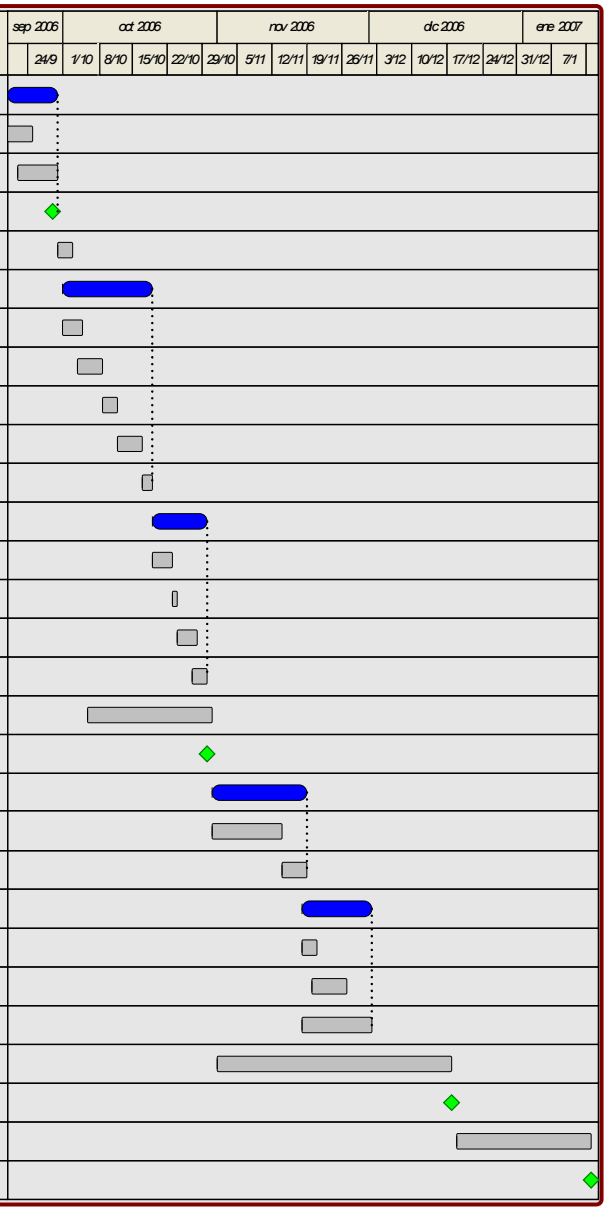
Entrega implementación → PEC3 (18/12/06)

Preparación documento final y presentación.

Entrega final → (15/01/07)

2.3. Diagrama de Grantt de la planificación:

Id	Nombre de tarea	Comienzo	Fin	Duración
1	**Análisis previo**	**20/09/2006**	**29/09/2006**	**1s 3d**
2	Definir los objetivos del TFC	20/09/2006	24/09/2006	5d
3	Desarrollar el plan de trabajo	22/09/2006	29/09/2006	1s 1d
4	**Entrega del plan de trabajo**	**29/09/2006**	**29/09/2006**	**0s**
5	Búsqueda/Instalación herramientas para el desarrollo del proyecto.	30/09/2006	02/10/2006	3d
6	**Análisis de requisitos**	**01/10/2006**	**18/10/2006**	**2s 4d**
7	Recogida y documentación de requisitos.	01/10/2006	04/10/2006	4d
8	Modelo de casos de uso.	04/10/2006	08/10/2006	5d
9	Documentación de requisitos de la interfaz de usuario.	09/10/2006	11/10/2006	3d
10	Especificación de las clases .	12/10/2006	16/10/2006	5d
11	Análisis de la interfaz de usuario.	17/10/2006	18/10/2006	2d
12	**Diseño**	**19/10/2006**	**29/10/2006**	**1s 4d**
13	Diseño de los casos de uso.	19/10/2006	22/10/2006	4d
14	Revisión del diagrama estático de diseño.	23/10/2006	23/10/2006	1d
15	Diseño de la persisténcia.	24/10/2006	27/10/2006	4d
16	Diseño de la interfaz gráfica de usuario	27/10/2006	29/10/2006	3d
17	Documentación del análisis de requisitos y diseño.	06/10/2006	30/10/2006	3s 4d
18	**Entrega del análisis de requisitos y diseño.**	**30/10/2006**	**30/10/2006**	**0s**
19	**Programación**	**31/10/2006**	**18/11/2006**	**2s 5d**
20	Codificación de los componentes.	31/10/2006	13/11/2006	2s
21	Creación de la base de datos.	14/11/2006	18/11/2006	5d
22	**Pruebas**	**18/11/2006**	**01/12/2006**	**2s**
23	Preparación de un juego de pruebas.	18/11/2006	20/11/2006	3d
24	Revisión de la codificación .	20/11/2006	26/11/2006	1s
25	Documentación de las pruebas.	18/11/2006	01/12/2006	2s
26	Documentación de la implementación.	01/11/2006	17/12/2006	6s 5d
27	**Entrega implementación**	**18/12/2006**	**18/12/2006**	**0s**
28	Preparación documento final y presentación.	19/12/2006	14/01/2007	3s 6d
29	**Entrega final**	**15/01/2007**	**15/01/2007**	**0s**



3. Análisis de requisitos

Explicar el método de ingeniería seguido (RUP...)

3.1. Recogida y documentación de requisitos.

3.1.1 Consideraciones previas.

La información con que cuento para realizar este proyecto de librería virtual, la he recogido de varias web de librerías a través de Internet, como pueden ser: "Díaz de Santos", "La Casa del Libro", "Andel Virtual", etc... En base a lo que he visto en estas librerías he realizado la parte pública de este proyecto, orientada a los clientes. Para el desarrollo del área privada del proyecto, administración de la librería, me he basado en mi propia experiencia con software de gestión.

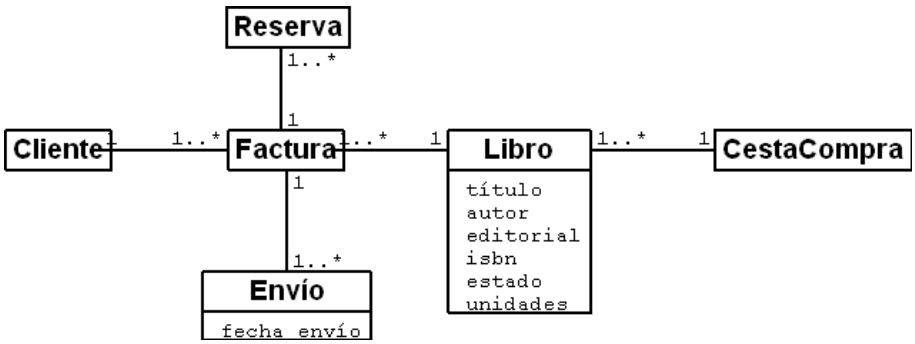
3.1.2 Información inicial.

Los clientes de la librería virtual acceden a esta a través de una página web. En esta web el cliente tiene que poder elegir los libros que le interesan de varias formas, la más básica sería buscándolos a través de su título, autor, editorial o ISBN; otra forma puede ser a través de secciones tipo: novedades, ofertas, recomendados o libros que serán publicados próximamente; una vez que el cliente ha localizado los libros que le interesan los irá añadiendo a una cesta de la compra, para que cuando finalice su proceso de selección se le puedan facturar los libros escogidos. Una vez realizado este último paso se le enviarán los libros adquiridos. Si de alguno de los libros solicitados por el cliente no hay stock se le dará la oportunidad de dejarlo reservado, de forma que cuando haya unidades se le pueda enviar.

La administración de la librería comprenderá la gestión de los libros, (altas, actualizaciones de stocks, etc.), otra de las funciones de la administración de la librería será el poner los libros en las diferentes secciones: ofertas, novedades, recomendados o próxima publicación. También se llevará la gestión de los pedidos informando a los clientes de las incidencias que se puedan dar y la gestión de las reservas de libros para enviar los libros cuando estén disponibles.

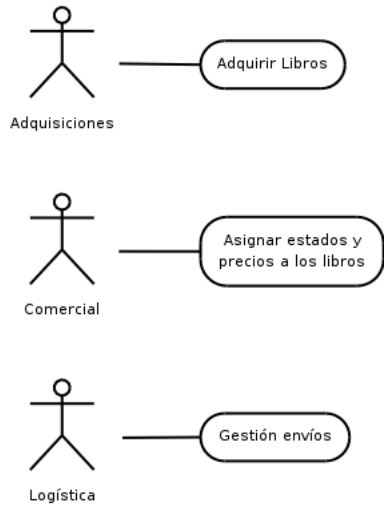
3.2. Modelo del dominio y del negocio.

Se identifican los siguientes objetos del dominio: "cliente", "libro", "factura", "envío", "reserva"y "cestaCompra".



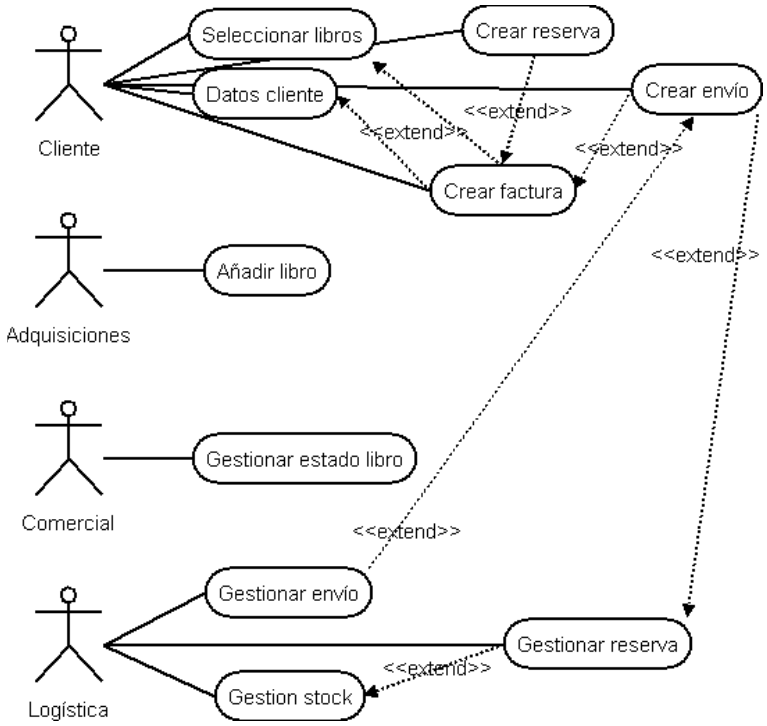
En el modelo del negocio dentro del grupo de los administradores podemos diferenciar tres funciones: una que realizaría un encargado de la adquisición de los libros; otra para el encargado de la gestión comercial, sería la persona encargada de situar los libros en las diferentes secciones de la librería, ofertas, novedades, etc.; y otra para el encargado de la gestión de los envíos y reservas.

Podemos ver el diagrama de casos de uso del modelo de negocio.



3.3. <u>Modelo de casos de uso</u>.

En los casos de uso tenemos cuatro actores que son: el cliente, el gestor de adquisiciones, el encargado de la administración comercial y el encargado de los envíos a los clientes. Al gestor de adquisiciones lo denominaremos por la palabra "adquisiciones", al gestor comercial por "comercial" y al encargado de los envíos por "logística".



Caso de uso 1: "Seleccionar libros"
Resumen de la funcionalidad: selecciona los libros que va a comprar el cliente.
Papel en el trabajo del usuario: es el caso de uso principal de los clientes.
Actores: cliente.
Casos de uso relacionados: Crear factura, Crear reserva, Crear envío y Gestionar envío.
Precondición: los libros tiene que existir en la base de datos.
Postcondición: si se selecciona algún libro se crea una factura y un envío o una reserva.

El cliente va seleccionando distintos libros de la base de datos de la librería, bien buscándolos a través de su título, autor, editorial o isbn o seleccionándolos directamente de las distintas secciones: novedades, ofertas, etc. Los libros que el cliente va seleccionando se guardan en una cesta de la compra.

Caso de uso 2: "Datos cliente"
Resumen de la funcionalidad: añade un cliente a la base de datos.
Papel en el trabajo del usuario: todos los clientes tiene que dar sus datos para poder comprar los libros.
*Actore*s: cliente.
Casos de uso relacionados: Crear factura, Crear envío y Crear reserva.
Precondición: el cliente ha seleccionado libros para comprar.
Postcondición: los datos del cliente si este no existía se incorporan a la base de datos.

Todos los clientes tiene que identificarse o introducir sus datos, si no esta dado de alta en la base de datos. Estos datos son los que se usarán para la factura y para el envío.
Cuestiones que hay que aclarar: En un principio todos los clientes son anónimos. Pueden realizar todo el proceso de compra sin tener que identificarse, sólo cuando formalizan la compra, tienen la obligación que identificarse, en ese momento pueden darse dos situaciones, que el cliente ya esté registrado en la base de datos y se identifique usando un nombre de usuario y una contraseña, o que el cliente no exista y tenga que introducir todos sus datos para poder completar el procedimiento.

Caso de uso 3: "Crear factura"
Resumen de la funcionalidad: crea la factura de la compra.
Papel en el trabajo del usuario: para el cliente es el paso final en el procedimiento de compra.
Actores: cliente.
Casos de uso relacionados: Seleccionar libros, Datos cliente, Crear reserva, Crear envío.
Precondición: el cliente tiene que haber seleccionado libros y tienen que existir los datos del cliente.
Postcondición: se añadirá la factura a la base de datos y se creará el envío y si se da el caso se creará una reserva.

Antes de crear la factura, se comprobará el stock de todos los libros solicitados por el cliente, si alguno de ellos no tiene stock, se informará al cliente de esa circunstancia y se le dará la oportunidad de eliminarlo de su compra o dejarlo reservado. Una vez conformada la factura por el cliente, se procederá a su emisión virtual y se creará el envío y en su caso la reserva.
Cuestiones que hay que aclarar: en el momento de pedir la conformación de la factura al cliente, también se le pedirá que seleccione el modo de envío, de forma que los gastos de este ya se carguen en la factura.

Caso de uso 4: "Crear reserva"
Resumen de la funcionalidad: reserva los libros que se han comprado pero no tienen stock.
Papel en el trabajo del usuario: la reserva se crea siempre que el cliente haya aceptado comprar libros de los que en ese momento no hay stock.
Actores: cliente.
Casos de uso relacionados: Crear factura y Crear pedido.

Precondición: que los libros no tengan stock y que el cliente haya aceptado comprarlos.
Postcondición: se añade la reserva a la base de datos.

Cuando algún libro de los solicitados por el cliente no tiene stock, se le ofrece al cliente si quiere eliminarlo de la compra o dejarlo reservado, en este último caso se crea la reserva.

Caso de uso 5: "Crear envío"
Resumen de la funcionalidad: se crea un envío con los libros comprados.
Papel en el trabajo del usuario: el envío se crea siempre que un cliente compra libros y estos están en stock.
Actores: cliente.
Casos de uso relacionados: Crear factura, Gestionar reserva y Gestionar envío.
Precondición: para poder crear el envío el cliente tiene que aceptar la factura y los libros deben tener stock.
Postcondición: se añade un envío a la base de datos.

En la propia factura al cliente se le pide que seleccione el medido de envío para los libros que ha comprado, de esta forma se le incluye el importe en la propia factura. Una vez que el cliente acepta la factura, se prepara un envío con los libros que ha comprado y de los cuales hay stock.
Cuestiones que hay que aclarar: el envío también se puede generar al gestionar una reserva, siempre que entren unidades de los libros reservados.

Caso de uso 6: "Añadir libro"
Resumen de la funcionalidad: añade nuevos libros a la base de datos.
Papel en el trabajo del usuario: el encargado de las adquisiciones es el encargado de añadir los nuevos libros.
Actores: adquisiciones.
Casos de uso relacionados: Crear pedido.
Precondición: ninguna.
Postcondición: se añade el libro a la base de datos.

El encargado de las adquisiciones se encarga también de añadir los libros nuevos a la base de datos, este procedimiento puede ser independiente o puede ser lanzado por el procedimiento de crear pedidos.

Caso de uso 7: "Gestionar estado libro"
Resumen de la funcionalidad: permite cambiar el estado de los libros y de esta forma los ponemos en las distintas secciones de la web.
Papel en el trabajo del usuario: el administrador encargado de la gestión comercial es el responsable de cambiar el estado de los libros, (ofertas, novedades, etc.).
Actores: comercial.
Casos de uso relacionados: Seleccionar libros.
Precondición: el libro tiene que existir en la base de datos.
Postcondición: se modifica el estado del libro.

Este proceso permite al gestor comercial modificar el estado de los libros atendiendo a criterios comerciales.

Caso de uso 8: "Gestionar envío"
Resumen de la funcionalidad: va registrando los estados por los que pasa un envío, desde su creación hasta la recepción por el cliente.
Papel en el trabajo del usuario: el administrador encargado de la logística va registrando todas las incidencias de los envíos y todas las comunicaciones con el cliente.
Actores: logística.

Casos de uso relacionados: Crear envío.
Precondición: tiene que existir el envío.
Postcondición: nuevo estado del envío.

Mediante este procedimiento el encargado de la logística registra todos los pasos del envío desde que se crea hasta que lo recibe el cliente. También envía al cliente las comunicaciones de los procesos del envío. Además podrá anotar las observaciones que crea pertinente acerca del envío.

Caso de uso 9: "Gestión stock"
Resumen de la funcionalidad: gestiona el número de unidades y precio de los libros de la librería.
Papel en el trabajo del usuario: el administrador encargado de la logística según recibe los pedidos de las editoriales varía el número de unidades de los libros y sus precios.
Actores: logística.
Casos de uso relacionados: Crear pedido, Gestionar reserva.
Precondición: tiene que existir un pedido y por supuesto el libro.
Postcondición: se modifica el número de unidades y precios de los libros.

Este procedimiento permite gestionar las unidades y precios de los libros de la tienda.

Caso de uso 10: "Gestionar reserva"
Resumen de la funcionalidad: permite gestionar las reservas de libros, cuando entran nuevas unidades de libros reservados convierte la reserva en un envío.
Papel en el trabajo del usuario: el encargado de logística gestiona las reservas para convertirlas en envíos.
Actores: logística.
Casos de uso relacionados: Gestión stock, Crear envío.
Precondición: tiene que existir la reserva y tienen que entrar nuevas unidades de los libros reservados.
Postcondición: la reserva, o parte de ella, se convierte en envío.

El encargado de logística gestiona las reservas de tal forma que cuando entran nuevas unidades de los libros reservados, las reservas se van convirtiendo en envíos.

3.4. Requisitos de la interfaz de usuario.

Vamos a tener dos tipos bien diferenciados de usuarios, por un lado los clientes y por otro los administradores de la librería. Por supuesto de los clientes no conocemos nada y por lo tanto el software tiene que ser lo mas sencillo e intuitivo posible, y que cuente siempre que se pueda con ayudas y/o ejemplos de las funciones que puede realizar. Los administradores son usuarios conocidos, por lo tanto pueden recibir cursillos de formación sobre el software a implementar.

3.5. Especificación de las clases de análisis.

Identificación de las clases de entidades a partir de los casos de uso:

- Caso de uso 1 "Seleccionar libros". Clases: Libro, CestaCompra.
- Caso de uso 2 "Datos cliente". Clases: Cliente.
- Caso de uso 3 "Crear factura". Clases: CestaCompra, Factura, Cliente y Libro.
- Caso de uso 4 "Crear reserva". Clases: Reserva, Factura, Cliente y Libro.
- Caso de uso 5 "Crear envío". Clases: Envío, Factura, Cliente y Libro.
- Caso de uso 7 "Añadir Libro". Clases Libro.
- Caso de uso 8 "Gestionar estado libro". Clases: Libro.
- Caso de uso 9 "Gestionar envío". Clases: Envío, Cliente, Factura y Libro.
- Caso de uso 10 "Gestión stock". Clases: Pedido y Libro.

- Caso de uso 11 “Gestionar reserva”. Clases: Reserva, Factura, Cliente, Libro y Envío.

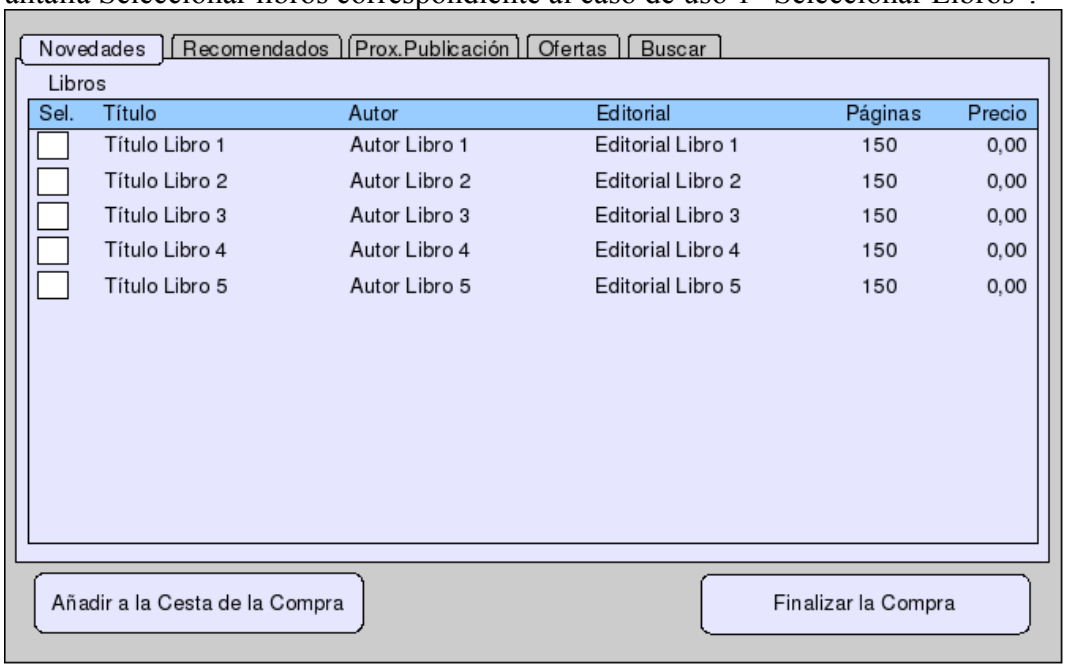
Aclaraciones: debido a que las clases Factura, Reserva y Envío deben contener normalmente datos de más de un libro, las he dividido en dos de forma que la clase Factura contendrá los datos principales de la factura y la nueva clase FacturaLinea contendrá los datos correspondientes a cada libro que pertenece a la factura. Y he hecho lo mismo con las otras tres clases.

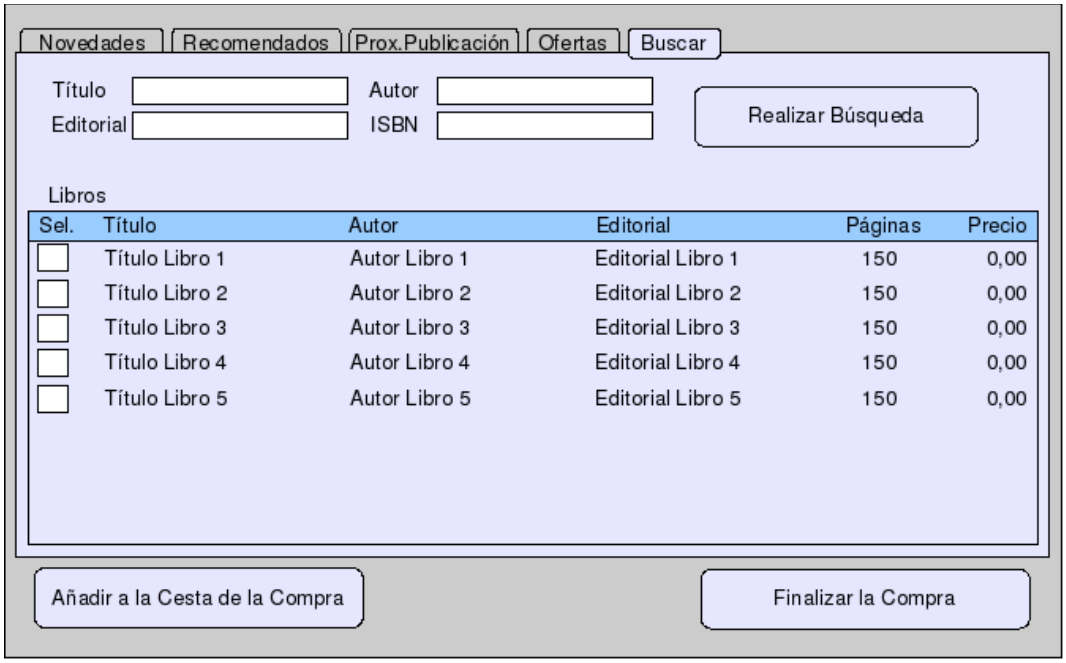
Especificación de los atributos de las clases de entidades.

- *Clase Libro*: titulo (string), autor (string), editorial (string), isbn (string), fecha_edicion (date), paginas (int), estado (string), unidades (int), reservados (int), precio (real), iva (real).
- *Clase Cliente*: nombre (string), nif (string), direccion_factura (string), telefono (string), correo (string), nombre_usuario (string), password (string).
- *Clase CestaCompra*: librosCesta (Libro`[]).
- *Clase Factura*: numero_factura (int), fecha (date), cliente (Cliente), direccion_entrega (string), format_envio (string), forma_pago (string).
- *Clase FacturaDetalle*: libro (Libro), unidades (int), precio (real), iva (real).
- *Clase Reserva*: numero_reserva (int), numero_factura (int), cliente (Cliente), fecha_reserva (date).
- Clase *ReservaDetalle*: libro (Libro), unidades (int).
- *Clase Envio*: numero_envio (int), numero_factura (int), cliente (Cliente), direccion_entrega (string), fecha_envío (date), estado_envio (string), observaciones (string).
- Clase *EnvioDetalle*: libro (Libro), unidades (int).
- *Clase Editorial*: nombre (string), nif (string), direccion (string), telefono (string), fax (string), correo (string).

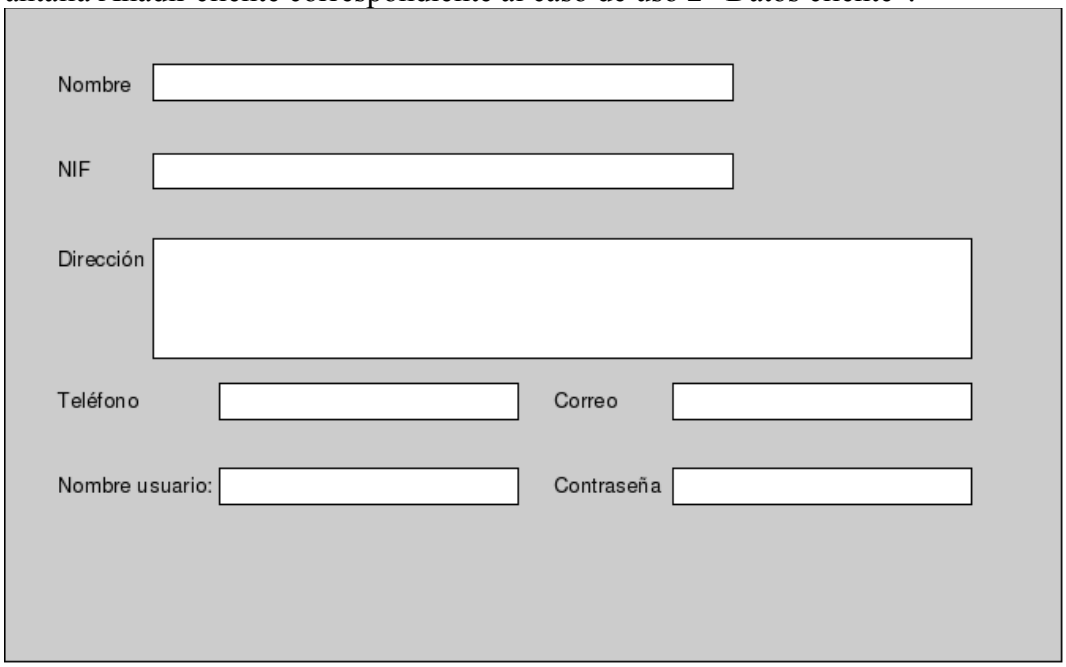
3.6. Análisis de la interfaz de usuario.

1) Pantalla Seleccionar libros correspondiente al caso de uso 1 “Seleccionar Libros”.



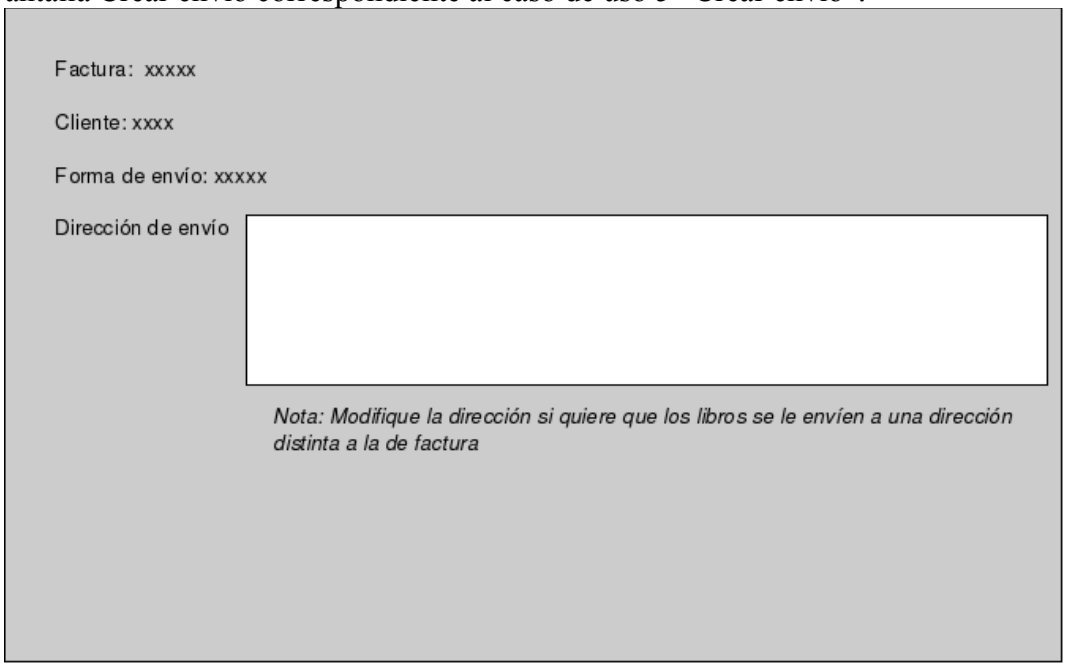


2) Pantalla Añadir cliente correspondiente al caso de uso 2 “Datos cliente”.

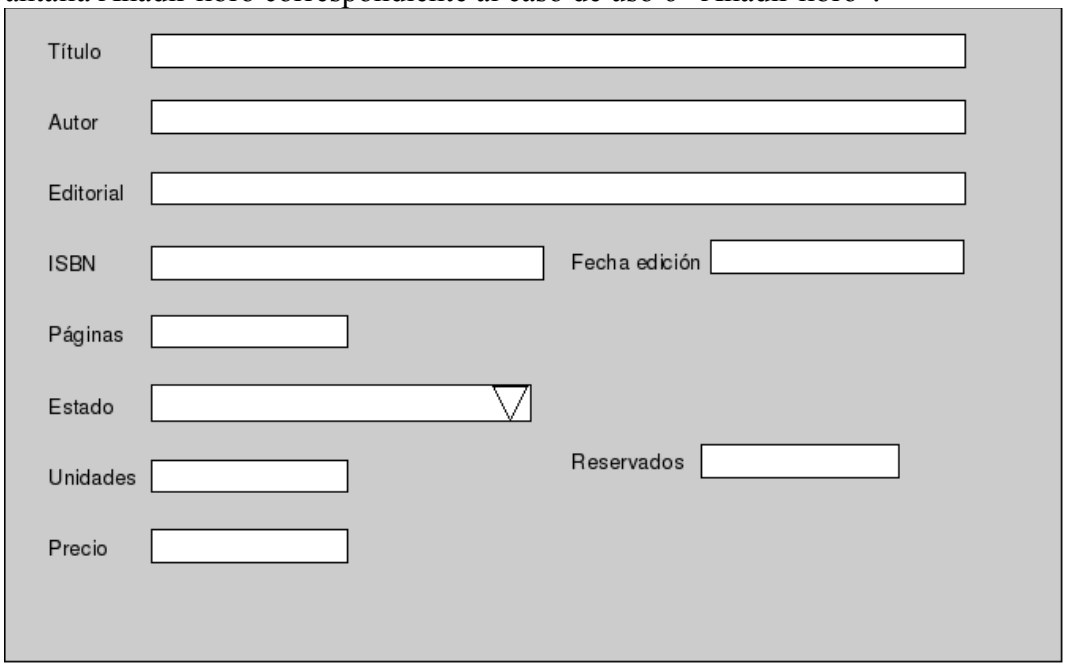


3) Pantalla Crear factura correspondiente a los casos de uso 3 y 4 “Crear factura” y “Crear reserva”.

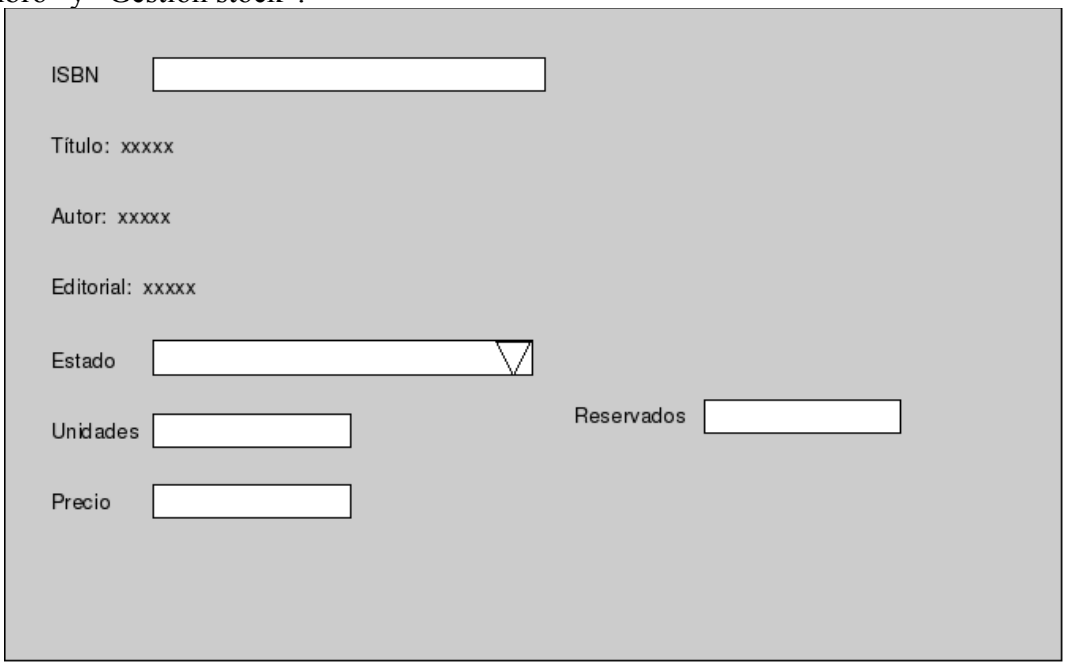
4) Pantalla Crear envío correspondiente al caso de uso 5 “Crear envío”.



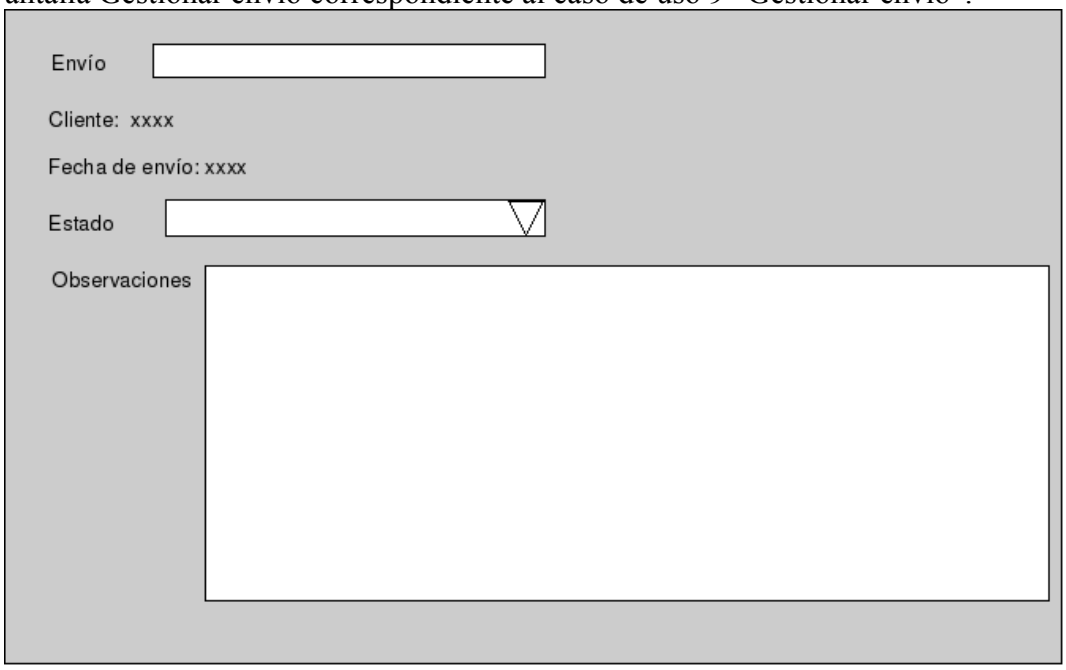
5) Pantalla Añadir libro correspondiente al caso de uso 6 “Añadir libro”.



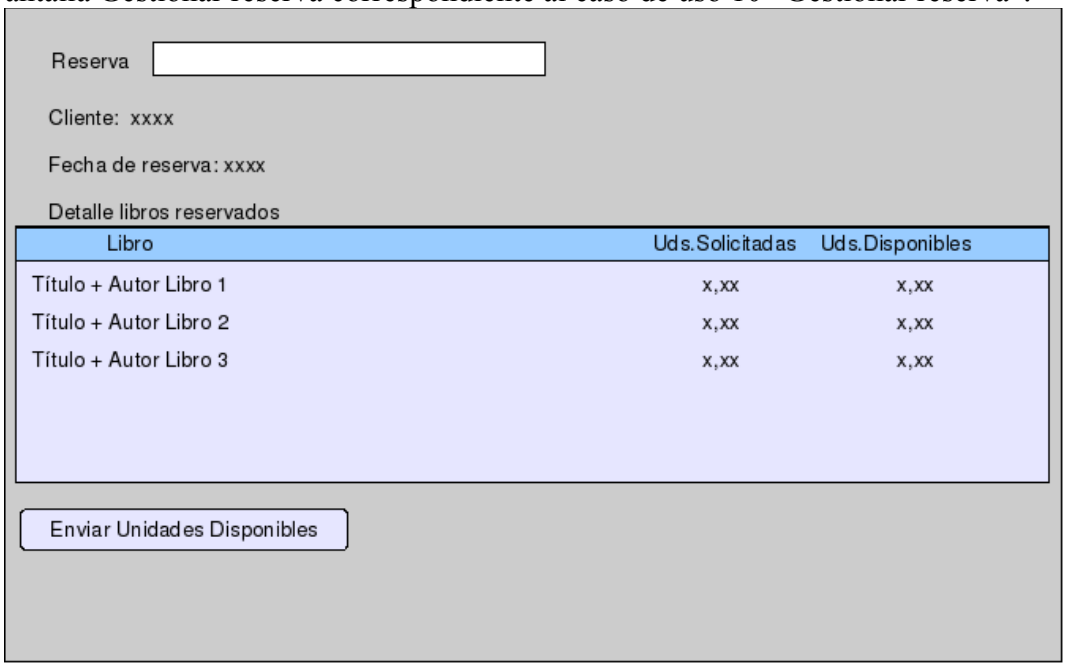
6) Pantalla Gestionar estado libro correspondiente a los casos de uso 7 y 9 “Gestionar estado libro” y “Gestión stock”.



7) Pantalla Gestionar envío correspondiente al caso de uso 9 “Gestionar envío”.



8) Pantalla Gestionar reserva correspondiente al caso de uso 10 “Gestionar reserva”.



4. Diseño

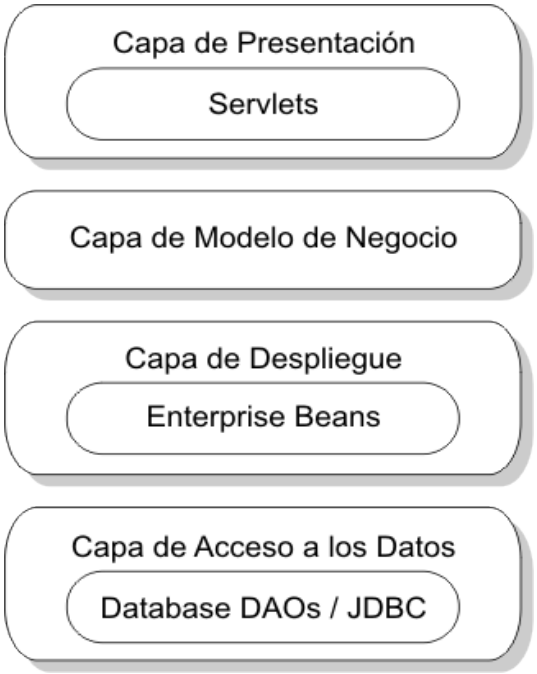
4.1. El diseño arquitectónico.

La aplicación a desarrollar consta de una arquitectura cliente/servidor y su ámbito de ejecución será Internet. Por ello he escogido el lenguaje de programación Java, ya que las aplicaciones creadas en éste se pueden ejecutar en prácticamente cualquier máquina con cualquier sistema operativo. Como en este caso no podemos conocer las máquinas ni los sistemas operativos de los usuarios/clientes de esta aplicación creo que Java el lenguaje de programación más adecuado para el objetivo perseguido.

Para la creación e implementación de la aplicación voy a utilizar Java 2 Edición Empresarial (J2EE) ya que esta edición empresarial del lenguaje java proporciona servicios web, modelos de componentes, gestión y APIs de comunicaciones que hacen de ella el estándar de la industria para el diseño de aplicaciones distribuidas.

Como servidor de aplicaciones voy a utilizar el middleware JBOSS AS, que es un servidor de aplicaciones J2EE.

Para la realización de esta aplicación voy a utilizar una arquitectura en cuatro capas que son: la capa de presentación, la capa de despliegue, capa de modelo de negocio y capa de acceso a los datos.



La capa de presentación es la encargada de la interfaz con los usuarios de la aplicación. Para esta aplicación y por motivos de simplicidad voy a utilizar html para la creación de la interfaz de usuario. El código html es generado dinamicamente por los servlets.

La capa de modelo de negocio comprende las clases java encargadas de modelar el negocio. Por ejemplo en esta aplicación serán las clases: CestaCompra, Cliente, Factura, etc.

La capa de despliegue es el corazón de la arquitectura J2EE. En el caso de esta aplicación estará formada por una sola clase EJB que va a ser la encargada de comunicarse con la base de datos.

La capa de acceso a los datos es la encargada de la persistencia de los datos. Para la persistencia voy a utilizar como método de acceso JDBC para comunicar con el servidor de bases de datos. Para el desarrollo de esta aplicación voy a utilizar como SGBD MySQL.

4.2. El Diseño de los casos de uso, Modelo de Negocio.

Caso de uso 1: "Seleccionar libros"

Cuando el usuario/cliente accede a la web de la librería accede directamente a la pantalla de selección de libros, por lo que se llama a la operación *presentar* de la clase *SeleccionarLibro*. Esta operación presenta un listado con los libros que pertenecen a la opción seleccionada (Novedades, Recomendados, etc..) también permite hacer búsquedas (por Título, Autor, etc..).

El usuario cuenta con un botón *"Añadir a la Cesta de la Compra"* para ir escogiendo los libros que vaya marcando, y de esta forma ir realizando su compra. Este procedimiento llama a la operación *anadirCesta* de la clase *SeleccionarLibro*.

Cuando el usuario quiere confirmar la compra de los libros tiene que pulsar el botón "*Finalizar la Compra"* de esta manera pasa a realizar la siguiente operación que esta descrita en el caso de uso 3 "*Crear factura"*. Se llama a la operación *preparaFactura* de la *SeleccionarLibro*.

Si el usuario no esta interesado en realizar ninguna compra, simplemente saliendo de la web se finaliza el proceso. Para que no queden, en el servidor, registros temporales de los clientes correspondientes a las cestas de la compra mientras selecciona los libros, estos datos se guardarán en el ordenador del usuario. De tal forma que si este cancela la compra, en el servidor no quedará registrado ningún dato.

Caso de uso 2: "Datos cliente"

En el caso de uso 3 "*Crear factura*", se le pide al cliente/usuario que se identifique mediante un nombre de usuario y una contraseña, si este no está registrado entonces deberá darse de alta. Cuando sucede esto se llama a la operación *crear* de la clase *Cliente*, y se abre la pantalla "*Datos cliente"*.

El cliente tiene que ir cubriendo los datos que se le piden en la pantalla. Si el nombre de usuario que elige ya existe se muestra un mensaje informándole de este hecho y no se le deja continuar hasta que elija un nombre de usuario nuevo.

Caso de uso 3: "Crear factura"

Una vez que el usuario pulsa el botón de *"Finalizar la Compra"* se llama a la operación *crear* de la clase *Factura*. Se presenta en la pantalla la lista de los libros seleccionados por el usuario. Dentro de esa lista puede indicar el número de unidades que quiere comprar de cada libro y también se le informa de si el libro esta disponible o no.

El usuario tiene que seleccionar la una forma de pago entre una lista de ellas y también tendrá que seleccionar la forma de pago.

Si dentro de la lista de libros seleccionados aparecen libros que no están disponibles puede eliminarlos pulsando el botón *Eliminar Artículos Sin Stock*. De todas formas puede eliminar cualquier libro poniendo su numero de unidades a 0.

Si el usuario pulsa el botón *Validar Factura*, esta se crea y se pasa al siguiente caso de uso "Crear envío". Si el usuario cancela la operación entonces no se guarda ningún dato.

Caso de uso 4: "Crear reserva"

Cuando el usuario valida una factura, si en esta hay libros de los que no disponemos de unidades se llama a la operación *crear* de la clase *Reserva*.

Caso de uso 5: "Crear envío"

Una vez validada la factura, si hay libros para enviar, se llama a la operación *crear* de la clase *Envío*, y se presenta la pantalla "*Crear envío*". En esta pantalla el usuario si tiene una dirección de envío distinta a la de su dirección de facturación puede escribir esta.

De esta pantalla solo se puede salir validando los datos. De tal forma que si el usuario cerrara el navegador o perdiera la conexión en ese momento, se tomaría como dirección de envío la dirección de la facturación.

Caso de uso 6: "Añadir libro"

El administrador selecciona la opción correspondiente del menú y se llama a la operación *crear* de la clase *Libro*. Se presenta la pantalla "*Añadir Libro"*. Si el usuario acepta esta pantalla entonces se llama a la operación *guardar* de la clase *Libro*.

Caso de uso 7: "Gestionar estado libro"

El usuario selecciona la opción correspondiente del menú y se llama a la operación *gestionEstado* de la clase *Libro*. Se presenta la pantalla "*Gestionar estado libro"*, en esta pantalla el usuario tiene que introducir el ISBN del libro que quiere gestionar, una vez que se selecciona el libro, el usuario sólo puede modificar su estado. Cuando el usuario acepta esta pantalla, se llama a la operación *guardaEstado* de la clase *Libro*.

Caso de uso 8: "Gestionar envío"

Cuando el usuario selecciona esta opción en el menú entonces se llama a la operación *gestionarEnvío* de la clase *Envio*. El usuario tiene que introducir el número del envío que quiere gestionar. Una vez seleccionado el envío puede modificar su estado y anotar observaciones. Cuando el usuario acepta esta pantalla se llama a la operación *guardaEstado* de la clase *Envio*.

Caso de uso 9: "Gestión stock"

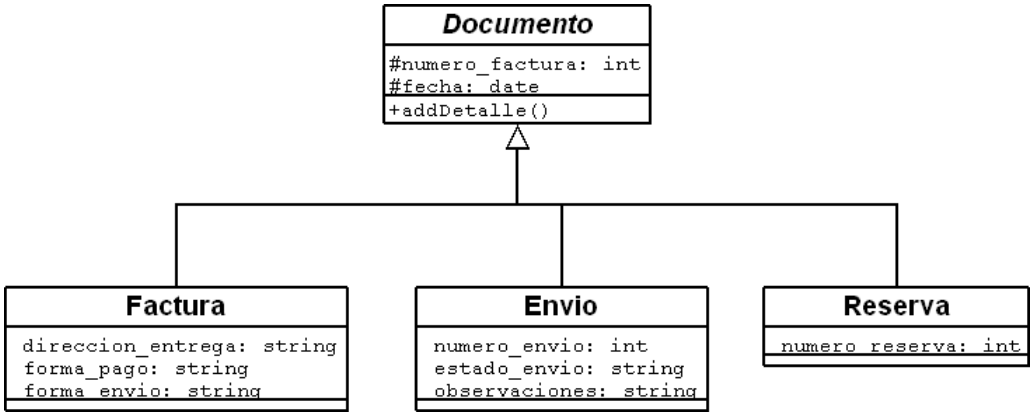
Al seleccionar esta opción se llama a la operación *gestionStock* de la clase *Libro*. Se muestra la pantalla "*Gestionar estado libro*". El usuario tiene que introducir el ISBN del libro al que quiere modificar el stock. En este caso la pantalla deja modificar únicamente los campos *Unidades* y *Precio*. Cuando el usuario acepta esta pantalla se llama a la operación *guardaStock* d ella clase *Libro*.

Caso de uso 10: "Gestionar reserva"

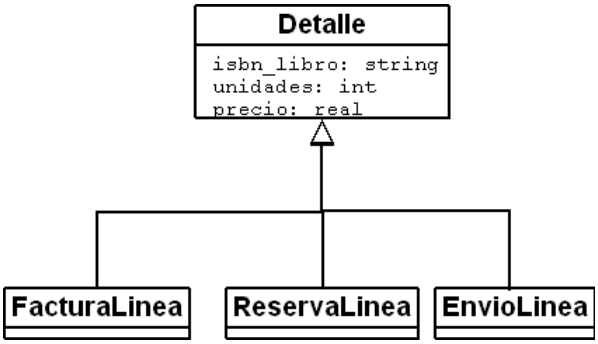
En este caso se llama a la operación g*estion* de la clase *Reserva*. El usuario introduce el número de la reserva que quiere gestionar y en la pantalla "*Gestionar reserva"* aparecerán los datos del cliente, la fecha de la reserva y la lista de los libros que tiene reservados. Para cada libro de la lista aparecerán las unidades solicitadas y las disponibles. El usuario lo único que puede hacer es, si hay unidades disponibles, crear un envío nuevo para el cliente pulsando el botón *Enviar Unidades Disponibles*. Esta acción llama a la operación *crearEnvio* de la clase *Reserva.*

4.2.1 Identificación de las relaciones entre clases.

Relaciones de herencia: ya que las clases Factura, Reserva y Envío comparten varios atributos y métodos, se puede que las tres clases hereden de una clase abstracta que denominaré documento.

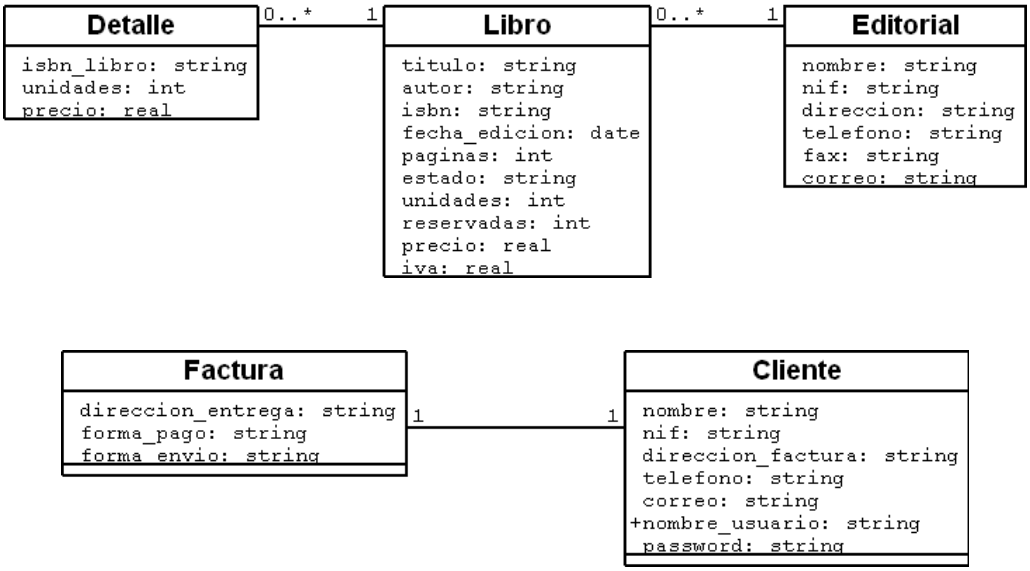


Las clases FacturaLinea, ReservaLinea, EnvioLinea tienen prácticamente todos los atributos comunes, podríamos considerar hacerlas subclases de una clase principal a la que podríamos denominar Detalle. El diagrama de estas clases quedaría así:

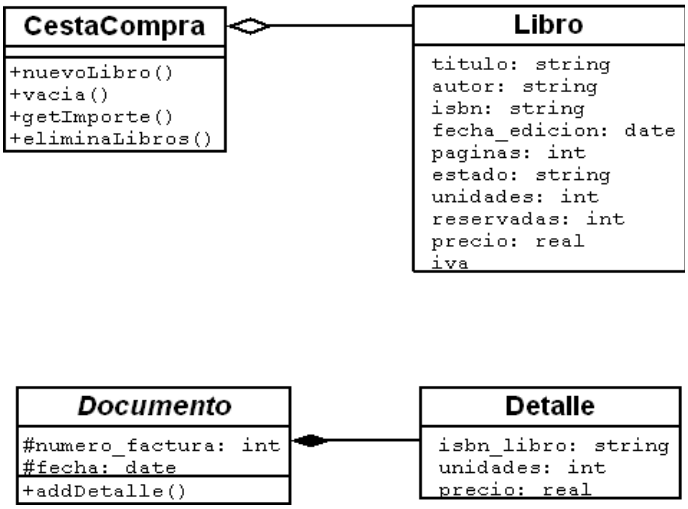


Como se ve las clases FacturaLinea, ReservaLinea, EnvioLinea quedan sin atributos propios, solo tienen los atributos heredados, por lo que podemos suprimirlas y sustituirlas por la clase Detalle ya que las cuatro tienen la misma función.

Asociaciones:

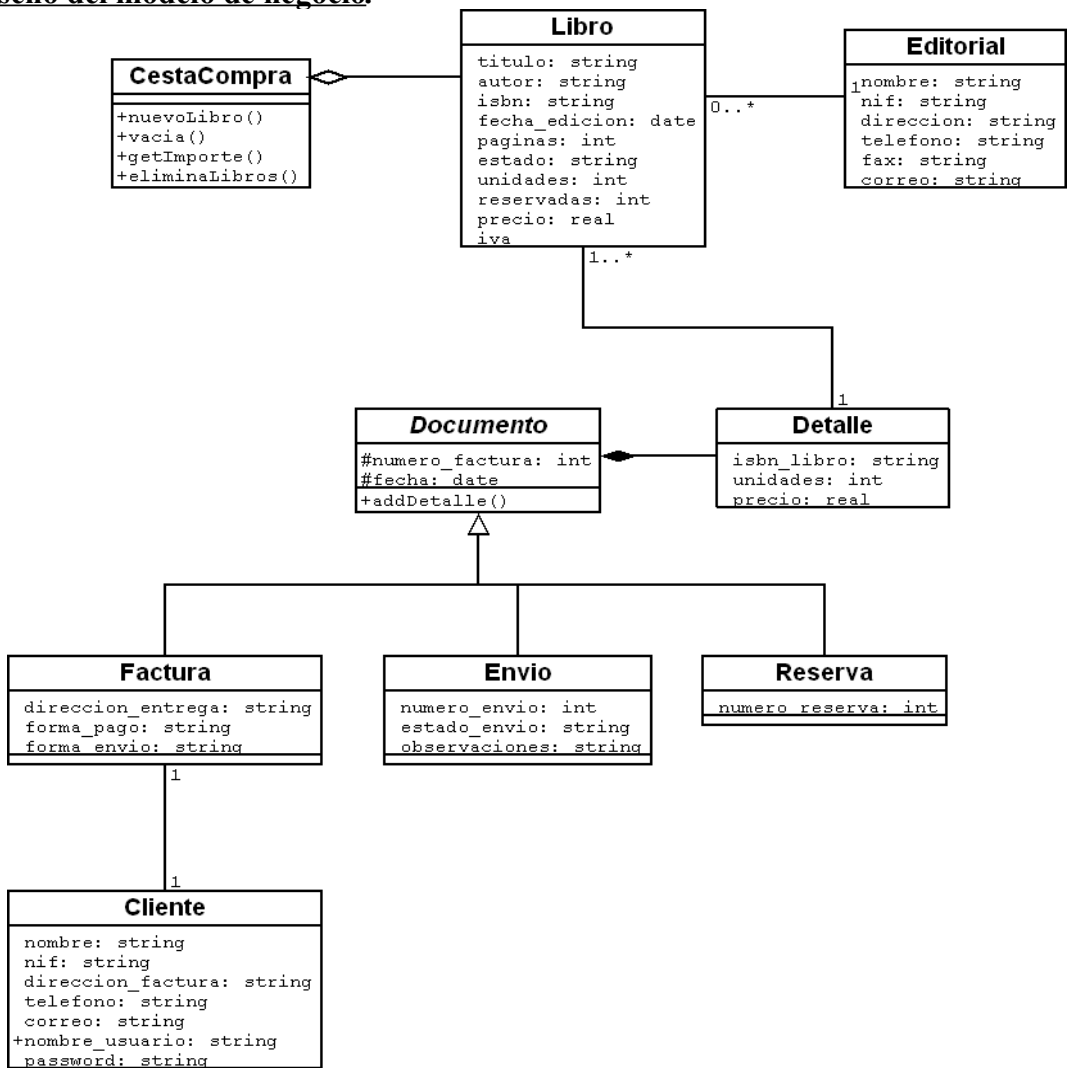


Agregaciones:

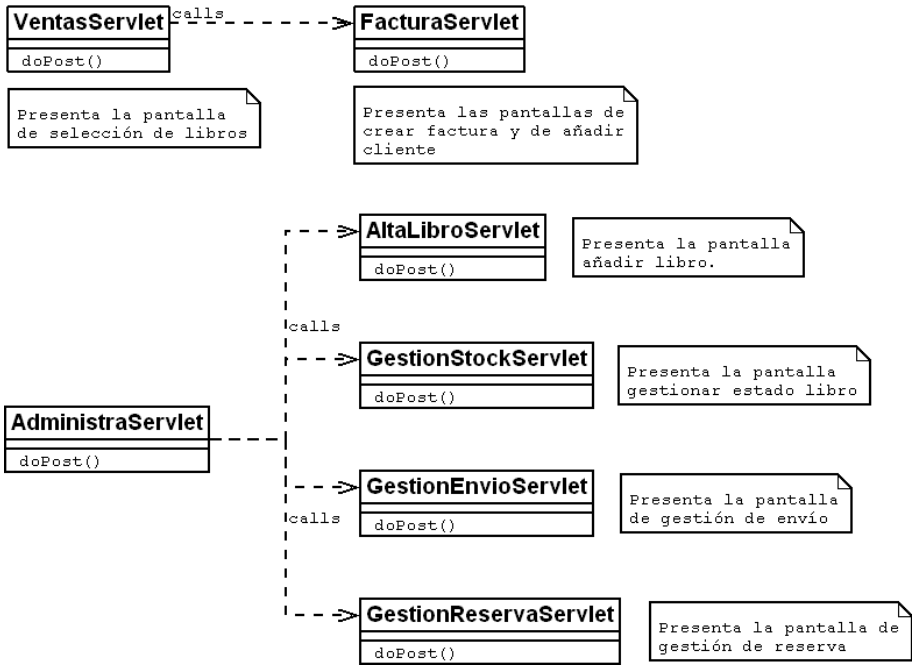


4.3. Revisión del diagrama estático de diseño.

4.3.1 Diseño del modelo de negocio.



4.3.2 **Diseño de los servlets. Interfaz de usuario**.

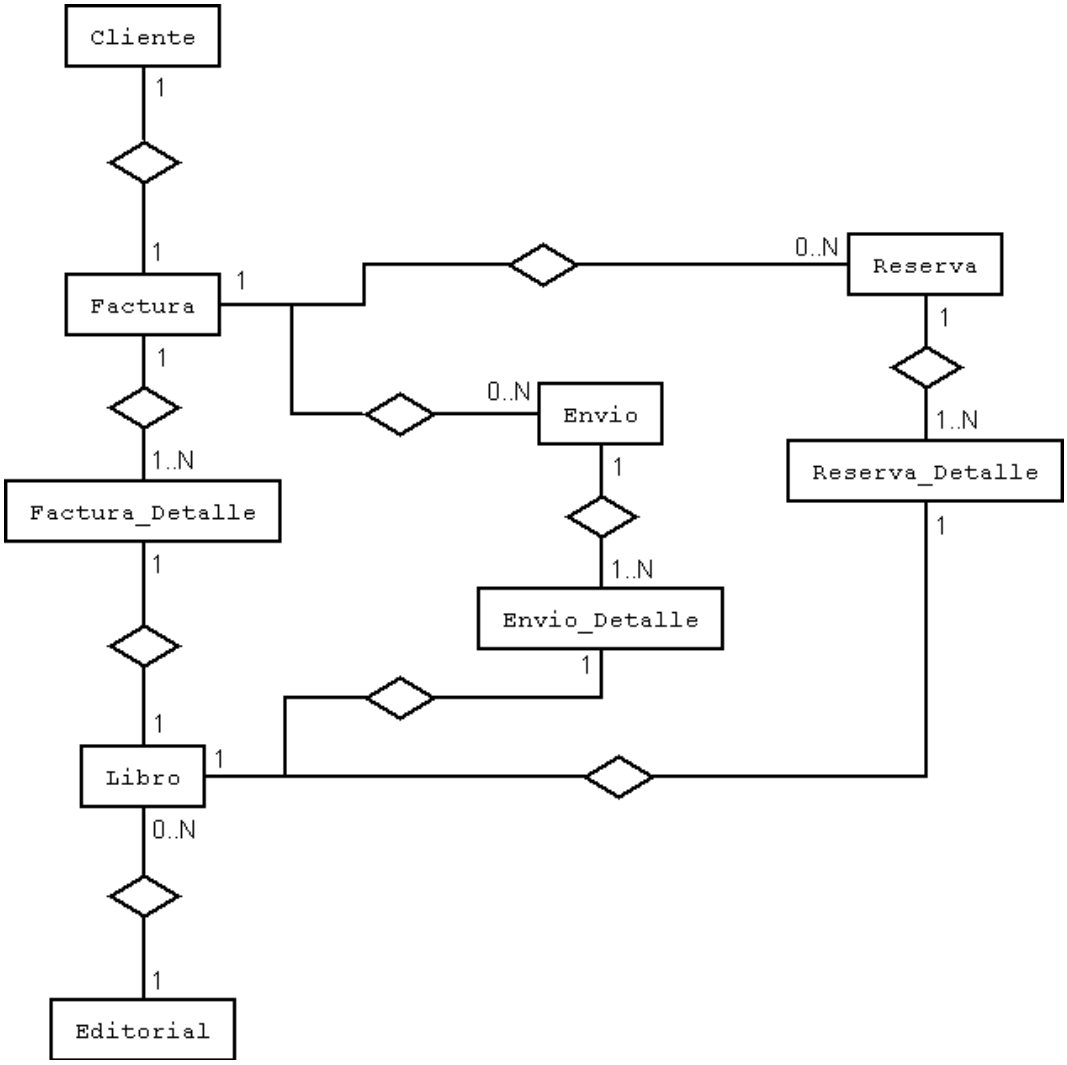


4.4. Diseño de la persistencia.

Para el diseño de las persistencia hay que eliminar las relaciones de herencia, en este caso sólo existe la herencia de la clase *Documento* a las clases *Factura, Reserva* y *Envio*. La clase *Detalle* que mantiene relaciones de agregación con las clases *Factura*, *Reserva y Envio* va a dar lugar a varias tablas, una por cada relación de agregación.

La clase CestaCompra no genera persistencia ya que esta clase se utiliza únicamente durante el proceso de selección de libros por el cliente. Una vez seleccionados los libros, la cesta de la compra se convierte en una factura y en un envío y/o reserva según corresponda, pero no se guardan sus datos.

4.4.1 Diagrama entidad-relación.



4.4.2 Tablas base de datos.

El conjunto de tablas relacionales obtenido es el siguiente: (Aparecen subrayadas las claves principales.)

Factura(numero_factura, fecha, nombre_cliente, nif_cliente, direccion_cliente, direccion_entrega, forma_pago, iva)
<nif_cliente> es clave foránea en Cliente.

Factura_Detalle(idDetalle, numero_factura, isbn_libro, unidades, precio)
<numero_factura> es clave foránea en Factura.
<isbn_libro> es clave foránea en Libro.

Cliente(nombre, nif, direccion_factura, telefono, correo, nombre_usuario, password)

Reserva(numero_reserva, numero_factura, fecha_reserva)
<numero_factura> es clave foránea en Factura.

Reserva_Detalle(idDetalle, numero_reserva, isbn_libro, unidades, precio)
<numero_reserva> es clave foránea en Reserva.
<isbn_libro> es clave foránea en Libro.

Envio(numero_envio, numero_factura, fecha_envio, estado_envio, observaciones)
<numero_factura> es clave foránea en Factura.

Envio_Detalle(idDetalle, numero_envio, isbn_libro, unidades, precio)
<numero_envio> es clave foránea en Envio.
<isbn_libro> es clave foránea en Libro.

Editorial(nombre, nif, dirección, telefono, fax, correo)

Libro(isbn, titulo, autor, editorial, fecha_edicion, paginas, estado, unidades, reservadas, precio)
<editorial> es clave foránea en Editorial>

5. Programación

5.1. Codificación de los componentes.

Consideraciones previas.

Para la implementación de la aplicación "Libreria" he utilizado cuatro tipos de componentes, los cuales son: Enterprise JavaBeans, Interfaces, Web Servlets y un conjunto de clases que modelan los componentes de la aplicación y que se encuentran en un apartado que he denominado *libreriaComunes*.

La implementación ha sido desarrollada utilizando el IDE Eclipse con JBoss, por lo que las clases y descriptores relacionados con el EJB son generados a través de la creación de una configuración XDoclet. De este modo los *Interfaces* y los documentos xml de las carpetas *META-INF* y *WEB-INF* son creados automáticamente.

Implementación aplicación

Para la implementación he creado un Enterprise JavaBean denominado *VentaBean*, esta clase java implementa todos los métodos de negocio. En este caso los métodos de negocio se encargan principalmente del acceso a la base de datos para recuperar y almacenar información.

Ventas.

Esta implementación cuenta con dos Web Servlets, que son VentasServlet y FacturaServlet, los cuales se encargan respectivamente de los apartados de selección de libros y de la facturación de las ventas. Estas clases para comunicarse con los usuarios utilizan código html que van generado de forma dinámica según las distintas circunstancias de la aplicación.

Administración

Este apartado de la implementación cuenta con varios Web Servlets, que son: AdministraServlet, AltaLibroServlet, GestiónStockServlet, GestionEnvioServlet y GestionReservaServlet, que son las clases encargadas de generar la interfaz con el usuario en esta sección.

Por último dentro de la implementación hay un apartado de clases comunes, las cuales no son ni Web Servles ni EJB, que permiten modelar los restantes componentes de la aplicación y realizar labores auxiliares como sería el caso de la clase *MetodosHTML*.

Toda la implementación está pensada para utilizar código html como forma de comunicación con los usuarios. Este código es interpretado por un navegador en forma de páginas web. Dentro de la implementación hay una carpeta *docroot* que contiene la página web inicial *index.html* que carga el navegador para iniciar la aplicación en el modo de ventas y también contiene la página web *administracion.html* que permite iniciar la aplicación en modo administrador. A partir de este punto todo el código html es generado de forma dinámica por la aplicación.

5.2. Descripción de las clases.

Clase VentaEJB

Esta clase es un Enterprise Java Bean, y principal objetivo es la comunicación de la aplicación con la base de datos, y está formada por los siguientes métodos de negocio:

- Libro[] **ordena**(String seccion). Lee la tabla libros y devuelve un array formado con los

libros de la sección indicada.

- Libro **getLibro**(String isbn). Devuelve el libro indicado por su isbn.
- Libro[] **buscarLibros**(String sTitulo, String sAutor, String sEditorial, String sIsbn). Busca libros por su título, autor, editorial o ISBN. Devuelve un array de libros con los encontrados.
- FormaPago[] **getFormasPago**(). Devuelve un array con todas las formas de pago.
- FormaEnvio[] **getFormasEnvio**(). Devuelve un array con todas las formas de envío.
- float **getImporteFormaEnvio**(String envio). Devuelve el importe de una forma de envío.
- boolean **validarUsuario**(String usuario, String contrasena). Comprueba si existe ese usuario con esa contraseña en la base de datos.
- double[] **crearFactura**(CestaCompra laCesta, String usuario, String dirEntrega, String formaEnvio, String formaPago). Crea la factura de la compra. También si hay unidades en stock crea el envio y si no crea la reserva. Devuelve un array con los números de factura, envío y/o reserva.
- boolean **usuarioUnico**(String usuario). Comprueba si el nombre del usuario ya existe en la base de datos.
- void **guardaDatosUsuario**(String nombre, String nif, String direccion, String telefono, String correo, String usuario, String contrasena). Este método da de alta un nuevo usuario.
- boolean **validarAdministrador**(String administrador, String contrasena). Este método comprueba si existe el administrador y tiene la contraseña correcta.
- boolean **compruebaEditorial**(String nifEditorial). Este método comprueba si existe la editorial indicada por su nif.
- void **guardaLibro**(Libro libro). Este método almacena un nuevo libro en la tabla de libros.
- boolean **compruebaLibro**(String isbn). Este método comprueba si existe un libro buscándolo por su isbn.
- void **actualizaLibro**(Libro libro). Este método actualiza el libro indicado por su isbn.
- boolean **compruebaEnvio**(double numeroEnvio). Comprueba si existe el envío buscándolo por su número.
- Envío **getEnvio**(double numeroEnvio). Crea un objeto Envio a partir de sus datos en la base de datos, el envío lo busca por su número.
- Cliente **getClienteFactura**(double numeroFactura). Crea un objeto Cliente a partir de la identificación del cliente que figura en la factura.
- void **actualizaEnvio**(Envio envio). Actualiza los datos de un envío en la base de datos, buscando el envío por su número.
- boolean **compruebaReserva**(double numeroReserva). Comprueba si existe la reserva en la base de datos buscándola por su número.
- Reserva **getReserva**(double numeroReserva). Crea un objeto Reserva a partir de sus datos contenidos en la base de datos, buscándola por su número.
- double **creaEnvioReserva**(double numeroReserva, double numeroEnvio). Convierte una reserva en un envío, siempre que haya unidades disponibles. Devuelve el número de envío creado.

Clases: VentasServlet, FacturaServlet, AdministraServlet, AltaLibroServlet, GestionStockServlet, GestionEnvioServlet, GestionReservaServlet.

Todas estas clases son Servlets y su misión es tratar con el usuario, por lo que se encargan de generar el código html necesario para crear las pantallas de la aplicación y de tratar todas las acciones de los usuarios. Su principal método es el siguiente:

- void **doPost**(HttpServletRequest req, HttpServletResponse resp). Trata todas las acciones de los usuarios con el formulario ventas.

Clase CestaCompra

Esta clase representa la cesta de la compra, donde se van anotando los libros que seleccionan los clientes.

- void **nuevoLibro**(Libro elLibro). Añade un nuevo libro a la Cesta de la Compra.
- void **nuevoLibro**(Libro elLibro, int unidades). Añade un nuevo libro a la cesta indicando el número de unidades.
- int **length**(). Devuelve el número de libros que contiene la cesta.
- boolean **vacia**(). Comprueba si la cesta esta vacía.
- float **getImporte**(). Devuelve el importe de los libros que figuran en la cesta de la compra.
- void **setUnidades**(int index, int unidades). Modifica el número de unidades pedidas del libro indicado por index.
- void **eliminaLibros**(). Elimina de la cesta los libros con cero unidades pedidas.
- Libro **getLibro**(int indice). Devuelve del array de libros de la cesta de la compra, el libro indicado por indice

Clase Cliente

Como indica su nombre esta clase corresponde con la entidad cliente y se compone en exclusiva de métodos constructores, métodos set y métodos get.

Clase Documento

Esta es una clase abstracta que sirve de clase principal para las clases Factura, Envio y Reserva. Además de los métodos constructores, set y get tiene:

- void **addDetalle**(Detalle detalle). Añade un nuevo objeto Detalle al array de detalles que mantiene el documento.

Clase Detalle

Esta clase es utilizada por la clase Documento y modela las líneas de detalle de los documentos. Se compone en exclusiva de métodos constructores, métodos set y métodos get.

Clase DBConector

Esta es una clase de utilidad que utilizo para crear las conexiones con la base de datos.

- Connection **conectar**(). Conecta con la base de datos de la librería.

Clase Envio

Esta clase extiende la clase documento y modela los envíos. Se compone en exclusiva de métodos constructores, set y get.

Clase EstadoLibro

Esta es una clase de utilidad, que he creado por comodidad, para tratar los posibles estados de los libros. Se compone en exclusiva de métodos constructores, set y get.

Clase Factura extends Documento

Esta clase extiende la clase documento y modela las facturas. Se compone en exclusiva de métodos constructores, set y get.

Clase FormaEnvio

Esta es una clase que he creado por comodidad para tratar las formas de envío. Se compone en exclusiva de métodos constructores, métodos set y métodos get.

Clase FormaPago

Esta es una clase que he creado por comodidad para tratar las formas de pago. Se compone en exclusiva de métodos constructores, métodos set y métodos get.

Clase Libro

Esta clase modela la entidad libro.

- float **getTotal**(). Devuelve el importe total de las unidades existentes del libro.
- float **getTotalReservado**(). Devuelve el importe total de las unidades reservadas del libro.

Clase MetodosHTML

Esta clase es una clase de utilidad se compone exclusivamente de métodos estáticos que contienen el código html que permite construir las páginas web de la aplicación. La he creado para tener junto todo el código html en una sola clase.

Clase Reserva

Esta clase es una extensión de la clase documento y modela las reservas. Se compone en exclusiva de métodos constructores, set y get.

5.3. Creación de la base de datos

La base de datos ha sido creada utilizando el sistema gestor de bases de datos MySQL. La basae de datos se denomina *libreria* y el usuario que he utilizado y su contraseña son: *fernando 123456*.

Para crear la base de datos he utilizado la siguiente orden sql:

```
CREATE DATABASE libreria;
```

Para darle el control de la base de datos al usuario:

```
GRANT ALL ON libreria.* TO 'fernando'@'localhost'
IDENTIFIED BY '123456' WITH GRANT OPTION;
```

Una vez creada la base de datos y dados permisos al usuario fernando, me conecto a la misma utilizando la sentencia:

```
mysql -u fernando -p libreria
```

5.3.1 Creación de las tablas:

```
/* Cabeceras de facturas */
CREATE TABLE factura (
      numero_factura BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
      fecha DATE NOT NULL,
      nombre_cliente VARCHAR(30) NOT NULL,
      nif_cliente VARCHAR(9) NOT NULL REFERENCES cliente(nif),
      direccion_cliente VARCHAR(40) NOT NULL,
```

```
	direccion_entrega VARCHAR(40),
	forma_envio VARCHAR(25) NOT NULL,
	forma_pago VARCHAR(25) NOT NULL,
	PRIMARY KEY (numero_factura));

/* Líneas de detalle de las facturas */
CREATE TABLE factura_detalle (
	id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
	numero_factura BIGINT UNSIGNED NOT NULL
		REFERENCES factura(numero_factura),
	isbn VARCHAR(13) NOT NULL REFERENCES libro(isbn),
	unidades INT NOT NULL,
	precio FLOAT not null,
	iva FLOAT not null,
	PRIMARY KEY (id));

/* Clientes */
CREATE TABLE cliente(
	nombre VARCHAR(30) NOT NULL,
	nif VARCHAR(9) NOT NULL,
	direccion_factura VARCHAR(40) NOT NULL,
	telefono VARCHAR(9),
	correo VARCHAR(20),
	nombre_usuario VARCHAR(15),
	contrasena VARCHAR(15),
	PRIMARY KEY (nif));

/* Cabecera de la reserva */
CREATE TABLE reserva (
	numero_reserva BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
	numero_factura BIGINT UNSIGNED NOT NULL
		REFERENCES factura(numero_factura),
	fecha DATE NOT NULL,
	PRIMARY KEY (numero_reserva));

/* Líneas de detalle de la reserva */
CREATE TABLE reserva_detalle (
	id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
	numero_reserva BIGINT UNSIGNED NOT NULL REFERENCES
reserva(numero_reserva),
	isbn VARCHAR(13) NOT NULL REFERENCES libro(isbn),
	unidades INT NOT NULL,
	precio FLOAT NOT NULL,
	iva FLOAT NOT NULL,
	PRIMARY KEY (id));

/* Cabecera del envío */
CREATE TABLE envio (
	numero_envio BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
	numero_factura BIGINT UNSIGNED NOT NULL
		REFERENCES factura(numero_factura),
	fecha DATE NOT NULL,
	estado VARCHAR(30),
	observaciones TEXT,
	PRIMARY KEY (numero_envio));
```

```
/* Líneas de detalle del envío */
CREATE TABLE envio_detalle (
      id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
      numero_envio BIGINT UNSIGNED NOT NULL REFERENCES envio(numero_envio),
      isbn VARCHAR(13) NOT NULL REFERENCES libro(isbn),
      unidades INT NOT NULL,
      precio FLOAT NOT NULL,
      iva FLOAT NOT NULL,
      PRIMARY KEY (id));

/* Datos de la Editorial */
CREATE TABLE editorial (
      nombre VARCHAR(30) NOT NULL,
      nif VARCHAR(9) NOT NULL,
      direccion VARCHAR(40) NOT NULL,
      telefono VARCHAR(9) NOT NULL,
      fax VARCHAR(9),
      correo VARCHAR(20),
      PRIMARY KEY (nif));

/* Datos del Libro */
CREATE TABLE libro (
      isbn VARCHAR(13) NOT NULL,
      titulo VARCHAR(60) NOT NULL,
      autor VARCHAR(60),
      editorial VARCHAR(9) NOT NULL REFERENCES editorial(nif),
      fecha_edicion DATE,
      paginas INT UNSIGNED,
      estado ENUM('novedad', 'recomendado', 'proximamente', 'oferta')
            NOT NULL,
      unidades INT NOT NULL DEFAULT 0,
      reservadas INT NOT NULL DEFAULT 0,
      precio FLOAT DEFAULT 0,
      iva FLOAT DEFAULT 0,
      PRIMARY KEY (isbn));

/* Formas de envío */
CREATE TABLE forma_envio (
      envio VARCHAR(10) NOT NULL,
      descripcion VARCHAR(25) NOT NULL,
      importe FLOAT NOT NULL,
      PRIMARY KEY (envio));

/* Formas de pago */
CREATE TABLE forma_pago (
      pago VARCHAR(10) NOT NULL,
      descripcion VARCHAR(25) NOT NULL,
      PRIMARY KEY (pago));


/* Administradores librería */
CREATE TABLE administrador(
      nombre VARCHAR(30) NOT NULL,
      telefono VARCHAR(9),
      correo VARCHAR(20),
      nombre_administrador VARCHAR(15),
      contrasena VARCHAR(15),
      PRIMARY KEY (nombre_administrador));
```

6. Pruebas y Despliegue

6.1. Juego de pruebas

He cargado los siguientes datos en la base de datos para poder realizar las pruebas.

```
/* Datos de prueba de la base de datos */
/* Tabla administradores */
INSERT INTO administrador VALUES (
      'José Nuñez Vázquez', '666555444', 'pepe@mail.es', 'pepe', '1234');
INSERT INTO administrador VALUES (
      'Esther Rodeiro Lamas', '999888777', 'esther@mail.es', 'esther',
'1234');

/* Tabla clientes */
INSERT INTO cliente VALUES (
      'Eduardo Lopez Lopez', '12312312H', 'Vigo, C/ Principal, 18 5Â°C',
      '555666777', 'edu@correo.es', 'eduardo', '1234');
INSERT INTO cliente VALUES (
      'María Soler Soler', '23423423J', 'CoruÃ±a, C/ Gran Vía, 434 8Â°H',
      '888777888', 'maria@correo.es', 'maria', '1234');
INSERT INTO cliente VALUES (
      'Julio Correa Vega', '34534534F', 'Vigo, C/ Terra, 23 1Â°I',
      '999888999', 'julio@correo.es', 'julio', '1234');

/* Tabla editoriales */
INSERT INTO editorial VALUES (
      'Anallas', 'B12345678', 'Vigo, C/ Provincial, 25-bajo',
      '999888111', NULL, NULL);
INSERT INTO editorial VALUES (
      'MasLibros', 'B87654321', 'Ourense, C/ Vella, 3-entresuelo',
      '777883774', '342342342', 'pedidos@maslibros.es');

/* Tabla libros */
INSERT INTO libro VALUES (
      '84-415-1031-8', 'El libro del Windows 2000', 'Varios',
      '777883774', '2001-09-25', 832, 'oferta', 3, 0, 25.13, 16);
INSERT INTO libro VALUES (
      '84-415-1000-8', 'El libro del Windows 95', 'Varios',
      '777883774', '1997-10-13', 420, 'oferta', 2, 0, 15.75, 16);
INSERT INTO libro VALUES (
      '84-7614-073-8', 'Nuevos viajes', 'Miguel Guerra',
      '999888111', '2006-10-20', 450, 'novedad', 10, 0, 30.90, 16);
INSERT INTO libro VALUES (
      '84-7714-074-8', 'Cross', 'James Paterson',
      '999888111', '2006-09-15', 278, 'novedad', 5, 0, 15.29, 16);
INSERT INTO libro VALUES (
      '84-7714-078-8', 'The inocent man', 'John Grishman',
      '999888111', '2006-04-15', 249, 'recomendado', 2, 0, 15.90, 16);
INSERT INTO libro VALUES (
      '84-745-4514-8', 'The god delusion', 'Richard Dawkins',
      '777883774', '2006-06-06', 300, 'recomendado', 2, 0, 14.85, 16);
INSERT INTO libro VALUES (
      '84-745-5845-8', 'Dear John', 'Nicholas Sparks',
      '777883774', '2006-12-15', 220, 'proximamente', 0, 0, 13.90, 16);
```

```
/* Tabla formas de envío */
INSERT INTO forma_envio VALUES (
      'transporte', 'Agencia de transporte', 25.0);
INSERT INTO forma_envio VALUES (
      'urgente', 'Correo urgente', 15.0);
INSERT INTO forma_envio VALUES (
      'correo', 'Correo normal', 7.0);

/* Tabla formas de pago */
INSERT INTO forma_pago VALUES (
      'reenvolso', 'Contra reenvolso');
INSERT INTO forma_pago VALUES (
      'transfer', 'Transferencia');
INSERT INTO forma_pago VALUES (
      'tarjeta', 'Tarjeta de crÃ©dido');
```

6.2. Documentación de las pruebas realizadas

Seleccionando libros de la sección de Recomendados.

Librería NovoLibro

Novedades | Recomendados | Prox. Publicación | Ofertas | Buscar

Libros

Sel.	ISBN	Título	Autor	Páginas	Precio
☑	84-745-4514-8	The god delusion	Richard Dawkins	300	14.85 €
☐	84-7714-078-8	The inocent man	John Grishman	249	15.90 €

Añadir a la Cesta de la Compra | Finalizar la compra

Finalizando la compra. Factura provisional.

Librería NovoLibro

Nombre usuario: maria Contraseña: ****

Alta Nuevo Usuario

Detalle de la compra

Libro	PVP	IVA	Unidades	Total	Disponible
Dear John. Nicholas Sparks	13.90 €	16.00 %	1	16.12 €	No
The god delusion. Richard Dawkins	14.85 €	16.00 %	1	17.23 €	Si
The inocent man. John Grishman	15.90 €	16.00 %	1	18.44 €	Si
			Subtotal	51.79 €	
Forma de envío: Agencia de transporte			Gastos envio	25.00 €	
Forma de pago: Contra reenvolso			TOTAL	76.79 €	

Escriba aquí su dirección de entrega si es diferente a la de la factura:

Recalcular Factura | Validar Factura

Alta de un nuevo cliente.

Librería NovoLibro

Rellena este formulario para darte de alta como cliente.

Nombre: María López López
NIF: 35456456H
Dirección: c/Gran Vía, 165 - 8ºA
Teléfono: 665665665
Correo: maria@yahoo.es
Nombre usuario: maria
Contraseña: ****

Aceptar

Detalle de la compra

Libro	PVP	IVA	Unidades	Total	Disponible
Dear John, Nicholas Sparks	13,90 €	16,00 %	1	16,12 €	No

Compra finalizada.

Librería NovoLibro

Ha finalizado correctamente su proceso de compra.

Tome nota de los siguientes datos:
Número de factura: 1
Número de envío: 1
Número de reserva: 1

Pulse aquí para seguir comprando.

Acceder al área de administradores.

Librería NovoLibro

Administración

Nombre administrador: pepe
Contraseña: ****
Aceptar

Alta nuevo libro.

Librería NovoLibro

Alta nuevo libro

Título: The J2EE Architect
Autor: Drek C. Ashmore
Editorial: B12345678
ISBN: 0-9729548-9-7 Fecha edición: 16/10/2006
Páginas: 284
Estado: novedad
Unidades: 5 Reservados: 0
Precio: 34.95 IVA: 16.00
Aceptar Cancelar

Gestión del estado del libro y del stock.

Librería NovoLibro

Gestión estado libro y stock

ISBN: 0-9729548-9-7 Buscar libro
Título: The J2EE Architect
Autor: Drek C. Ashmore
Editorial: B12345678
Estado: novedad
Unidades: 5 Reservados: 0
Precio: 34.95 IVA: 16.00
Aceptar Cancelar

Gestionar el envío.

Librería NovoLibro

Gestionar envío

Número envío: 1 Buscar envío
Cliente: MarÃa Soler Soler, nif: 23423423J
Fecha de envío: 25/11/2006
Estado: enviado
Observaciones: Portes pagados
Aceptar Cancelar

Gestionar reserva.

Librería NovoLibro

Gestionar reserva

Número reserva: 2 [Buscar reserva]

Cliente: MarÃa Soler Soler, nif: 23423423J

Fecha de reserva: 25/11/2006

Detalle libros reservados:

Libro	Uds.Solicitadas	Uds.Disponibles
The god delusion, Richard Dawkins	1	1
The inocent man, John Grishman	3	0
Dear John, Nicholas Sparks	1	0

[Enviar Unidades Disponibles]

6.3. Despliegue de la aplicación.

Esta aplicación fue realizada utilizando el entorno de programación Eclipse con el plugin de Jboss. Una vez compilada la aplicación se obtiene un archivo EAR que contiene toda los archivos de la aplicación empaquetados y listos para desplegar.

Para el despliegue de la aplicación he utilizado el servidor de aplicaciones de Jboss en su versión 4.0.5. Una vez instalado este servidor hay que copiar el archivo EAR, en mi caso el archivo es 'VentaApp.ear', en la carpeta 'deploy' del servidor.

Para poder acceder a la base de datos en MySQL también hay que instalar el conector jdbc de mysql 'mysql-connector-java-5.0.4-bin.jar' en la carpeta /server/default/lib.

Además hay que hacer que el servidor de aplicaciones tome por defecto el jdbc de mysql. Para hacer esto hay que crear un archivo 'mysql-ds.xml' que tendremos que copiar en la carpeta deploy. Además hay que modificar el archivo 'hsqldb-ds.xml' que esta configurado con el conector de base de datos que trae el servidor de aplicaciones por defecto.

También hay que tener instalada la versión 5 de Java.

Contenido del archivo 'mysql-ds.xml':

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 $Id: mysql-ds.xml 41016 2006-02-07 14:23:00Z acoliver $
 -->
- <!--
  Datasource config for MySQL using 3.0.9 available from:
http://www.mysql.com/downloads/api-jdbc-stable.html
 -->
- <datasources>
- <local-tx-datasource>
  <jndi-name>DefaultDS</jndi-name>
  <connection-url>jdbc:mysql://localhost:3306/libreria</connection-url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <user-name>fernando</user-name>
```

```
 <password>123456</password>
 <exception-sorter-class-
name>org.jboss.resource.adapter.jdbc.vendor.MySQLExceptionSorter</exception-sorter-
class-name>
- <!--
 should only be used on drivers after 3.22.1 with "ping" support
    <valid-connection-checker-class-
name>org.jboss.resource.adapter.jdbc.vendor.MySQLValidConnectionChecker</va
lid-connection-checker-class-name>

 -->
- <!--
 sql to call when connection is created
    <new-connection-sql>some arbitrary sql</new-connection-sql>

 -->
- <!--
 sql to call on an existing pooled connection when it is obtained from pool
- MySQLValidConnectionChecker is preferred for newer drivers
    <check-valid-connection-sql>some arbitrary sql</check-valid-connection-
sql>

 -->
- <!--
 corresponding type-mapping in the standardjbosscmp-jdbc.xml (optional)
 -->
- <metadata>
 <type-mapping>mySQL</type-mapping>
 </metadata>
 </local-tx-datasource>
 </datasources>
```

En el archivo 'hsqldb-ds.xml' hay que modificar la línea *<jndi-name>DefaultDS</jndi-name>* por *<jndi-name>OLDDefaultDS</jndi-name>*.

Una vez realizadas todas estas configuraciones ya se puede ejecutar el servidor de aplicaciones.

Ventas

Para acceder a la aplicación en la sección de ventas de libros hay que abrir un navegador y escribir el siguiente URL: 'http://localhost:8080/venta' (localhost o la ip de la máquina que este ejecutando el servidor de aplicaciones).

Administración

Para acceder a la aplicación en la sección de administración ha que abrir también un navegador y escribir el siguiente URL: 'http://localhost:8080/venta/administracion.html' (localhost o la ip de la máquina que este ejecutando el servidor de aplicaciones).

7. Conclusiones

Java EE, y las herramientas que lo rodean, nos permite de una forma sencilla crear aplicaciones Web centrándonos casi exclusivamente en el diseño de negocio, si tener prácticamente en cuenta el entorno en el que van a ser desplegadas. Esto permite un gran ahorro de tiempo y trabajo a la hora de desarrollar las aplicaciones.

Además yo creo que la principal ventaja que tienen estas aplicaciones es que casi no hay que preocuparse por el entorno en el que va a trabajar el cliente de la misma, ya que a este le basta con tener un acceso a Internet y un navegador para poder acceder a la aplicación. Con el beneficio añadido de que el cliente siempre va a trabajar en un entorno que el ya conoce.

La aplicación que he creado en este trabajo creo que, aunque es sencilla, ilustra perfectamente lo que indico en los párrafos anteriores. Centrándome básicamente en el diseño y desarrollo del modelo de negocio, he construido una aplicación que se despliega sin ningún problema en la red y puede ser accedida por cualquier cliente simplemente con un navegador.

Otra de las cuestiones que yo considero muy importante es que todo este trabajo se puede hacer usando únicamente software libre y de código abierto. Y que estas herramientas que he utilizado: entorno de programación (Eclipse), servidor de aplicaciones (JBoss), base de datos (MySQL) y por supuesto el Java EE; se encuentran disponibles para distintas plataformas (Linux, Windows, etc.)

Dentro de este apartado de conclusiones también quiero comentar las mejoras que se le podrían hacer a la aplicación que he desarrollado.

Por supuesto la aplicación es bastante sencilla por lo que en este aspecto las mejoras que se pueden implementar son múltiples, tanto en el área de ventas como en el área de administración. Debo aclarar que yo no he perseguido hacer tanto una aplicación completa, teniendo en cuenta todos los aspectos que se podrían implementar en una librería virtual, si no una aplicación que me permitiera en primer lugar aprender JavaEE y en segundo lugar mostrar o demostrar lo aprendido.

En el apartado de programación yo personalmente creo que se debería mejorar la interfaz de usuario. Para construir esta interfaz, por motivos de falta de tiempo, he utilizado código html muy sencillo. Por lo que en primer lugar debería utilizar hojas de estilo css y utilizar páginas JSP.

8. Bibliografía

- JBoss Eclipse IDE Tutorial. An introduction and walkthroug of JBoss Eclipse IDE. 1.5.0
- Getting Started with JBoss 4.0. Release 5. JBoss, Inc.
- The JBoss 4 Aplication Server J2EE Reference. JBoss AS 4.0.5. Rlease 2. JBoss Inc.
- Manual de Referencia de MySQL versión 5
- Mastering Enterprise JavaBeans.Ed Roman. Wiley Publishing, Inc.
- The J2EE Architect's Handbook. Derek C. Ashmore. DVT Press.
- Google