Universitat Oberta de Catalunya
Eng. Técnica en Informática de Sistemas

TRABAJO FIN DE CARRERA:

Análisis, diseño e implementación de una tienda virtual con tecnología J2EE

Nombre del estudiante:

Marisol Rodríguez Goyanes

Nombre del consultor:
José Juan Rodríguez

Fecha de entrega:
14 de Enero del 2008

1 DEDICATORIA Y AGRADECIMIENTOS

Quisiera expresar en estas líneas mi agradecimiento a todas aquellas personas que han estado a mi lado todos estos años, por su paciencia y compresión.

Gracias a todos los compañeros de trabajo por entender los momentos de la UOC, a los amigos por entender que únicamente podía salir según la entrega de las pacs, y a la familia por entender la ausencia en reuniones y aniversarios.

Quisiera transmitir un especial agradecimiento a Anna, por motivarme y hacerme entender que todo es posible cuando se intenta.

Por ultimo, a Stephane por soportar todos estos años de estudio, con paciencia y compresión, dándome soporte y ánimos.

2 Resumen

El TFC o trabajo final de carrera es la culminación de los estudios de Ingeniería Técnica de Informática de Sistemas, en el cual se irán aplicando los diferentes conocimientos adquiridos a lo largo de la carrera.
Tiene como finalidad profundizar en los conocimientos de plataformas tecnológicas que permiten el desarrollo de soluciones e-business, y conocer el negocio del comercio electrónico, en este TFC se propone desarrollar una aplicación de comercio electrónico usando la tecnología J2EE.

El vertiginoso desarrollo de la informática y las comunicaciones donde Internet y Intranet son lo máximo, y a la vez con una mayor demanda de la empresa por aplicaciones de calidad que den soluciones a sus necesidades, ha hecho que java como lenguaje de desarrollo, juntamente con la arquitectura J2EE se hayan convertido en un Standard en el mundo de la industria para el desarrollo distribuido de aplicaciones empresariales en internet.

La idea principal de este modelo es la arquitectura multi-capas, donde la interface del usuario, la lógica de negocio y las base de datos se separan por componentes. La descomposición de la aplicación se caracterizan por:

- Proporcionar escalabilidad, capacidad de administración y uso de recursos mejorados.
- Cada capa es un grupo de componentes que realiza una función especifica.
- Se puede utilizar una capa sin recompilar otras capas.

Para el desarrollo de la aplicación de comercio electrónico de este TFC, una librería virtual llamada newBooks, presentaremos 3 capas:

Capa de presentación, será la interface grafica que muestra los datos al usuario.
Capa lógica de negocio será la responsable de procesar los datos recuperadas y enviarlas a la capa de presentación.
Capa de datos, será donde se encuentre la base de datos relacional.

3 Índice

3.1 Índices y contenidos

3.2 Índices de figuras

4 Memoria

4.1 Introducción

4.1.2 Motivación del proyecto

El mercado actual de las aplicaciones Web no deja de sorprendernos. Cada vez más aparecen nuevas herramientas que hacen que la construcción y diseño de un entorno Web sea más fácil y rápido de hacer. En un mundo competitivo como en el que vivimos donde todo tiene que estar al momento ("just in time"), son de agradecer las herramientas de desarrollo que siguen los conocidos patrones de diseño y que permiten implementar aplicaciones de una manera rápida y estructurada. Pero no solo eso, sino que además de todo se desea que las estructuras de las aplicaciones sean extensibles, es decir, no nos sirve de nada que empleemos cientos de horas en fijar los cimientos de un proyecto si luego éstos no nos van a servir para uno futuro.
Este proyecto se utiliza la tecnología J2EE y se basa en Struts, uno de los Frameworks de desarrollo para aplicaciones Web más utilizados dentro del mundo empresarial.
La intención del proyecto es poder demostrar como Struts puede facilitar y decrementar el tiempo de producción de una aplicación. Para ello se ha decidido implementar una aplicación de comercio electrónico.

En este TFC, un estudiante con conocimientos de Java, pero sin experiencia previa en aplicaciones distribuidas, diseña e implementa un ejemplo típico de una aplicación distribuida que consiste en una aplicación de comercio electrónico utilizando la tecnología J2EE y con el seguimiento de patrones de diseño.

4.1.3 Descripción y objetivos del proyecto

Tiene como finalidad profundizar en el conocimiento de las plataformas tecnológicas que permiten el desarrollo e-business y conocer el negocio del comercio electrónico. El objetivo al desarrollarlo es no construir un producto explotable comercialmente, sino llegar a entender la tecnología J2EE.
Este objetivo se concreta en lo siguiente:

- Analizar un problema complejo de tipo práctico transformándolo en un proyecto informático.

- Planificar y estructurar el desarrollo del proyecto mediante la elaboración de un plan de trabajo aplicando la metodología adecuada.
- Profundizar en los aspectos formales del desarrollo del proyecto.
    - Uso del Patrón MVC:
      Permite mantener por separado las funcionalidades de vista (diseño), modelo (datos, objetos) y controlador (lógica de negocio).
      Además permite que se le añadan mejoran en una de las capas sin que las otras deban ser modificadas. Este patrón se implementa en el proyecto desarrollado dentro del grupo Apache - Jakarta denominado Struts.

    - Servidor de aplicaciones.
      Gracias a las especificaciones que se han establecido para la arquitectura J2EE, es posible diseñar aplicaciones que van a correr en entornos Web. Tomcat es un servidor web que cumple con las especificaciones para ser un container de servlets y Jsps.

    - Capa de abstracción en BD
      De igual manera se ha demostrado la utilidad de no tratar de forma directa el acceso base de datos, que en tecnología java se realiza con la tecnología JDBC. Se trata de manejar siempre objetos, con el uso de frameworks que interactúan con las bases de datos para obtener una comunicación orientada a objetos pura, ya que estos frameworks nos devuelven como resultado, objetos con lo que podemos trabajar directamente, ya que ahora son clases de nuestro diseño.

- Sintetizar una solución viable y realista del problema propuesto.
- Elaborar una memoria del proyecto según una estructura prefijada.
- Elaborar una presentación del desarrollo y resultados finales del proyecto.

La aplicación consta de 3 partes diferenciadas debido a los diferentes tipos de usuarios:

- Usuario publico, en un área publica donde cualquier usuario sin necesidad de estar validado además de poder consultar el catalogo de libros disponibles en la web podrá simular una compra, añadiendo o borrando productos y especificando las unidades que desea, pero no podrá comprar.
- Un usuario registrado y identificado:
    - Además de las funcionalidades de búsqueda del producto podrá hacer simulaciones de compra, añadiendo o borrando productos y especificando las unidades que desea.
    - Podrá consultar y modificar en cualquier momento el contenido de la cesta de compra y aceptarla.
    - Podrá darse de baja y hacer consulta y modificaciones de sus datos personales. No se podrá realizar la baja si existe algún pedido en estado.

- Administrador donde solo podrá acceder el administrador de la aplicación donde podrá gestionar todo lo referente al producto (altas, bajas, modificaciones, etc) y errores de compra producidos.

4.1.4 Método a seguir

Para el desarrollo del proyecto se seguirá la metodología Rational Unified Process (RUP), es un ciclo de vida iterativo e incremental.

RUP divide el proceso de desarrollo en los siguientes ciclos:

- Planificación, descripción del proyecto, objetivos plan de trabajo y funcionalidades.
- Análisis de requisitos, identificación de subsistemas, diagramas de casos de uso, prototipos de la aplicación.
- Diseño, representación grafica de los subsistemas, diagramas de clases y jerarquías en UML, relación de clases, diagramas de colaboración, diseño de la persistencia, variaciones del prototipo de la interficíe grafica.
- Implementación del software se ponen en práctica las tecnologías J22E y páginas JSP.
- Testing.

4.1.5 Planificación del proyecto

La planificación va definida en función de las pacs. Las fechas son:

- 28/09/2007 Pac1 – Plan de trabajo
- 29/10/2007 Pac2 – Análisis y Diseño
- 17/12/2007 Pac3 – Implementación y Testing
- 14/01/2008 Entrega Final – Documento Final

La planificación de cada tasca y la temporización vienen detallada en el siguiente diagrama de Gantt.

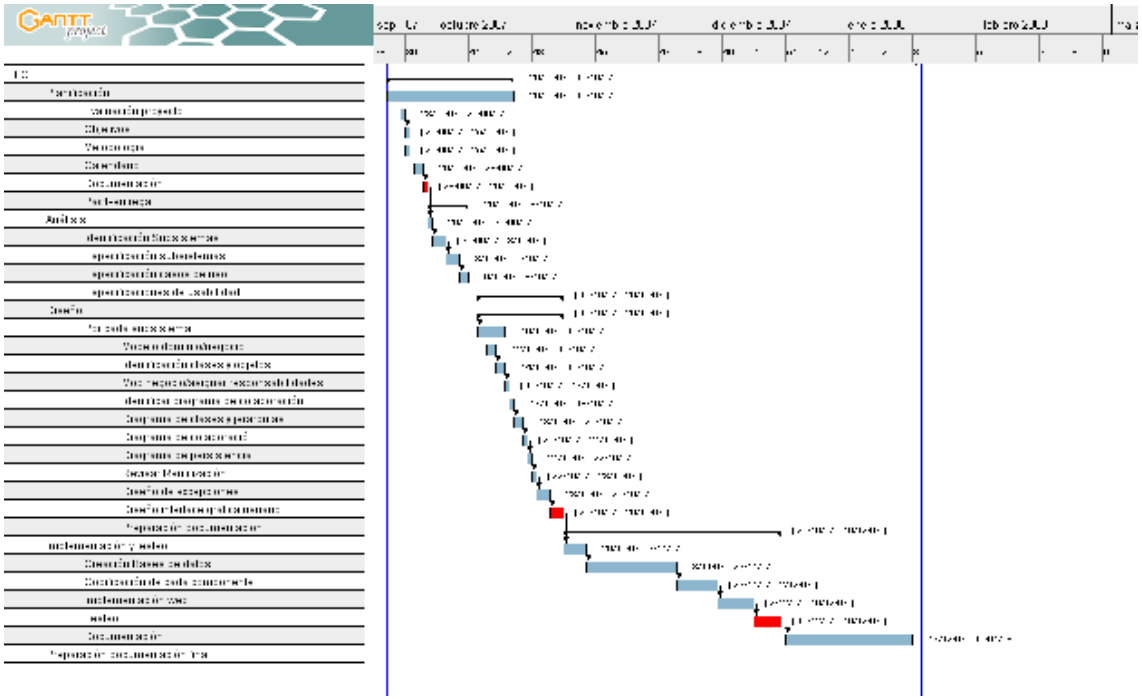


Figura 1. Diagrama de Gantt . TFC NewBooks

4.2 Análisis

4.2.1 Definición del concepto operacional

4.2.1.1 Objetivos

El software **NewBooks** es una aplicación con la funcionalidad de una tienda electrónica, es decir una librería. El principal objetivo de esta librería es aprender a utilizar la tecnología J2EE.

La funcionalidad de la tienda electrónica se ofrecerá a cualquier usuario conectado a través de Internet usando cualquier navegador. A de más de ofrecer la posibilidad de comprar a un usuario debidamente registrado, se ofrece los servicios de mantenimiento del administrador.

4.2.1.2 Composición del software

El software consistirá en una parte servidora implementada con la tecnología J2EE, ubicada en una sede central, un cliente ligero que corresponde a los compradores, que accederán a la tienda desde Internet usando cualquier navegador convencional y otro cliente el cual realiza las tareas de mantenimiento.
El proyecto se va a realizar utilizando el lenguaje de programación Java en su especificación J2EE. Otra característica importante de la implementación es la utilización del Framework Struts ya que sigue el patrón MVC que nos permitirá agilizar el desarrollo y el posterior mantenimiento de la aplicación. También se pretende utilizar en la medida que sea posible el máximo de tecnologías de libre distribución.

4.2.1.3 Requisitos funcionales

- **Catálogo de productos organizados por materia**

La aplicación debe permitir al cliente poder navegar por las diferentes materias viendo los productos que hay en cada una de ellas.

- **Búsqueda de productos.**

También se va a contar con un buscador de productos que permita buscar directamente productos. A partir de la búsqueda se mostrará un listado de productos que cumplan los criterios seleccionados.

- **Cesta de la compra.**

Un cliente podrá seleccionar un producto y añadirlo a la cesta de la compra. Esta cesta contará con todas las funciones típicas (añadir elemento, borrar)

- **Área privada: Inserción, modificación y eliminación de productos.**

La aplicación implementará un área privada donde los administradores de la aplicación contarán con una herramienta que servirá para añadir nuevos productos y materias, modificar características de los ya existentes, o eliminar productos y materias.

4.2.1.4 Requisitos no funcionales

Se pide específicamente que el diseño y la implementación de la tienda virtual ha de ser en arquitectura de 3 capas, usando J2EE, donde se separe claramente la presentación, la lógica de negocio y el acceso a los datos. Se maximiza el uso estándar y la persistencia de los datos es una BD relacional. La escalabilidad fiabilidad y portabilidad son inherentes en la tecnología J2EE.
Se utilizara un protocolo de acceso seguro HTTPS para acceder a la aplicación.

4.2.2 Casos de uso

4.2.2.1 Identificación de los actores

Cada actor tiene un papel para cada caso de uso en el cual interviene.
Después de ver las funcionalidades del software podemos detectar claramente 3 actores con roles diferentes, usuario anónimo, usuario registrado y administrador.
El usuario anónimo no se ha identificado en el sistema, por lo que solo puede hacer consultas, es la parte publica y simular una compra.
Sus funciones son:

- Alta usuario
- Login
- Consultas catálogo
- Añadir, eliminar y consultar cesta (simular compra)

Usuario registrado o cliente, esta identificado en el sistema. Tiene acceso a una parte restringida del software. Hereda todas las funcionalidades propias del usuario anónimo, y a demás podrá:

- Confirmar compra
- Consulta, modificación y baja usuario
- Ver pedidos pendientes

Super usuario o administrador, se identifica con el rol de administrador, se encargara del mantenimiento de la tienda virtual. Tiene acceso a toda la parte privada del software. No hereda las funcionalidades de los otros dos actores,

ya que si quiere hacer una compra se tiene que registrarse como usuario registrado (cliente). Sus funcionalidades son:

- Alta, baja, modificar materia
- Alta, baja, modificar catalogo productos

En la relación de los actores, vemos que usuario registrado es una especialización de usuario anónimo.

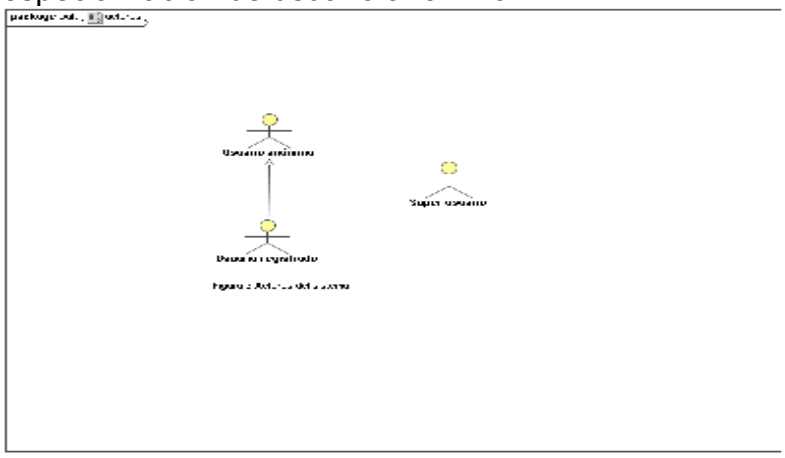


Figura 2. Actores del sistema

4.2.2.2 Relación entre casos de uso y actores. Diagrama

Los casos de uso se clasifican en función del perfil de usuario, donde cada usuario tiene acceso a diferentes partes de la aplicación.

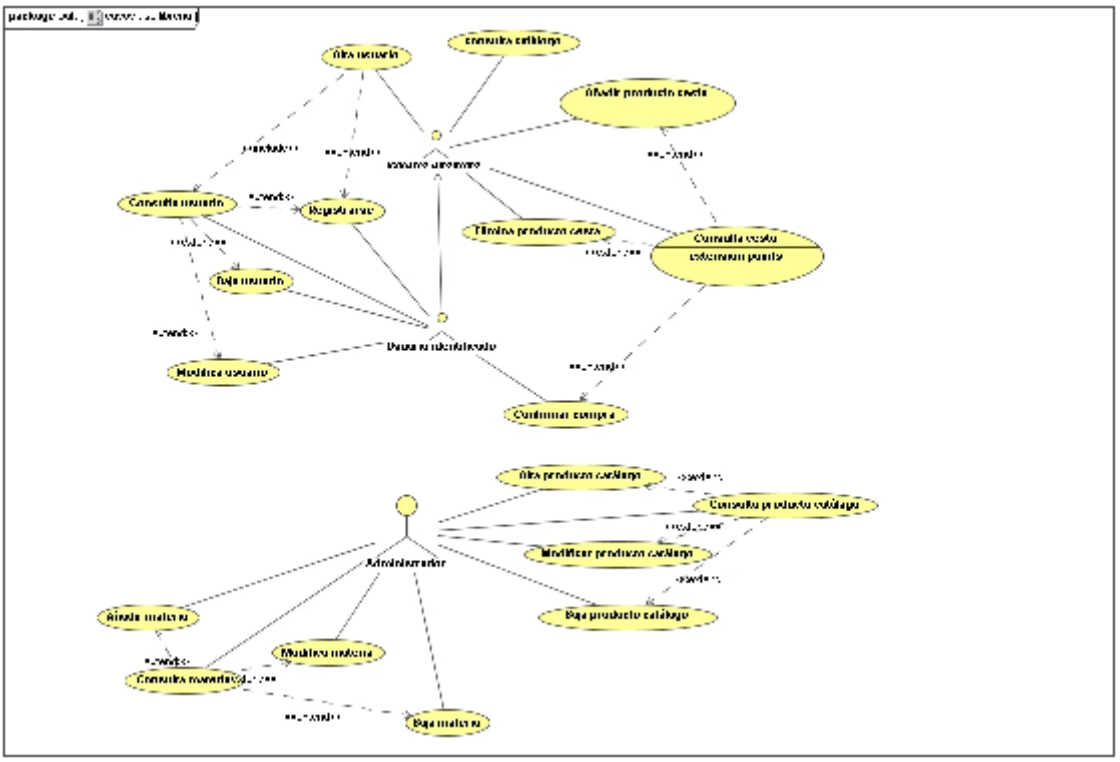


Figura 3. Casos de uso de la aplicación.

4.2.2.2.1Descripción textual de casos de uso

Caso de uso: Alta usuario

Resumen de la funcionalidad: Da de alta un usuario en la base de datos, se mira que no exista en la BBDD, introducir los datos del usuario.
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario anónimo
Casos de usos relacionados: Registrarse, consulta usuario,
Precondición: El usuario no puede existir en la base de datos.
Poscondición: Se crea un nuevo usuario.
Flujo de eventos principal:

- El usuario escoge la opción de Alta usuario
- El sistema le presenta un formulario a rellenar por el usuario donde le pide: Login, password, repetir password, nombre, apellidos, dirección, código postal, población, provincia, teléfono, email, cuenta corriente. Pulsar aceptar.
- El sistema almacenara el nuevo usuario en la BBDD con el perfil tipo “1” ya que es el perfil del cliente .

Flujos alternativos:

- Si el sistema detecta que el login existe lanza mensaje de error.
- Si el sistema detecta que el password es incorrecto lanza mensaje de error.
- Si el sistema detecta que falta algún dato obligatorio lanza mensaje de error.

Caso de uso: Baja usuario

Resumen de la funcionalidad: Da de baja un usuario en la base de datos.
Papel dentro del trabajo del usuario: Habitual

Actores: Usuario identificado
Casos de usos relacionados: Consulta usuario,
Precondición: El usuario ha de existir en la base de datos y no tener ningún pedido en estado pendiente.
Poscondición: El usuario se dará de baja en el sistema si no tiene ningún pedido en estado
Flujo de eventos principal:

- El usuario consulta sus datos y pulsa el eliminar.
- El sistema pondrá eliminara el registro.
- El usuario deja de estar registrado y pasa a la pagina principal.

Flujos alternativos:

Caso de uso: Consulta usuario
Resumen de la funcionalidad: Muestra los datos del usuario registrados en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario identificado
Casos de usos relacionados: Baja usuario, Alta usuario, Modifica usuario, Registrarse
Precondición: El usuario ha de existir en la base de datos.
Poscondición: El usuario registrado consulta sus datos registrados en la BBDD
Flujo de eventos principal:

- El usuario pulsa la opción consulta datos.
- El sistema mostrara los datos del usuario.

Flujos alternativos:
Caso de uso: Modifica usuario
Resumen de la funcionalidad: Modifica los datos del usuario registrados en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario identificado
Casos de usos relacionados: Consulta usuario.
Precondición: El usuario se ha de registrar y existe en la base de datos.
Poscondición: El usuario registrado Modifica sus datos almacenados en la BBDD
Flujo de eventos principal:

- El usuario esta consultando sus datos, edita los campos del formulario deseado, y pulsa la opción modificar datos.
- El sistema grabara los datos del usuario.

Flujos alternativos:

- Si el sistema detecta que falta algún campo obligatorio lanzara un mensaje de error.

Caso de uso: Registrarse
Resumen de la funcionalidad: verifica que el login y password existan en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario identificado
Casos de usos relacionados: Consulta usuario y Alta usuario
Precondición: El usuario ha de existir en la base de datos.

Poscondición: El sistema valida el usuario, si existe, le permite acceder a las opciones correspondientes al rol validado (Usuario registrado, Super usuario), si no existe el sistema le mostrara un mensaje informándole.
Flujo de eventos principal:

- El usuario introduce login y password escoge la opción de registrarse
- El sistema identificara el usuario y accederá a las funcionalidades según el Usuario registrado, Super usuario**.**

Flujos alternativos:

- Este caso de uso puede ser llamado desde Alta usuario.

Caso de uso: Consulta Catálogo
Resumen de la funcionalidad: Lista todos los productos correspondientes a un tema y materia.
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario anónimo, Usuario Identificado.
Casos de usos relacionados: Alta producto
Precondición: El tema y la materia han de existir.
Poscondición: Se obtiene una lista completa de todos los productos del filtro aplicado.
Flujo de eventos principal:

- El usuario selecciona un tema y materia disponibles en la combo.
- El sistema mostrara una lista de todos los productos que pertenecen al filtro seleccionado.

Flujos alternativos:

- El sistema mostrara un mensaje informando que no hay ningún producto que de la categoría seleccionada.

Caso de uso: Añadir producto cesta
Resumen de la funcionalidad: Añadir producto a la cesta del usuario
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario anónimo, Usuario Identificado.
Casos de usos relacionados: Consulta cesta
Precondición: N/A
Poscondición: El usuario incorpora un producto a la cesta, indicando el nº de unidades por producto.
Flujo de eventos principal:

- El usuario tiene una lista de productos. Pulsa añadir, y podrá especificar las unidades deseadas del producto.
- **Flujos alternativos**:

Caso de uso: Elimina producto cesta
Resumen de la funcionalidad: Elimina producto a la cesta del usuario
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario anónimo, Usuario Identificado.
Casos de usos relacionados: Consulta cesta
Precondición: El producto ha de existir en la cesta
Poscondición: El usuario elimina un producto a la cesta.
Flujo de eventos principal:

- El usuario consulta la cesta. Pulsa eliminar a la línea correspondiente del producto en cuestión.
- El sistema elimina el producto de la línea de la cestilla del usuario.
- **Flujos alternativos**:

Caso de uso: Consulta producto cesta
Resumen de la funcionalidad: Consulta de productos en la cesta del usuario
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario anónimo, Usuario Identificado.
Casos de usos relacionados: Elimina producto cesta, Añadir producto cesta
Precondición: N/A

- **Poscondición**: El sistema mostrará la información de todos los productos incorporada la cestilla del usuario.

Flujo de eventos principal:

- El usuario consulta la cesta. Pulsa consultar cesta.
- El sistema mostrara la información de todos los productos incorporada la cestilla del usuario.
- **Flujos alternativos**:

Caso de uso: Confirmar compra
Resumen de la funcionalidad: Crea pedido y se registra a la base de datos
Papel dentro del trabajo del usuario: Habitual
Actores: Usuario Identificado.
Casos de usos relacionados: Consulta producto cesta
Precondición: Ha de existir al menos un producto en la cesta, y el usuario ha de estar identificado, si no esta identificado se le presenta la pantalla de login.

- **Poscondición**: El usuario registrado hace un pedido de todos los productos incorporada la cestilla.

Flujo de eventos principal:

- El usuario consulta la cesta. Pulsa confirmar pedido.
- El sistema genera un pedido, que incluye todos los productos incorporada la cestilla del usuario, también se registrara el usuario de la compra .

Flujos alternativos

Caso de uso: Alta producto catálogo
Resumen de la funcionalidad: Dar de alta un producto en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Super usuario.
Casos de usos relacionados: Consulta producto catálogo.
Precondición: El administrador ha de estar registrado. El producto no ha de existir en la base de datos, y un producto solo puede pertenecer a una materia y esta ha de existir.
Poscondición: El sistema ha incorporado un producto a base de datos.
Flujo de eventos principal:

- El administrador pulsa alta producto.
- El sistema muestra un formulario del producto.
- El administrador introduce los datos del producto: autor, titulo, ISBN, idioma, materia, precio, descuento, stock. Pulsa aceptar.
- El sistema incorpora un producto en la base de datos.
- **Flujos alternativos**: Si el sistema detecta que falta por rellenar algún campo mostrara un mensaje.

Caso de uso: Baja producto catálogo
Resumen de la funcionalidad: Dar de baja un producto en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Super usuario.

Casos de usos relacionados: Consulta producto catálogo.
Precondición: El administrador ha de estar registrado. El producto ha de existir en la base de datos.
Poscondición: El sistema ha eliminado el producto de la base de datos.
Flujo de eventos principal:

El administrado busca el producto a dar de baja.

- El administrador tiene el producto en la pantalla.
- El administrador pulsa Eliminar producto.
- El sistema elimina un producto de la base de datos.

Flujos alternativos:

Caso de uso: Modifica producto catálogo
Resumen de la funcionalidad: Modifica los datos de un producto a excepción del identificador.
Papel dentro del trabajo del usuario: Habitual
Actores: Super usuario.
Casos de usos relacionados:
Precondición: El administrador ha de estar registrado. El producto ha de existir en la base de datos, y un producto solo puede pertenecer a una materia y esta ha de existir.
Poscondición: El sistema ha incorporándolos cambios del producto a la base de datos.
Flujo de eventos principal:

- El administrador pulsa Modifica producto.
- El sistema muestra un formulario del producto.
- El administrador modifica los datos del producto, todos los campos son editables excepto el id del producto Pulsa Modificar.
- El sistema incorpora los cambios del producto en la base de datos.

Flujos alternativos:

- Si el sistema detecta que falta por rellenar algun campo mostrara un mensaje.

Caso de uso: Alta materia
Resumen de la funcionalidad: Dar de alta una materia en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Super usuario.
Casos de usos relacionados: Consulta materia.
Precondición: El administrador ha de estar registrado. La materia no ha de existir en la base de datos, y una materia solo puede pertenecer a tema y este ha de existir.
Poscondición: El sistema ha incorporado un materia a base de datos.
Flujo de eventos principal:

- El administrador pulsa alta materia.
- El sistema muestra un formulario de la materia.
- El administrador introduce la materia y nombre. Pulsa aceptar.
- El sistema incorpora una materia en la base de datos.
- **Flujos alternativos**: Si el sistema detecta que falta por rellenar algún campo mostrara un mensaje.

Caso de uso: Baja materia
Resumen de la funcionalidad: Dar de baja una materia en la base de datos.

Papel dentro del trabajo del usuario: Habitual
Actores: Super usuario.
Casos de usos relacionados: Consulta materia.
Precondición: El administrador ha de estar registrado. La materia ha de existir en la base de datos.
Poscondición: El sistema ha eliminado la materia de la base de datos.
Flujo de eventos principal:

- El administrador tiene la lista de las materias en la pantalla.
- El administrador pulsa Eliminar materia.
- El sistema elimina una materia de la base de datos.

Flujos alternativos:

Caso de uso: consulta materia
Resumen de la funcionalidad: Muestra un listado con todas las materias existentes en la base de datos.
Papel dentro del trabajo del usuario: Habitual
Actores: Super usuario.
Casos de usos relacionados:
Precondición: El administrador ha de estar registrado
Flujo de eventos principal:

- El administrador pulsa la opción Consulta una materia.
- El sistema muestra un una lista con todas las materias existentes.

Flujos alternativos:

- Si no hay ninguna materia el sistema muestra un mensaje informándolo.

4.2.3 Requisitos de la interficie de usuario

4.2.3.1 Perfiles de usuario

Los diferentes perfiles que nos encontramos son usuario anónimo (no registrado), usuario registrado (cliente), y super usuario (administrador del sistema).

El usuario anónimo puede ser cualquier persona que se conecte desde cualquier sitio a la página web newBooks. Este perfil podrá consultar el catalogo y simular una compra, pero no podrá comprar. Por lo tanto podemos encontrarnos dentro de este perfil cualquier usuario con diferentes niveles de conocimientos informáticos.

El usuario registrado, usuario identificado por el sistema, el cual dispone de los datos personales registrados. Tendrá las mismas funcionalidades que un usuario anónimo, pero a demás podrá confirmar la compra, consultar y modificar sus datos personales. Igual que el perfil anterior podemos encontrar usuarios con diferentes niveles de conocimientos informáticos.

Finalmente el Super usuario (administrador del sistema), tendrá acceso a un área restringida previa identificación en el sistema. Podrá hacer el

mantenimiento del catálogo. Únicamente estos son expertos en la aplicación y el uso del ordenador.

4.2.3.2 Requisitos de usabilidad

El software newBooks ha desarrollar tiene 2 áreas bien diferenciadas, parte publica y parte privada.
Los usuarios anónimos y usuarios registrado, que tienen acceso a la parte publica pueden ser personas poco familiarizadas con internet y tiendas virtuales. Por lo tanto tendremos que hacer una interface de usuario sencilla, intuitiva, consistente, y una buena distribución de la información adaptándonos a sus posibilidades.

El super usuario (administrador), tendrá acceso exclusivo a la parte privada del software, se le considera un experto en la materia por lo tanto no hay requisitos de usabilidad. Los administradores han de ser capaces de utilizar el sistema después de una explicación práctica de una hora.

4.2.4 Especificación de las clases de análisis

4.2.4.1 Identificación de las clases de entidades

La identificación de las clases de entidades consiste en identificar unas primera clases por medio de las cuales se puedan especificar los casos de uso en forma de iteraciones.

Caso de uso “Alta usuario ” clases: Usuarios, .
Caso de uso “Registrarse”,clases: UsuariosRegistrados, Usuarios
Caso de uso “consulta catálogo”, clases: Usuarios, UsuariosRegistrados *, SuperUsuarios*, producto, materia.
Caso de uso “Consulta usuario”, clases UsuariosRegistrados *
Caso de uso “Modifica usuario”, clases: UsuariosRegistrados *
Caso de uso “Baja usuario”, clases: UsuariosRegistrados *
Caso de uso “Añadir productos cesta”, clases: UsuariosAnónimo*, UsuariosRegistrados *, Productos *.
Caso de uso “Elimina productos cesta”, clases: Usuarios* UsuariosRegistrados *, Productos *.
Caso de uso “Consulta cesta”, clases Usuarios*, UsuariosRegistrados *, Productos *, Cesta*, Líneas*
Caso de uso ‘‘Confirmar compra”, clases UsuariosRegistrados *, Productos *, Cesta*, LíneaVenta, Pedido.
Caso de uso “Alta materia”, clases: SuperUsuarios *, Materias
Caso de uso “Baja materia”, clases: SuperUsuarios *, Materias*
Caso de uso “Modifica materia”, clases: SuperUsuarios *, Materias*
Caso de uso “Consulta materia”, clases: SuperUsuarios *, Materia*
Caso de uso “Alta producto catálogo”, clases: SuperUsuarios *, Productos
Caso de uso “Baja producto catálogo”, clases: SuperUsuarios *, Productos
Caso de uso “Modifica producto catálogo”, clases: SuperUsuarios *, Productos*
Caso de uso “Consulta producto catálogo”, clases: SuperUsuarios *, Productos*

Por lo tanto, la primera lista de clases de entidades es esta: Usuario anónimo, Usuario registrado, Super usuario, Productos, Materia, Pedido, Cesta, Línea venta.

4.2.4.2 Especificación de los atributos de las clases de entidades

Clase Usuario: nif(string),nombre(string), apellido1(string), apellido2(string), usuario(string) perfil(string), password(string)
Clase UsuarioRegistrado: nif, dirección(string), codi_postal(string), población(string), provincia(string),telefono(integer), email(string), cuenta_corriente(string), estado(string).
Clase Productos: idProducto(integer), idMateria(integer), autor(String), titulo(String), ISBN(String), idioma(string), precio(float), descuento(float), stock(integer), novedad(String).
Clase Materias: codMateria (string), nomMateria(string).
Clase LineaVentas: idLinea(integer), cantidad(integer), importeLinea(float), idProducto(integer), precio(float)
Clase Pedidos: idPedido(integer), importe(float), idUsuario(string),

4.2.4.3 Diagrama de clases de entidades

El diagrama de clases tiene como objetivo identificar los objetos que forman parte del sistema a implementar.

Este es el diagrama de clases donde podemos ver que las clases UsuariosRegistrados y SuperUsuario heredan de un clase padre llamada Usuario, esta es una herencia por especialización.
Entre Pedido y Productos se genera una líneaVenta , por lo tanto, se define como una clase asociativa.
Un Producto pertenece a una materia y una materia esta formada por ninguno o varios productos.

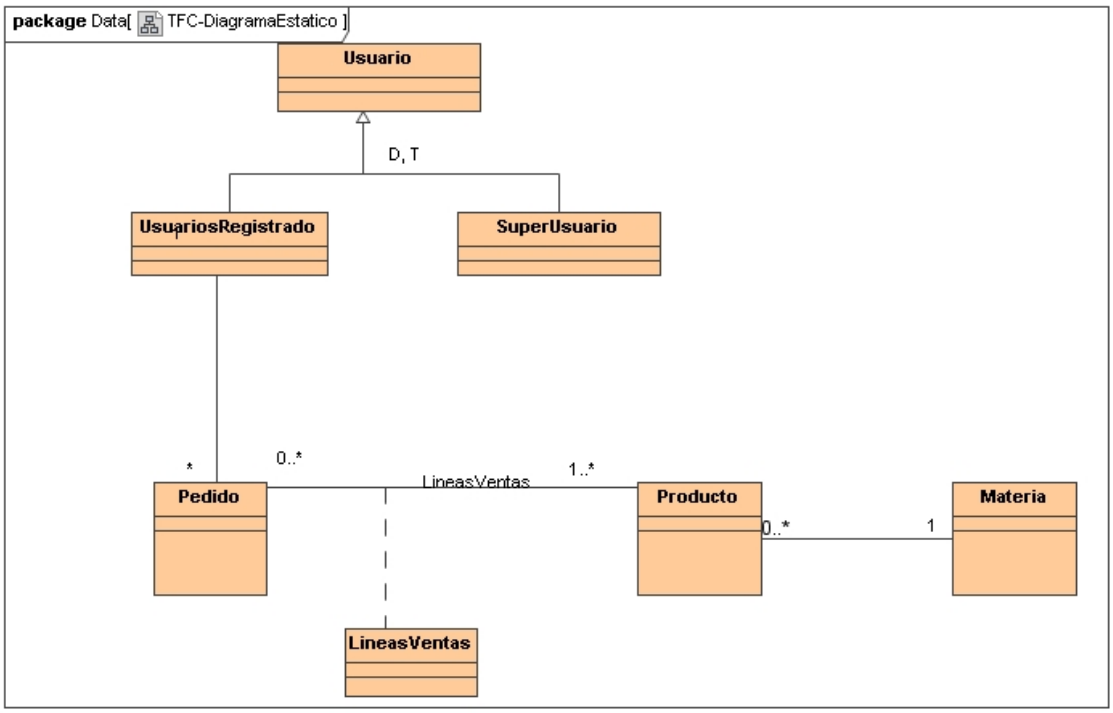


Figura 4. Diagrama de clases

4.2.4.4. Identificación de las clases frontera, las clases de control y de las operaciones

4.2.4.4.1 Diagramas de colaboración

A continuación se detallan los diagramas de colaboración uno por cada caso de uso.

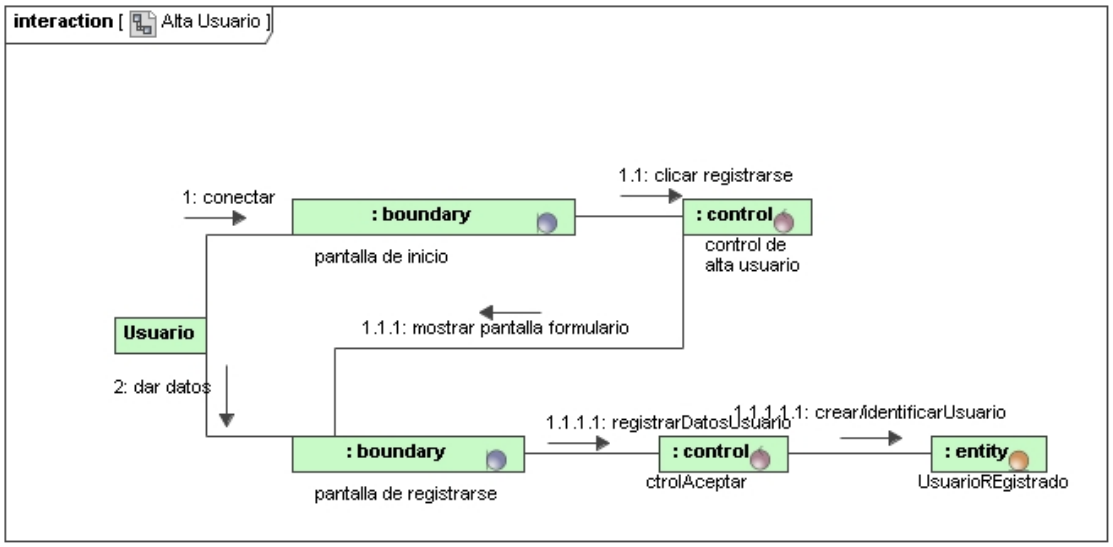


Figura 5. Diagrama de colaboración Alta usuario

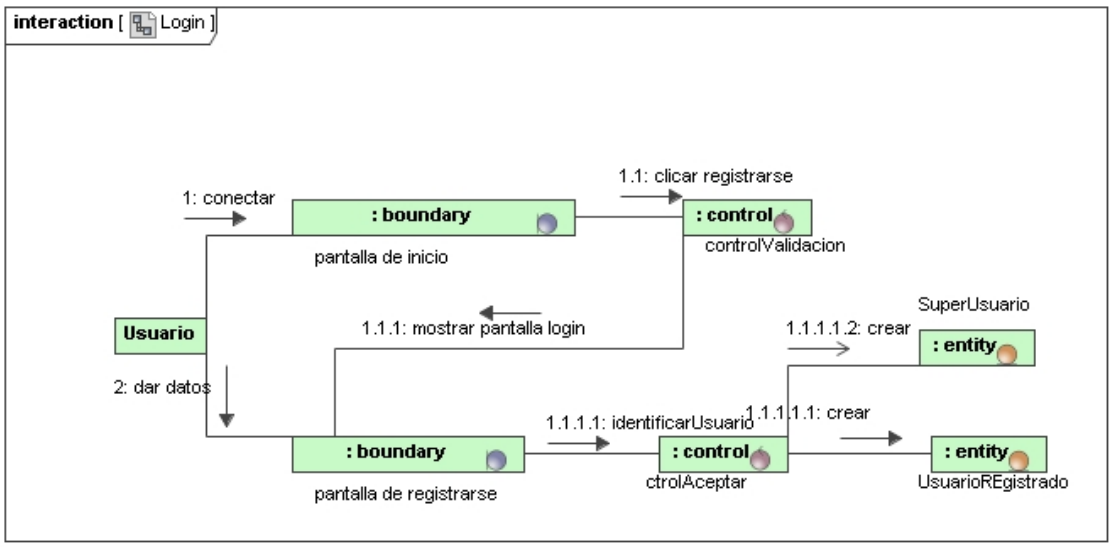


Figura 6. Diagrama de colaboración Login

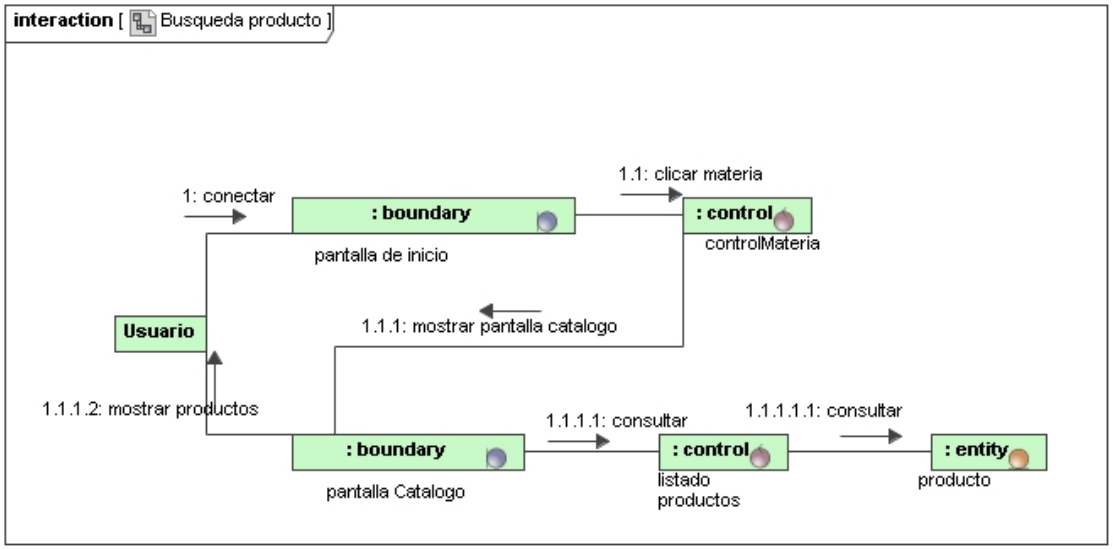


Figura 7. Diagrama de colaboración Búsqueda producto por materia

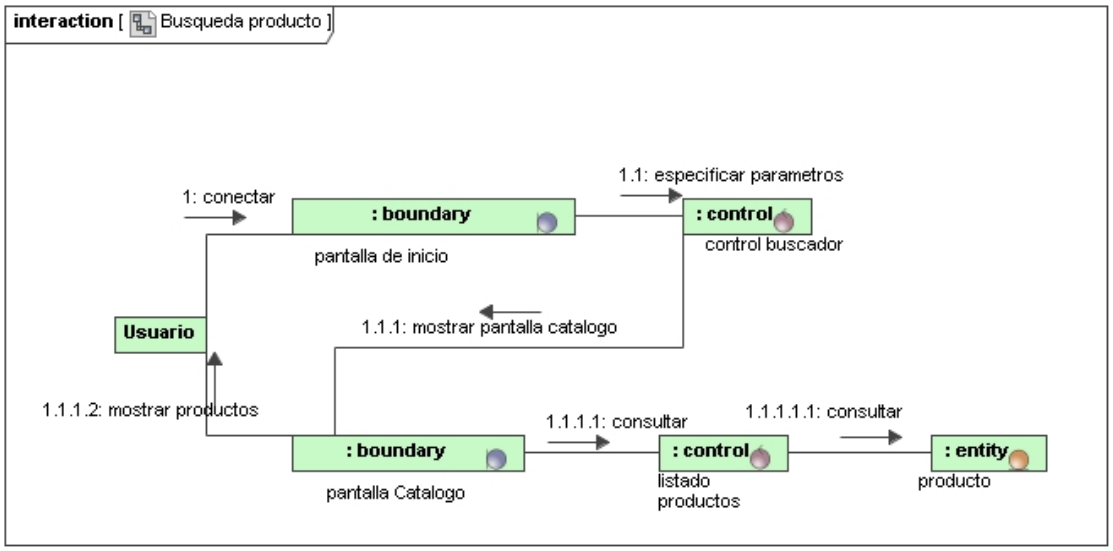


Figura 8. Diagrama de colaboración Búsqueda producto

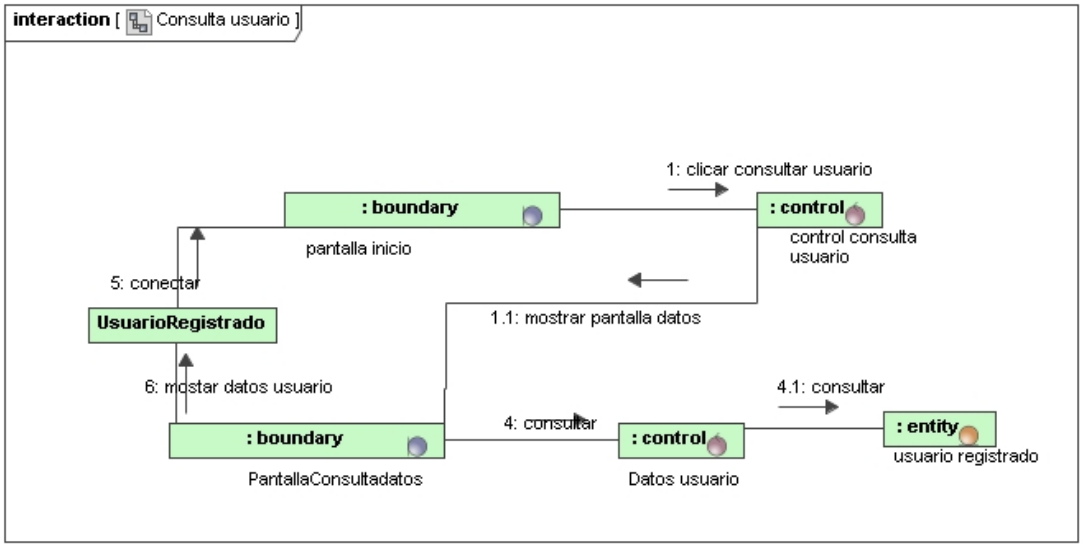


Figura 9. Diagrama de colaboración Consulta usuario

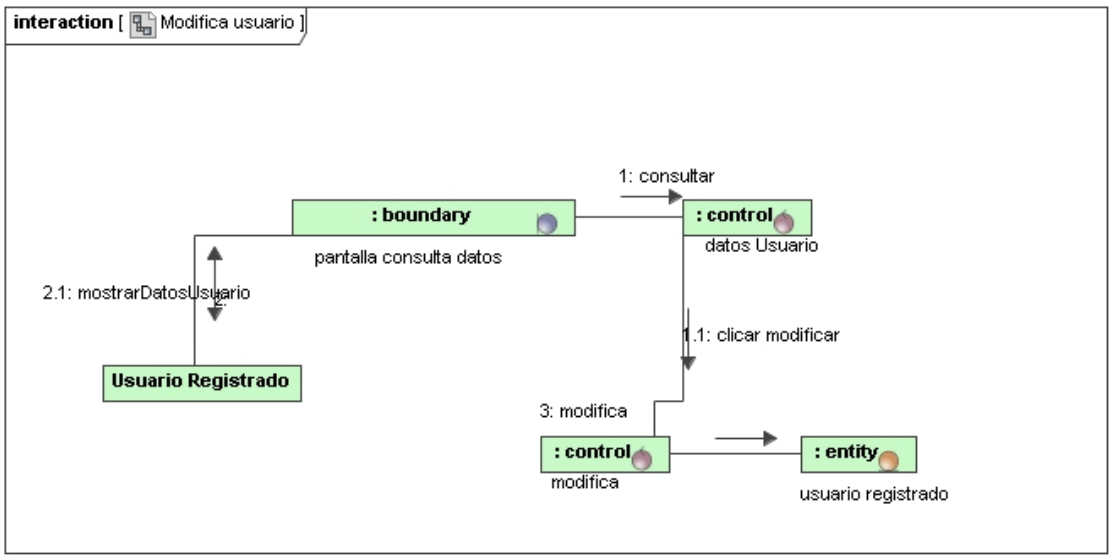


Figura 10. Diagrama de colaboración Modifica usuario

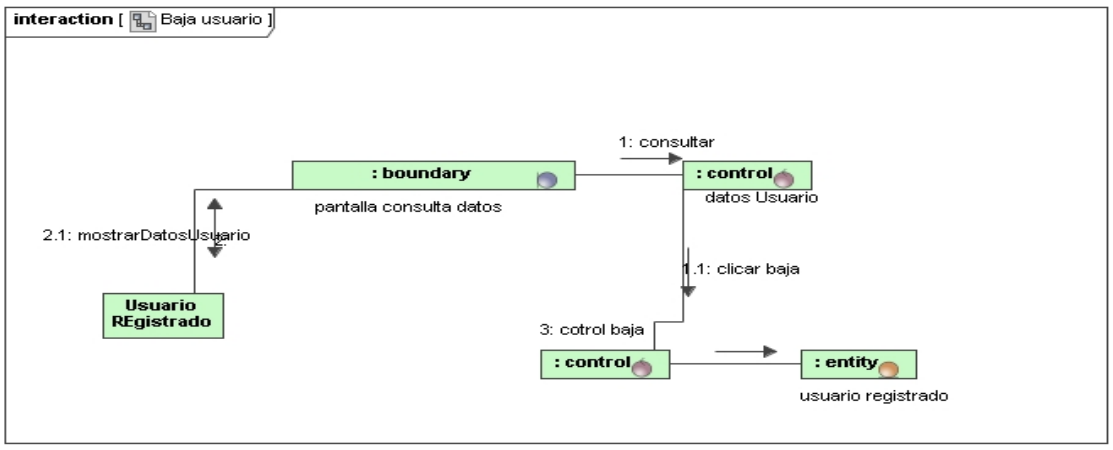


Figura 11. Diagrama de colaboración Baja usuario

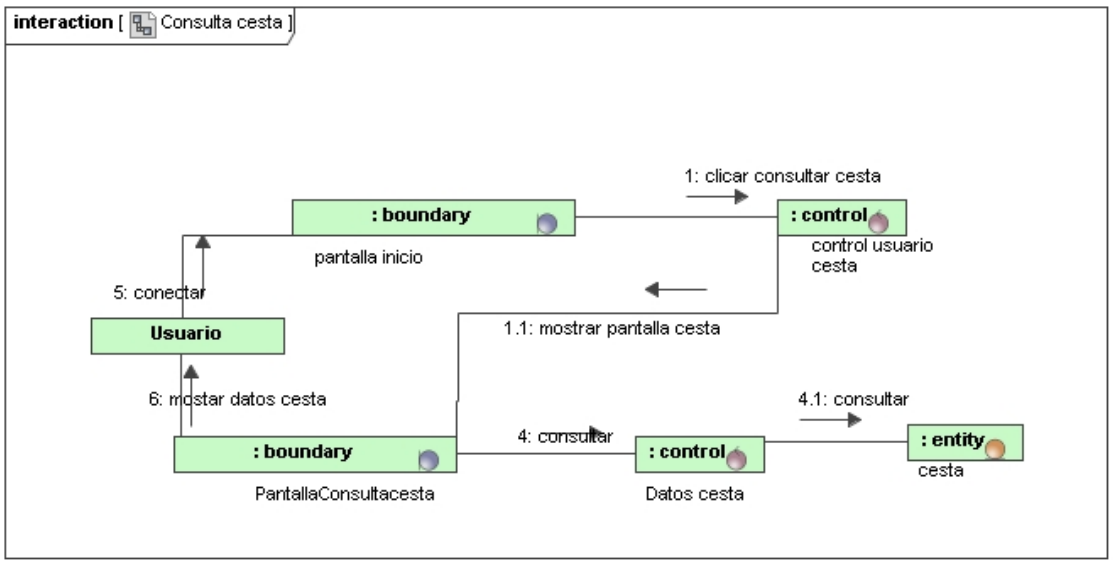


Figura 12. Diagrama de colaboración Consulta cesta

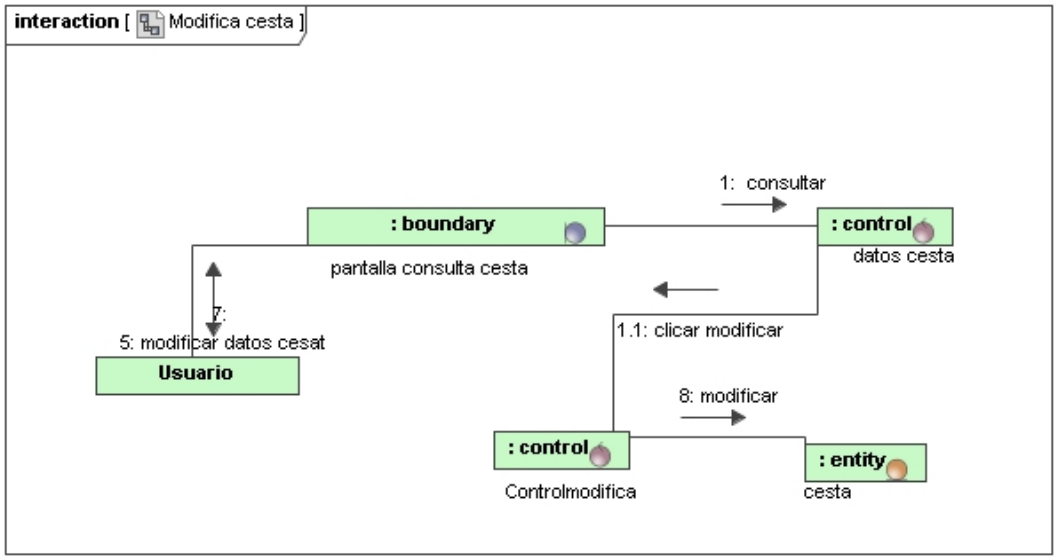


Figura 13. Diagrama de colaboración Modifica cesta

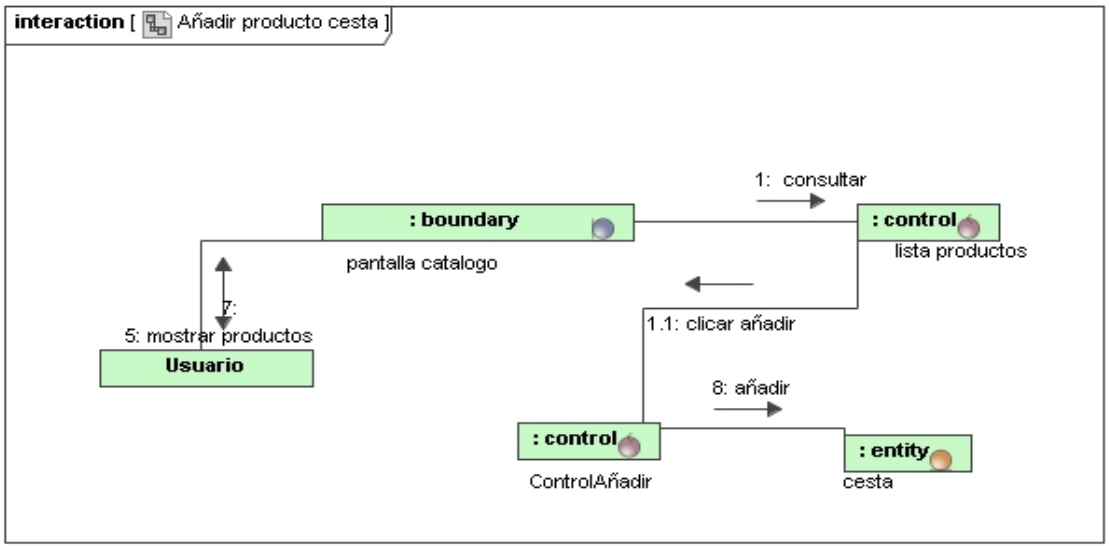


Figura 14. Diagrama de colaboración Añadir productos cesta

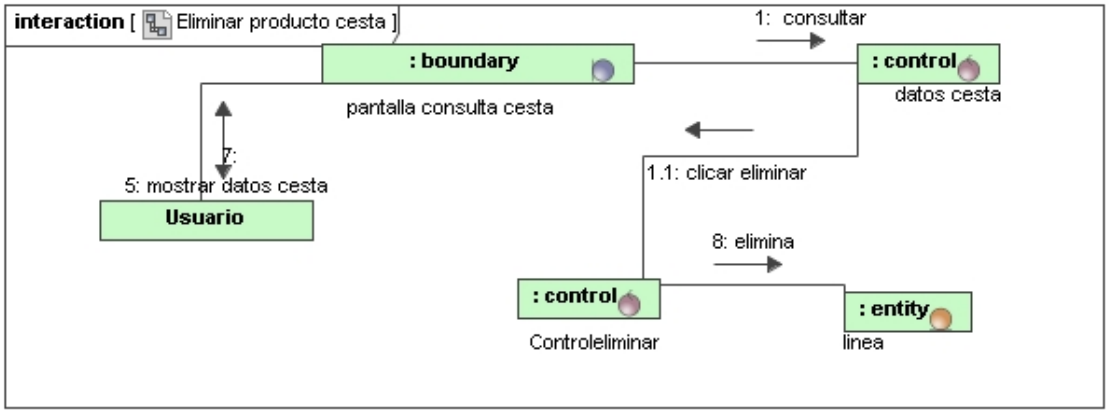


Figura 15. Diagrama de colaboración Elimina productos cesta

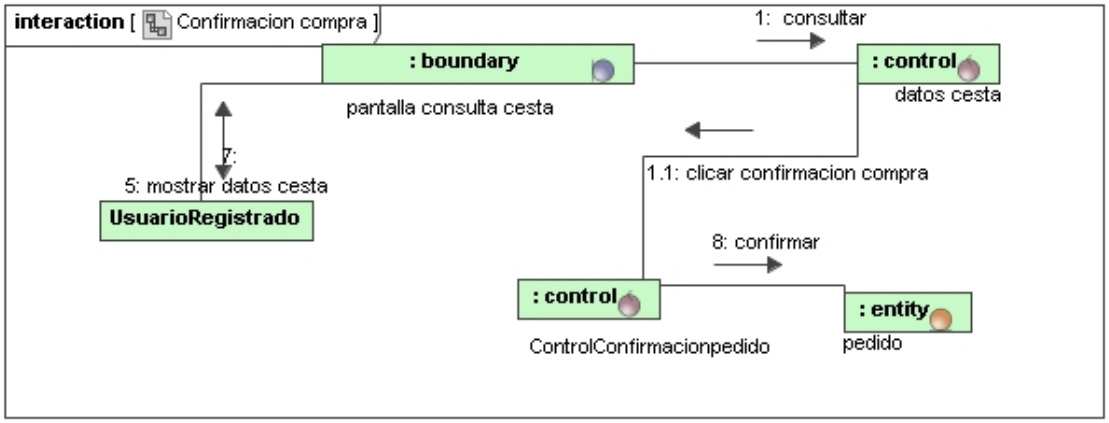


Figura 16. Diagrama de colaboración Confirmación compra

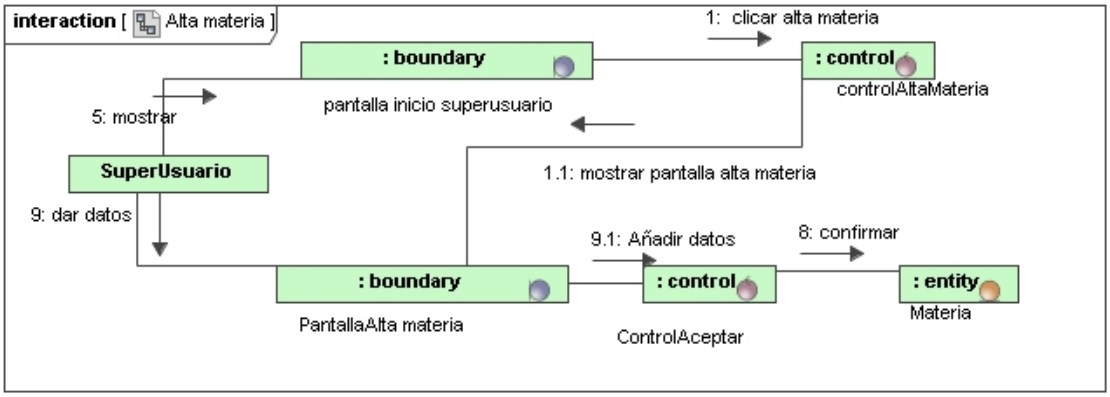


Figura 17. Diagrama de colaboración Alta materia

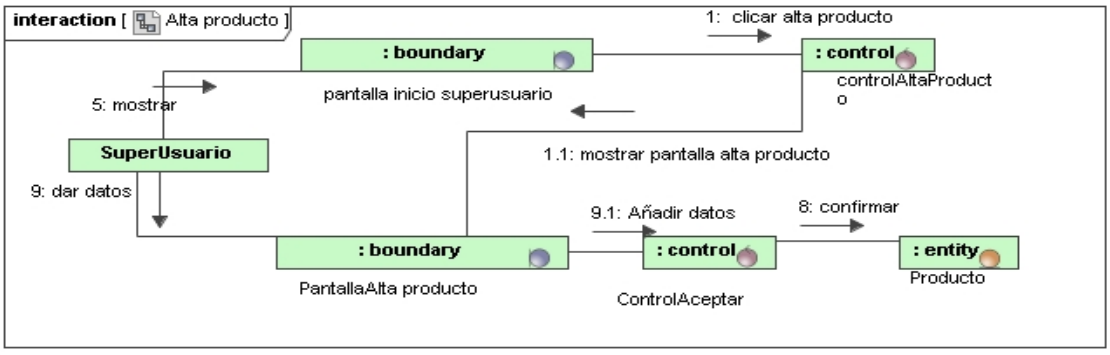


Figura 18. Diagrama de colaboración Alta producto

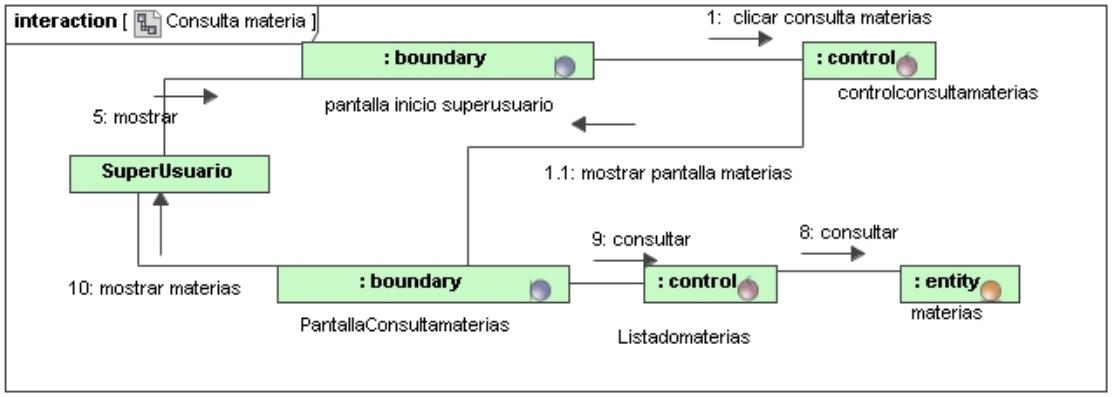


Figura 19. Diagrama de colaboración Consulta materia

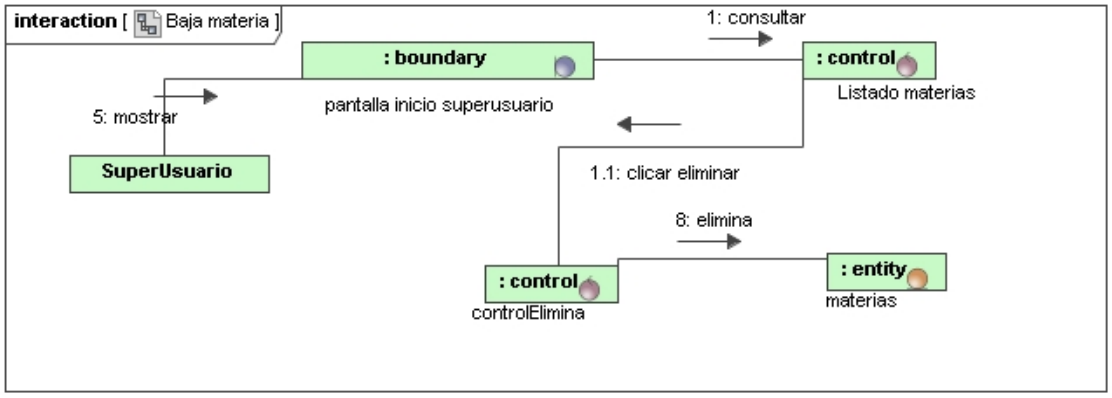


Figura 20. Diagrama de colaboración Baja materia

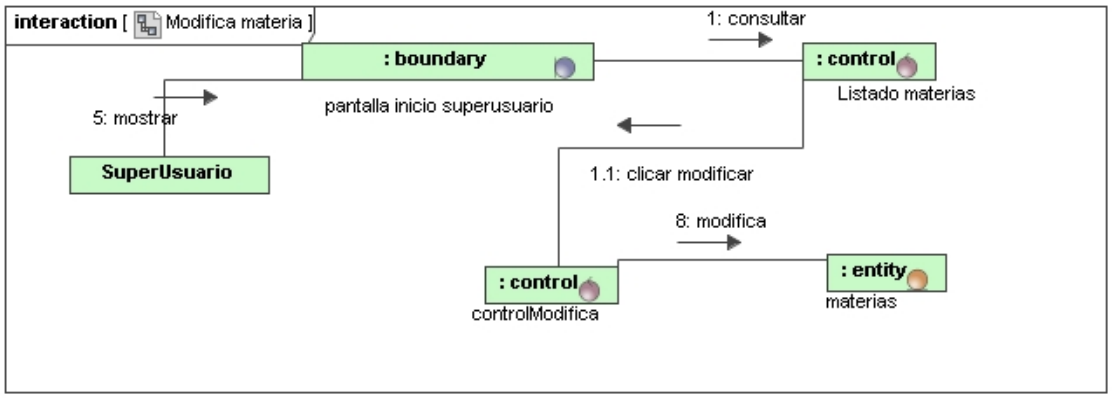


Figura 21. Diagrama de colaboración Modifica materia

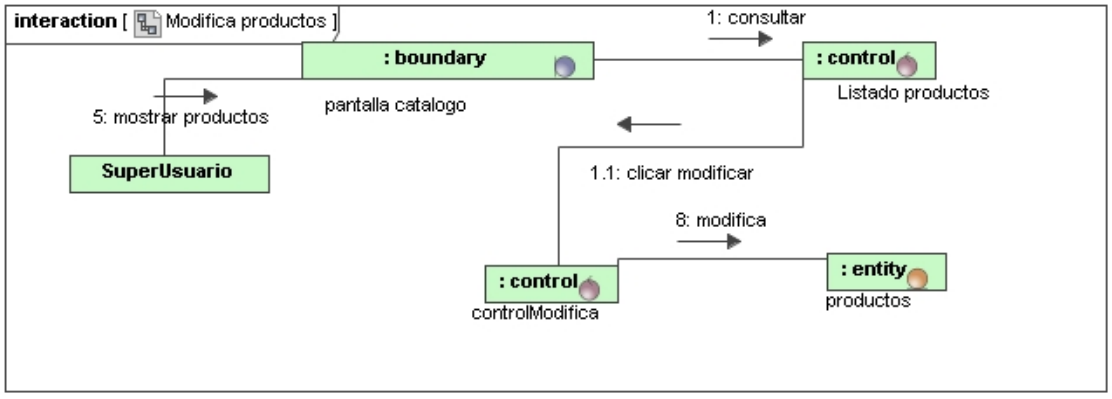


Figura 22. Diagrama de colaboración Modifica Productos

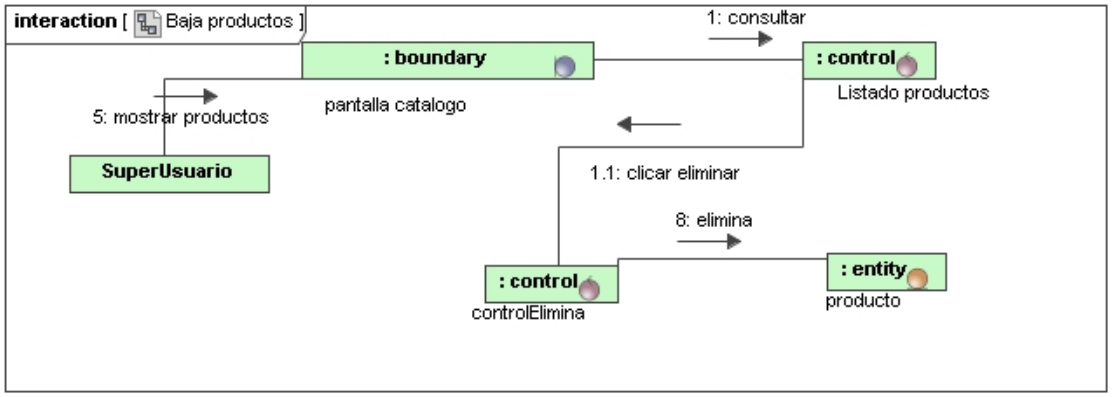


Figura 23. Diagrama de colaboración Baja Productos

4.2.4.4.2 Diagramas de secuencias

A continuación detallamos los diagramas de secuencias, únicamente se mostraran los diagramas más representativos.

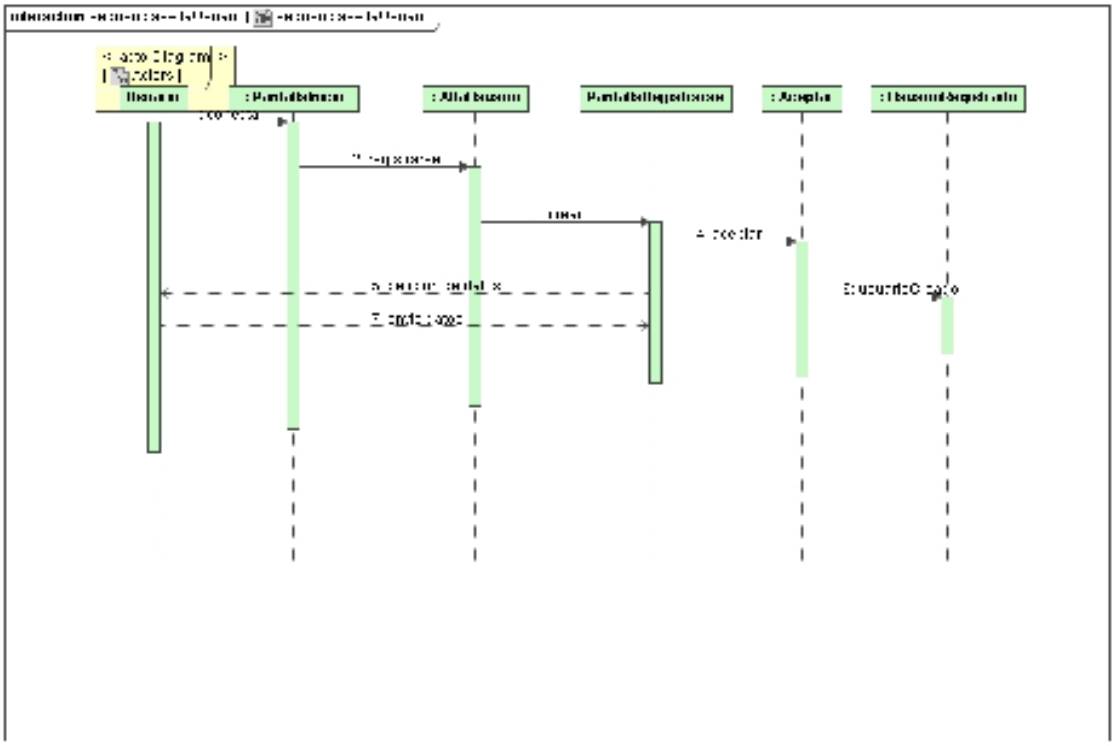


Figura 24. Diagrama de secuencias Alta usuarios

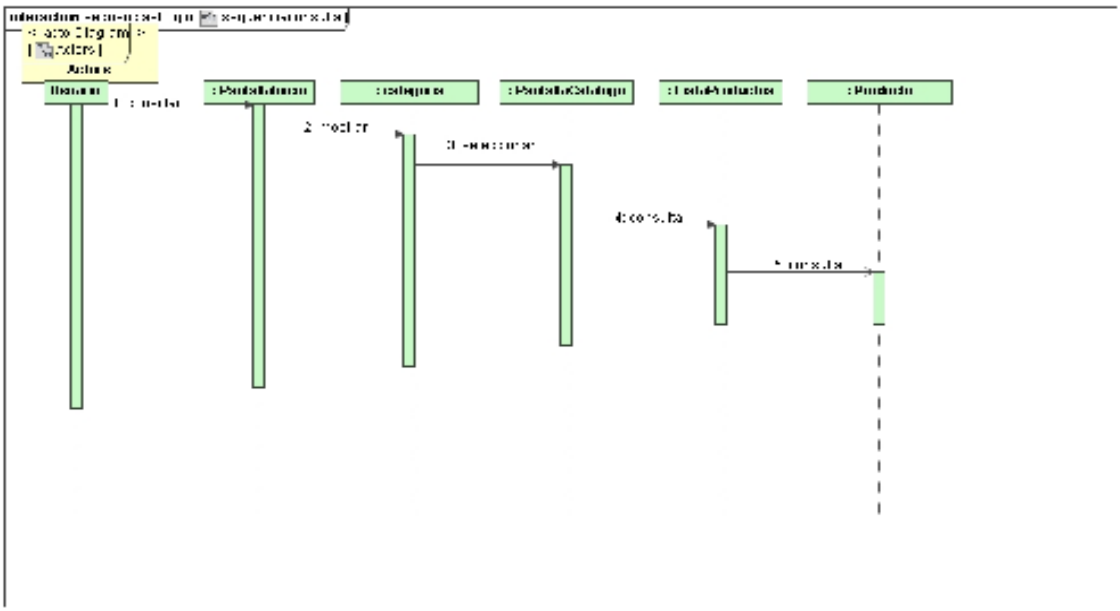


Figura 25. Diagrama de secuencias consulta

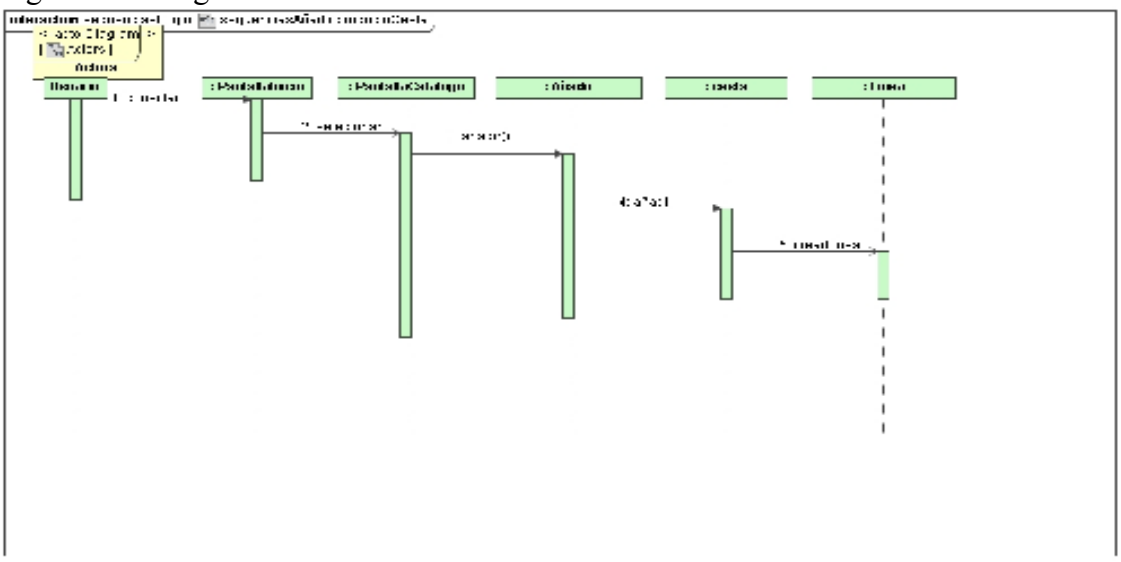


Figura 26. Diagrama de secuencias añadir productos cesta

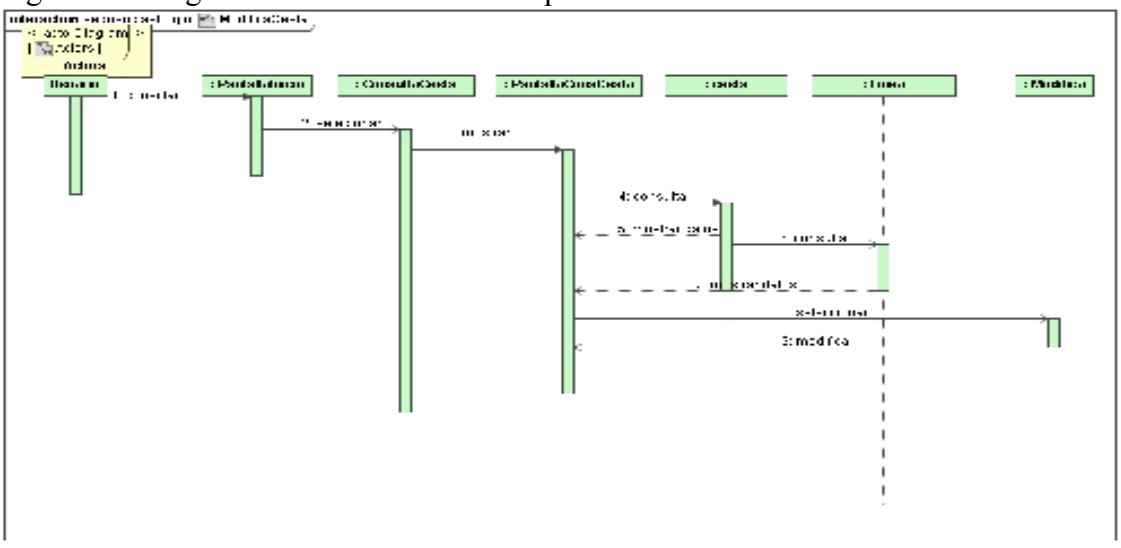


Figura 27. Diagrama de secuencias modificación cesta

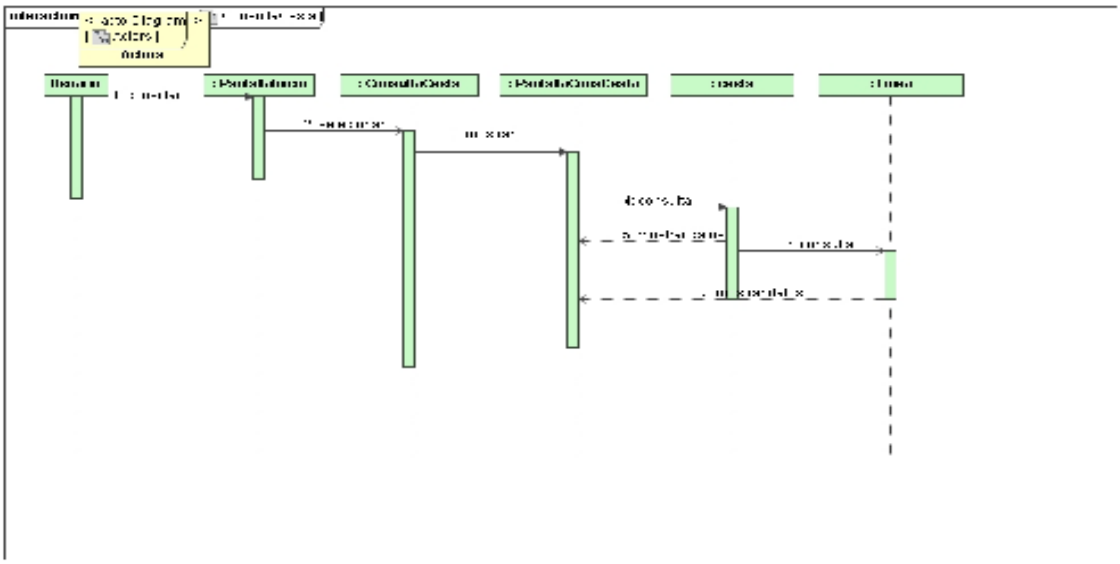


Figura 28. Diagrama de secuencias consulta cesta

4.3 Diseño

4.3.1 Diseño arquitectónico

En el diseño arquitectónico se deciden los aspectos que condicionaran de manera decisiva el resto del diseño. Se establecerán los subsistemas y módulos que formaran el sistema a diseñar, la forma de controlarlos y comunicarlos. Nos permite dividir la tarea del diseño en diseños más pequeños y asequibles.
Un subsistema es un sistema por si mismo, la operación la cual es independiente de los otros servicios aunque interaccionen. Un modulo es una parte del subsistema que provee un conjunto de funcionalidades a los otros módulos, pero que no puede funcionar independientemente.

4.3.1.1 Identificación de subsistemas

Se han detectado 4 subsistemas:
Subsistema de usuario: permite gestionar los usuarios del sistema (mantenimiento de usuarios). Todas las acciones serán utilizada por un usuario registrado.
Subsistema tienda: Tiene todas las funcionalidades propias de una tienda virtual. Será utilizado por el usuario anónimo y registrado.
Subsistema consultas: Permite generar listados de búsqueda de productos por pantalla mostrando información sobre los productos.
Subsistema catálogo: Permite al administrador gestionar los artículos de la librería, creando las materias área de clasificación de libros y mantenimiento de libros.

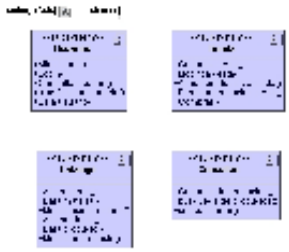


Figura 28. Subsistemas

4.3.1.2 Introducción a la plataforma J2EE

J2EE es un conjunto de especificaciones de APIs Java para la construcción de aplicaciones empresariales

- La mayor parte de las abstracciones de las APIs corresponden a interfaces y clases abstractas.
- Existen múltiples implementaciones de distintos fabricantes, incluso algunas OpenSource.
- Una aplicación construida con J2EE no depende de una implementación particular.

La plataforma J2EE cuentan con las siguientes características:

- Escalabilidad
- Portabilidad
- Seguridad

¿ Cómo se debe diseñar una aplicación empresarial para que sea mantenible y contenga partes reusables ?

- Debería estar diseñada siguiendo la arquitectura que fijan los patrones arquitectónicos Model-View-Controller(MVC)
- Un patrón arquitectónico es un patrón de alto nivel que fija la arquitectura global de una aplicación
- Posteriormente, el diseño hará uso de patrones de diseño para resolver problemas específicos

En el patrón arquitectónico MVC existe una separación clara entre el modelo (lógica de negocio) y la vista (interfaz gráfica), gracias a un controlador que los mantiene independientes unos de otros (desacoplados). Ventajas:

- El modelo es reusable con distintas vistas (ej.: una vista web y una con interfaz de ventanas)
- División clara de trabajo entre los miembros de un equipo, que estará formado por personas con distintos niveles de especialización

4.3.1.3 MVC Y FRAMEWORKS PARA J2EE

El patrón Modelo-Vista-Controlador es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información, el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema.

La mayoría, por no decir todos, de los Frameworks para Web implementan este patrón. Una aplicación de este patrón en entornos Java para programación Web es lo que se conoce con el nombre de arquitectura model 2. Esta arquitectura consiste, a grandes rasgos, en la utilización de servlets para procesar las peticiones (controladores) y páginas JSP para mostrar la interfaz de usuario (vistas), implementando la parte del modelo mediante JavaBeans o POJOs. (En la arquitectura MVC de tipo 2 las diferentes páginas que el usuario ve lanzan acciones utilizando un único controlador, que despacha las diferentes peticiones a un conjunto de acciones previamente registradas en el controlador. Un único servlet es llamado desde el cliente, quedando visible un único punto de entrada al controlador. Esta arquitectura contrasta con el tipo 1, en la que la lógica de control de la aplicación (acciones que hay que llamar, vistas que se deben generar y control de navegación entre páginas) va integrada en la capa de presentación. Esta arquitectura es la utilizada en ASP.NET y en JSF (Java Server Faces). En este caso el usuario genera eventos que son tratados en el controlador, asociando acciones a los diferentes eventos.)

En el desarrollo de software, un Framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, un Framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Provee de una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.
En general, con el término Framework, nos estamos refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un Framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un Framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Un Framework Web, por tanto, podemos definirlo como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.

Para facilitar el desarrollo de las aplicaciones J2EE se han ideado varios Frameworks de los cuales destacamos:

Framework	Función
Java Server Faces	Creación Interfaces
Hibernate	Facilitar conectividad con BBDD
Spring	Facilitar configuración beans app
Struts	Facilitar control de eventos

- ***Java Server Faces:*** orientado a la creación de interfaces de usuario.
- ***Hibernate:*** que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos.
- ***Spring:*** que tiene como objetivo facilitar la configuración de los java beans dentro de una aplicación. Su meta es conseguir separar los accesos a datos y los aspectos relacionados con las transacciones, para permitir objetos de la capa de negocio reutilizables que no dependan de ninguna estrategia de acceso a datos o transacciones.
- ***Struts:*** orientado a la parte de control de eventos. Este será empleado para la implementación de nuestra aplicación por lo cual será explicado detalladamente más adelante.

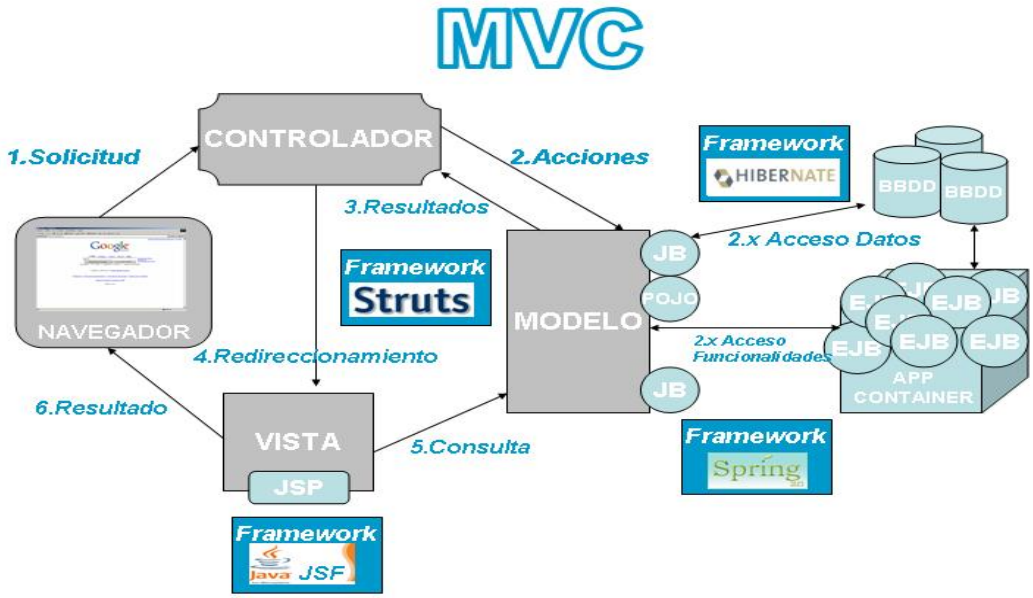


Figura 29. Framework

Estos Frameworks pueden ser utilizados entre sí ya que en la mayoría de los casos están destinados a dar solución a determinados problemas, pero no a todos. Con lo cual es posible una combinación de los 4 dependiendo de las necesidades de la aplicación. En nuestro proyecto solo utilizaremos el Framework Struts ya que la utilización de más Frameworks incrementaría la carga de trabajo más allá de lo estipulado. Para un futuro proyecto se podría

mejorar la aplicación , aplicando las ayudas que pudiera proporcionar otro Framework.

4.3.1.4 Patrón arquitectónico Modelo MVC para aplicaciones web (Struts)

El patrón Modelo-Vista-Controlador es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Este patrón organiza la aplicación en tres modelos separados, tal como se explica en el punto 4.3.1.3.

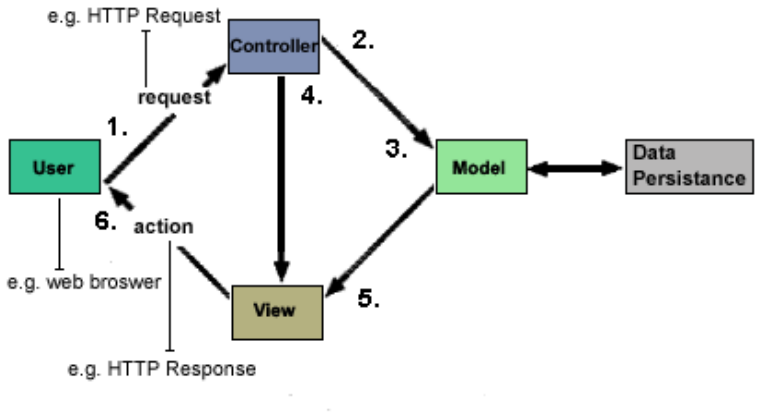


Figura 30. Modelo MVC

4.3.1.5 El Framework Struts

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts permite reducir el tiempo de desarrollo. Su carácter de "software libre" y su compatibilidad con todas las plataformas en que Java Entreprise esté disponible, lo convierte en una herramienta altamente disponible.

4.3.1.5.1 Funcionamiento de Struts

Struts se basa en el patrón del Modelo Vista Controlador (MVC) el cual se utiliza ampliamente y es considerado de gran solidez. De acuerdo con este modelo, el procesamiento se separa en tres secciones diferenciadas, llamadas el modelo, las vistas y el controlador.

Cuando se programan aplicaciones Web con el patrón MVC, siempre surge la duda de usar un solo controlador o usar varios controladores, pues si consideramos mejor usar un solo controlador para tener toda nuestra lógica en un mismo lugar, nos encontramos con un grave problema, ya que nuestro controlador se convierte en lo que se conoce como "fat controller", es decir un controlador saturado de peticiones, Struts surge como la solución a este problema ya que implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

Componentes de modelo
Corresponden a la lógica del negocio con el cual se comunica la aplicación Web. Usualmente el modelo comprende accesos a Bases de Datos o sistemas que funcionan independientemente de la aplicación Web.

Componentes de control
Los componentes de control son los encargados de coordinar las actividades de la aplicación, que van desde la recepción de datos del usuario, las verificaciones de forma y la selección de un componente del modelo a ser llamado. Por su parte los componentes del modelo envían al control sus eventuales resultados o errores de manera de poder continuar con otros pasos de la aplicación.

Esta separación simplifica enormemente la escritura tanto de vistas como de componentes del modelo: Las páginas JSP no tienen que incluir manejo de errores, mientras que los elementos del control simplemente deciden sobre el paso siguiente a seguir.

4.3.1.5.2 Características de Struts

Entre las características de Struts se pueden mencionar:

- Configuración del control centralizada.
- Interrelaciones entre Acciones y página u otras acciones se especifican por tablas XML en lugar de codificarlas en los programas o páginas.
- Componentes de aplicación, que son el mecanismo para compartir información bidireccionalmente entre el usuario de la aplicación y las acciones del modelo.

- Librerías de entidades para facilitar la mayoría de las operaciones que generalmente realizan las páginas JSP.
- Struts contiene herramientas para validación de campos de plantillas bajo varios esquemas que van desde validaciones locales en la página (en JavaScript) hasta las validaciones de fondo hechas a nivel de las acciones.

Struts permite que el desarrollador se concentre en el diseño de aplicaciones complejas como una serie simple de componentes del Modelo y de la vista intercomunicados por un control centralizado. Diseñando de esta manera se debe obtener una aplicación más consistente y más fácil de mantener.

4.3.1.6 LOG4J

Log4j es un framework escrito para Java, que nos abstrae mediante una capa este servicio para que podamos emplear su funcionalidad de forma transparente; Loggers, Levels, Appenders y Layouts.
Se emplea Log4j por que de esta manera podemos tener es un servicio de diario de actividad. Nos va a permitir a los desarrolladores un seguimiento detallado del funcionamiento de una aplicación, con la que poder pulir todos los detalles de funcionamiento de la aplicación, así como, localizar y aislar los errores que se hayan cometido.

4.3.1.7 Tiles

El "framework Tiles component view" es conocido como tiles. Este framework usa un archivo de configuración para estos tiles. Este framework no solo te permite rehusar tiles, sino que también que permite organizar layouts además permite la integración con struts. Se utiliza este framework para facilitar el mantenimiento de las paginas jsp.

4.3.1.8 Filtros

Los filtros son componentes que pueden utilizarse para analizar y/o transformar tanto los datos solicitados como los enviados en una petición web.
Pueden trabajar en conjunto con páginas jsp o servlets.
Se emplean los filtros para un control de acceso a la aplicación definiendo una referencia por patrones /*.do.

4.3.1.9 Diagrama de la arquitectura

Esta es la arquitectura que he planteado para desarrollar la aplicación. Estos son los paquetes que forman la arquitectura:
Actions, Comandos/Facade, Forms, DAO, Persistencia y Los paquetes de generación de la vista (páginas Jsp).

He empleado un modelo vista controlador (MVC implementado en el framework de Struts), su definición obliga a desarrollar la funcionalidad de Actions, Forms y JSP.
Los Actions se comunican con los Forms o formularios de donde recogen información suministrada por el usuario. Estos Action llaman a la lógica de negocio que esta encapsulada en los Comandos.
Si es preciso acceder a base de datos los Comandos realizan una llamada sobre un DAO (Data Access Object), cuya única misión es realizar las operaciones con Base de Datos que le pide la lógica de negocio.

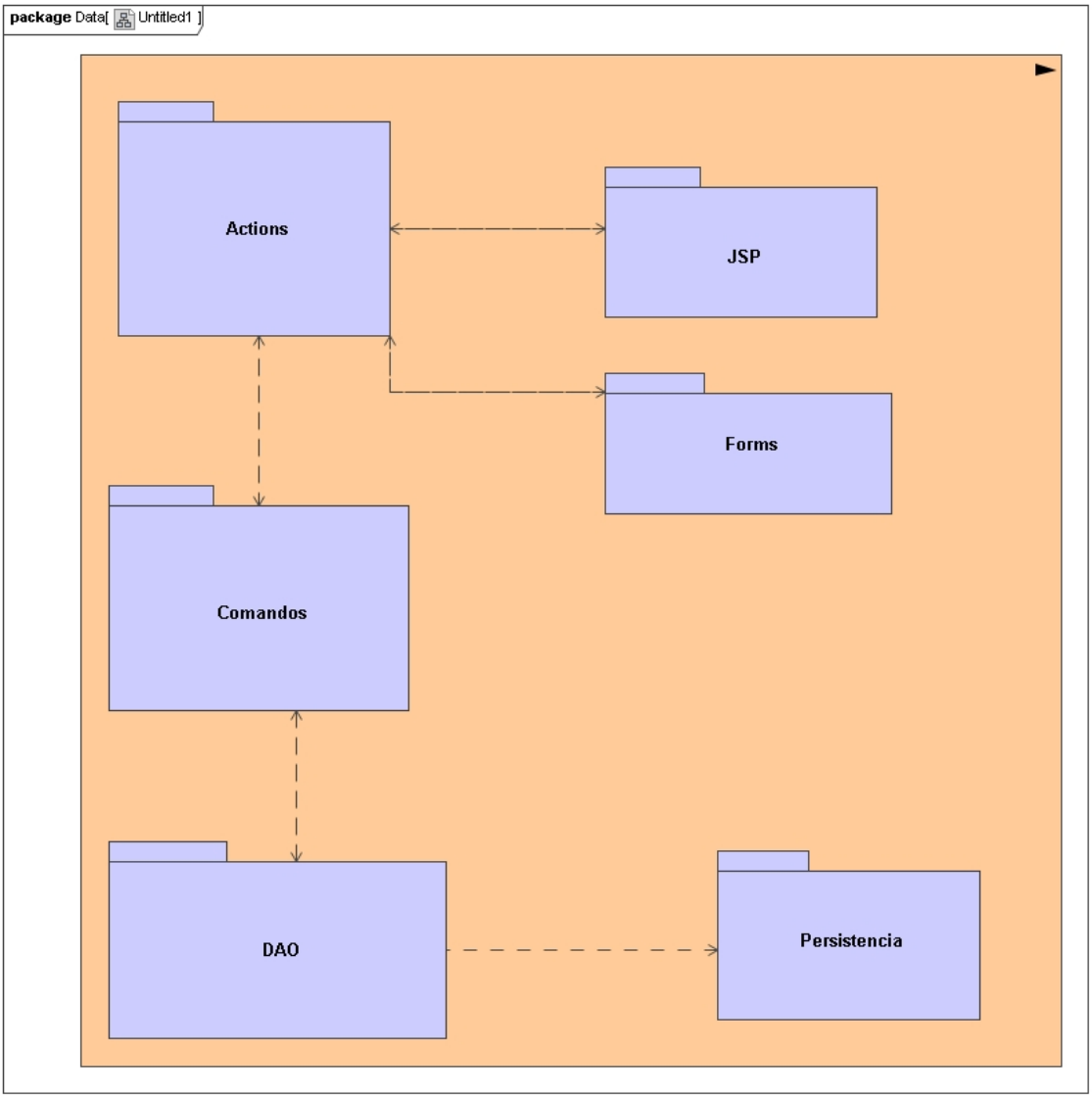


Figura 31. diagrama de arquitectura

4.3.2 Diseño de la persistencia

Nos interesa tan solo hacer persistentes las Clases de entidad y para conseguir que nuestras clases sean persistentes, nuestra ideas es mapear nuestros objetos al mundo relacional, tendremos que trabajar con una base de datos para conseguirlo, lo realizaremos a través de JDBC y usaremos como gestor de base de datos MYSQL, para ello instalaremos el driver JDBC correspondiente.
Aprovechando que el Framework Struts permite definir un pool de conexiones, este es el que se utilizara en vez del que puede ofrecer el propio Tomcat para facilitar la instalación de la aplicación (tener un pool de conexiones configurado a nivel tomcat, no solamente requiere parametrización al web.xml, si no que además requiere una parametrización adicional en el fichero de configuración del Tomcat, server.xml, dificultando a si la distribución del producto.

Podríamos haber elegidos otras soluciones al estilo de Hibernate, pero como tenemos un modelo de clases no excesivamente complejo hemos optado por un mapeo directo entre ambos mundos. Otros de los motivos es el tiempo de aprendizaje de dicha tecnología y los problemas que conlleva, ya que el tiempo de proyecto es muy ajustado. Este punto se ha pensado para una mejora del proyecto.

Fundamentalmente nos interesara que sean persistentes las clase de entidad que hemos definido para ello trasformamos el diagrama de clases en el siguiente diagrama Entidad/Relación.

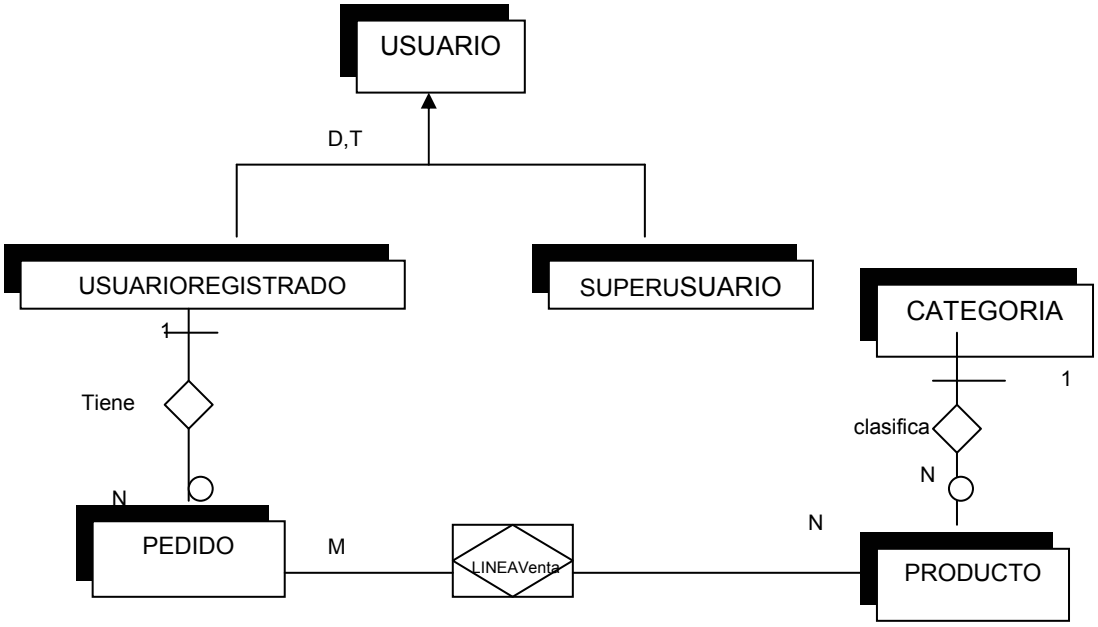


Figura 32. diagrama E-R

Las tablas obtenidas son:

Tabla Usuario		
nif	String	Clave primaria
nombre	String	
Apellido1	String	
Apellido2	strnig	
Usuario	string	
password	string	
perfil	integer	

Tabla Usuarios_registrados entidad subclase de usuario		
nif	String	Clave primaria, clave foránea a Usuario
Direccion	String	
Cp	String	
Poblacion	String	
Provincia	String	
Telefono	String	
Email	String	
Cuenta	String	
Estado	string	

Tabla super_usuario entidad subclase de usuario		
nif	String	Clave primaria, clave foránea a Usuario
Nivel_privilegio	Int	

Tabla Productos		
Id_producto	int	Clave primaria
autor	string	
titulo	string	
editorial	String	
isbn	String	
idioma	String	
descuento	float	
stock	int	
precio	Int	
Id_materia	Int	Referencia a materia
novedades	boolean	

Tabla materia		
Id_materia	Int	Clave primaria
nombre	string	

Tabla pedidos		
Id_pedido	int	Clave primaria
importe	float	
Estado	string	Por defecto pendiente
nif	String	Clave foranea a usuarioregistrado

Tabla linea de venta		
Id_linea	int	Clave primaria
Cantidad	int	
Id_producto	int	Clave foranea a producto
Id_pedido	int	Clave foránea a pedido
Nom_producto	String	
Precio	float	

4.3.3 Diseño de la interfaz del usuario

La interfaz queda identificada de la siguiente forma:

- **En la claridad de objetivos**

Los objetivos quedan definidos en tres bloques: el frontal superior donde aparece el menú, el bloque vertical derecho donde aparecen los temas disponibles a seleccionar por el usuario y la zona central donde aparecerán los contenidos que se hayan solicitado desde el menú o bloque derecho.

- **En su visibilidad**

La interfaz se identifica claramente por sus tres bloques indicados anteriormente, bloque de temas, bloque de menús o acciones y bloque de contenidos.

- **En su adecuación a los usuarios**

Los enlaces son textuales, los colores y tamaños de las letras son uniformes y comunes en todo el conjunto de la interfaz.

- **En la consistencia y estándares**

Los menús situados en la parte superior e izquierda están en todas las páginas para facilitar la navegación por la web.

- **En su flexibilidad y eficacia de uso**

El usuario puede acceder a todas las acciones y contenidos de manera sencilla y sin repetir pasos.

- **En el diseño minimalista**

No existe ninguna página que cargue imágenes u otros contenidos que dificulten el acceso rápido a la aplicación.

Otra de las características de la interfaz es la facilidad del cambio de formatos y

posición de la mayoría de los elementos y componentes de la aplicación, gracias al uso de un fichero de estilos CSS. Un simple cambio en este fichero se hace efectivo en toda la aplicación asegurando de esa manera una aplicación homogénea.

- **Utilización de la tecnología Tiles**

Esta tecnología nos facilita la modificación de menús, toda la aplicación utiliza la misma página de menús, es decir esta centralizado, con un simple cambio en el fichero correspondiente se hace efectivo en toda la aplicación asegurando de esa manera una aplicación homogénea.

Pantalla de inicio

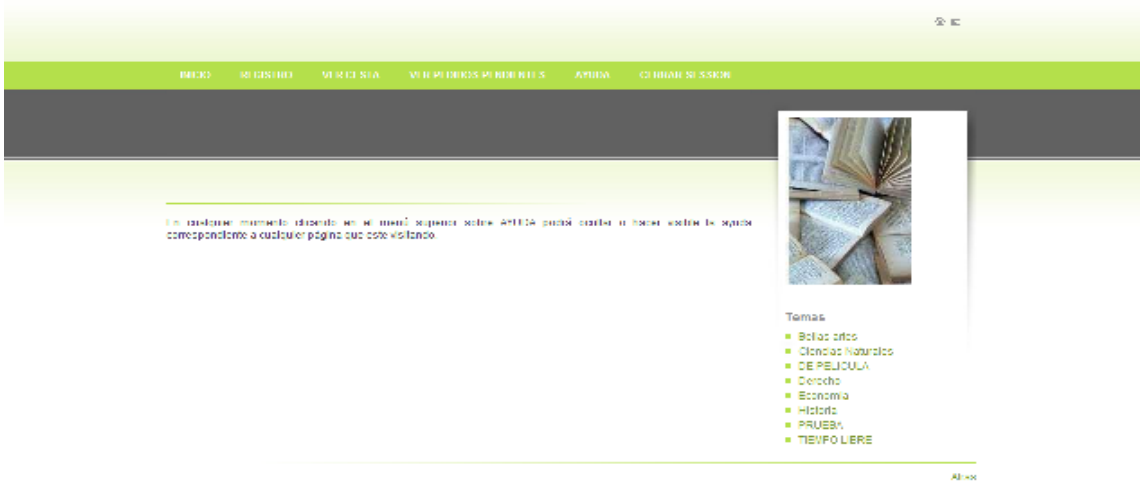


Figura 33. Pantalla inicio

Pantalla de acceso registrarse/modificación datos usuario / alta usuario

Figura 34. Pantalla registro

Pantalla alta usuario / Baja usuario

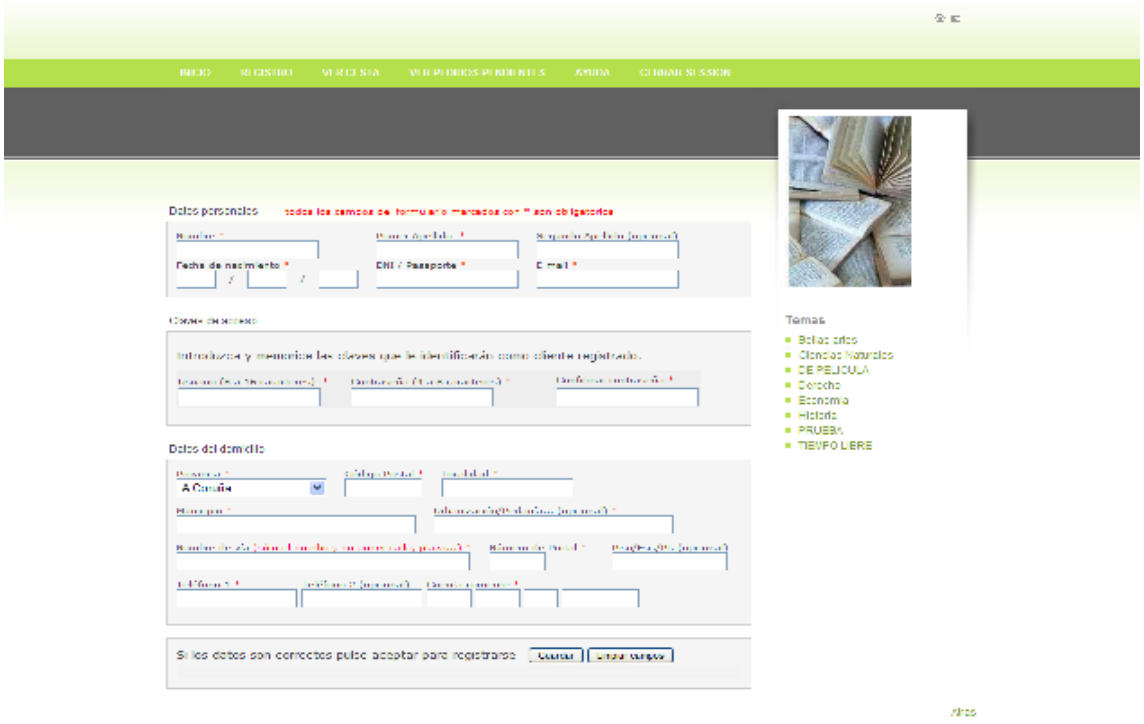


Figura 35. Pantalla alta/ bja usuario

pantalla de login

Figura 36. Pantalla login

Pantalla de búsqueda libros

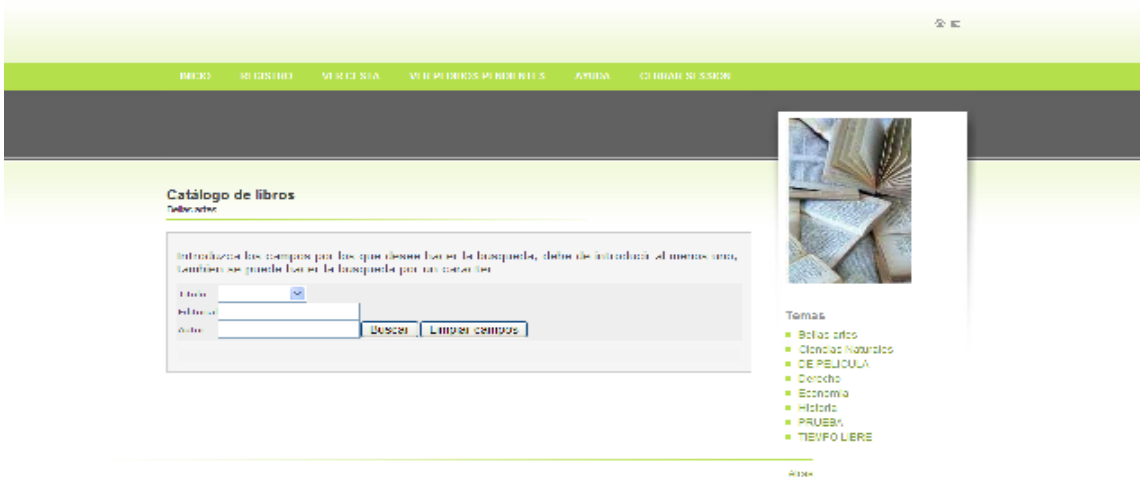


Figura 37. Pantalla busqueda libros

Pantalla de listado

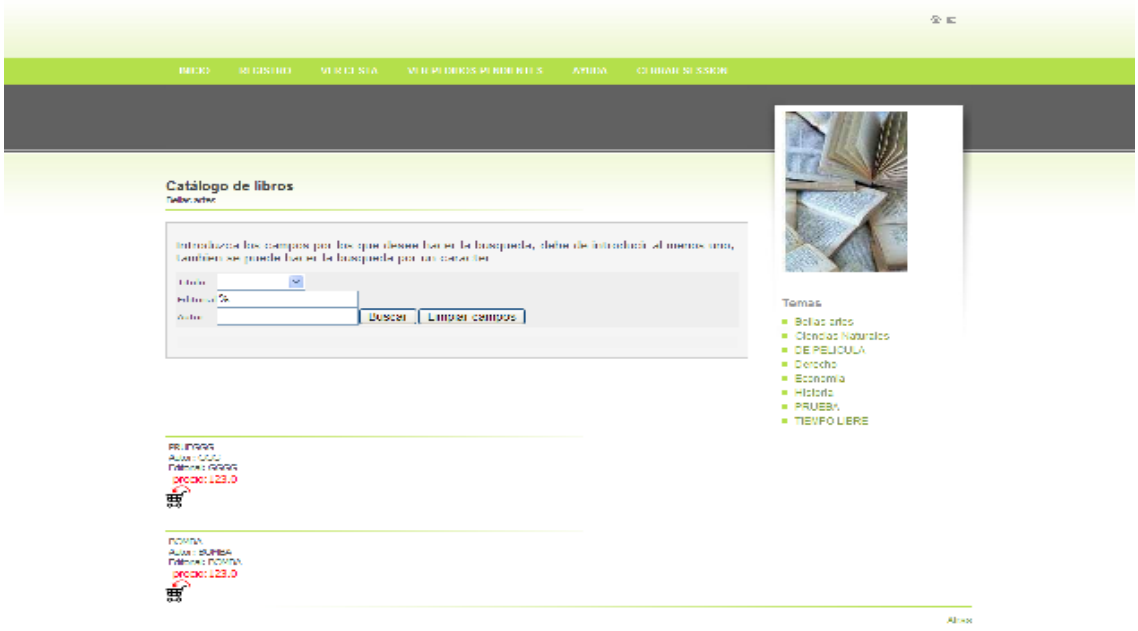


Figura 38. Pantalla listados

Pantalla de añadir producto a cesta

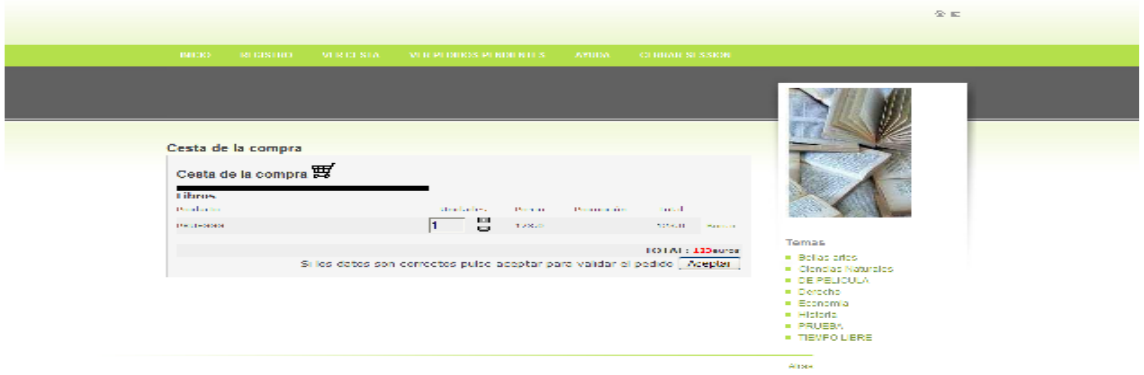


Figura 39. Pantalla añadir productos cesta

Pantalla de listado pedidos pendientes

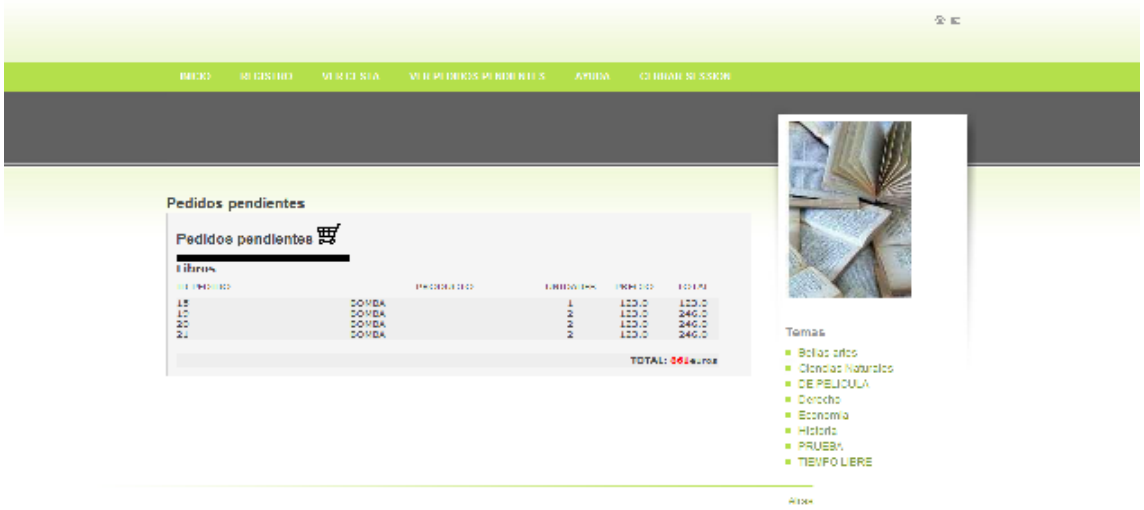


Figura 40. Pantalla listado pedidos pendientes

Pantalla de administrador

Figura 41. Pantalla Administrador

pantalla alta / modificaciones / baja libros

Figura 42. Pantalla alta / baja / modificación libros

Pantalla alta materia

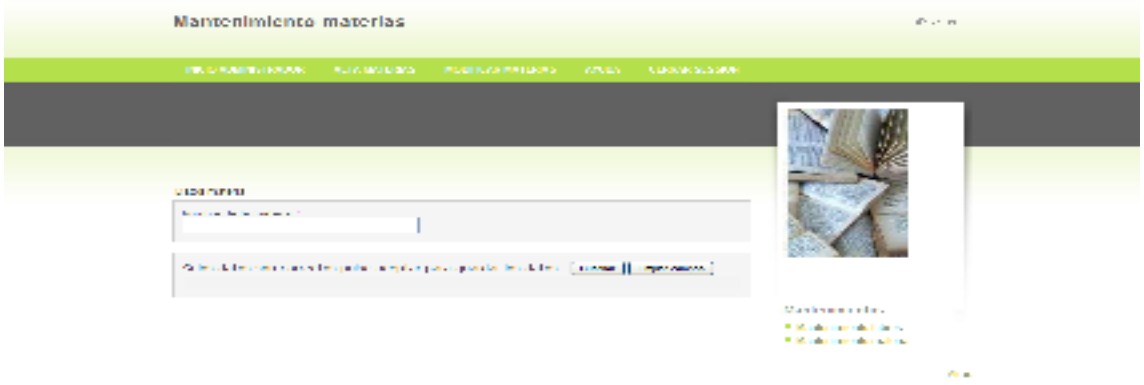


Figura 43. Pantalla alta materia

Pantalla baja materia

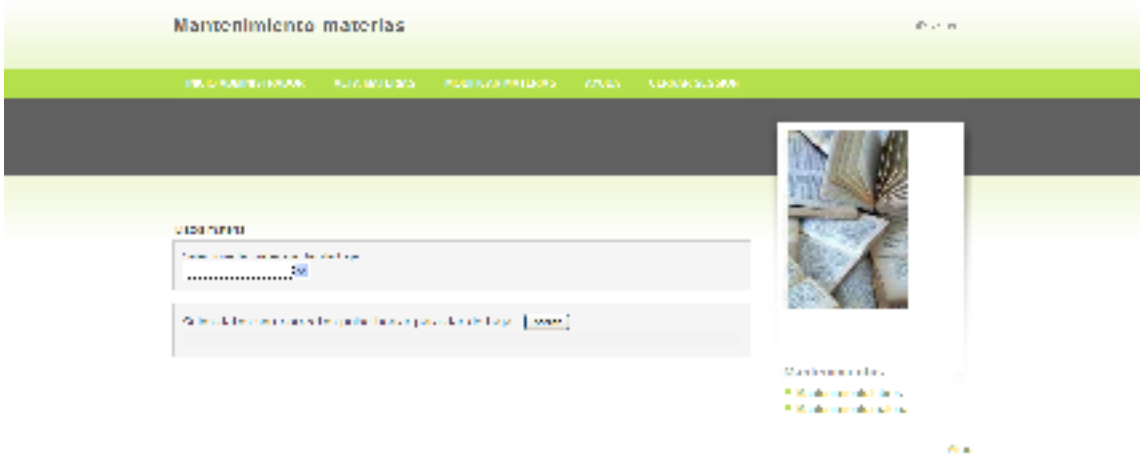


Figura 44. Pantalla baja materia

4.4 Implementación

4.4.1 Herramientas de desarrollo

Pera realizar este proyecto, el programari que se utilizara es el siguiente:

- Java 2 Platform, Enterprise Edition SDK 1.5: base del J2EE, útil para compilar y ejecutar el software desarrollo.
- MySQL 5.0: base de datos relacional muy utilizada en aplicaciones web, ya que es ideal en aplicaciones con una baja concurrencia en la modificación de datos, y donde hay un elevado numero de datos (lectura de datos) .
- MySQL Administrator: herramienta gratuita que permite administrar la base de MySQL de forma directa, permitiendo modificar fácilmente los datos existentes en ella, facilitando la modificación y creación de datos como los usuarios, esquemas de tablas, columnas y restricciones.
- Jakarta Struts 1.0.2.: framework de código abierto escrito en Java que apoya la creación de las aplicaciones web utilizando el patrón de diseño MVC.
- Internet Explorer 6.0:navegador utilizado pera acceder a los módulos web de la aplicación.
- Mozilla Firefox 2.0.8 : navegador utilizado pera acceder a los módulos web de la aplicación.
- Ant para facilitar la compilación, el empaquetado y el desplegamiento de las diferentes partes en que esta dividida la aplicación.
- Eclipse 3.2 utilizado como entorno de programación.
- Tomcat5.5 como servidor web.

4.4.3 Decisiones tomadas en la implementación

Patrón / herramientas / componentes	Motivación	Tecnología aplicada	Motivación
Aplicación distribuida	Dado que los usuarios potenciales de la aplicación a desarrollar no están situados en ningún sitio en concreto, pueden estar en cualquier sitio, la decisión de utilizar esta arquitectura esta clara.	J2EE	A parte de que, obviamente, el proyecto de estudios pertenece a esta área temática, es indudable que actualmente la arquitectura J2EE se ha convertido en un estándar “de facto” en el desarrollo de las aplicaciones Java distribuidas
Aplicación web	Como que la aplicación esta orientada al público en general, he creído que el medio más fácil para que	Tomcat 5.5	Para dar respuesta a esta arquitectura he decidido utilizar el servidor web Tomcat 5.5.

	llegue a todo el mundo es Internet.		Esta servidor a muy buen rendimiento, es gratuito, open source y implementa el Api J2EE
Modelo-Vista-Controlador	Tal que el mantenimiento futuro de la aplicación y sus ampliaciones sean más sencillas, he decidido de utilizar esta patrón. De esta manera puedo separar el modelo de datos, la persistencia, las vistas y los controladores que los gestionan.	Apache Struts	Aunque hay otros frameworks de desarrollo del patrón MVC para web i J2EE, por ejemplo Spring o Faces, me he decidido por Apache Struts por que hace mucho tiempo que se utiliza, por lo tanto dispone de muchas librerias y documentación de soporte, y además es gratuito.
Base de datos	Para almacenar los datos de la aplicación he decidido utilizar una base de datos relacional.	MySQL	Entre la opciones de las base de datos gratuitas, me decidido por MySQL por que es una base de datos relacional con un buen rendimiento para pequeñas y medianas aplicaciones.
Entorno RAD	Para que el desarrollo de las aplicación sea eficiente, creo que es muy importante utilizar uno de los entornos RAD existente (Rapid Application Development).	Eclipse	He decidido de utilizar este entorno de desarrollo por que, todo y que es gratuito, es muy potente en las diferentes fases de desarrollo (generación de código, compilación, depuración)
Sistema logs	Para disponer de datos de depuración cuando hay algún problema, creo es muy importante que la aplicación utilice un sistema de logs configurable.	Log4J	El Log4J es un sistema de logs muy configurable y que permite diferentes opciones: Nivel de criticidad del error. Activación / desactivación del loggin. Definición de diferentes tipos de log(consola, fichero, bbdd...)
Tiles Mantenimiento paginas jsp	Para facilitar el mantenimiento de los menús de las paginas jsp, ya que es una parte común a todas las páginas, creo que es muy útil implementar Tiles.	Tiles definition	Simplifica el desarrollo de las aplicaciones web y además permite la integración con struts.
Componente Web filtro	Como es una aplicación en la que se puede navegar por la parte publica sin una previa identificación, he decidido de poner un control de las páginas jsp por si se intenta entrar por otra página	Servlet filter	Un filtro o “servlet filter” es un componente Web que reside en el servidor junto con el resto de componentes de una aplicación web, fácil de implementar y permite, podemos introducir filtros

	que no se la de inicio y a si añadir la sesión de anónimo.		declarativamente, en web.xml, sin tener que modificar los recursos que finalmente se invocan.
Estilos	Para facilitar el cambio de formatos y posiciones en la mayoría de los componentes de la aplicación.	Fichero de estilos CSS	Fácil de utilizar, y permite en cualquier momento cambiar la presentación.

Inicialmente no tenia muy claro de cómo implementarla parte de la confirmación de la compra del usuario, y si necesitaría una clase cesta para ir añadiendo las compras. Finalmente he decidido no implementar una clase cestilla y utilizar una collection y la clase lineasVenta para ir añadiendo o quitando productos y guardar los datos de la compra del usuario correspondiente en sesión http.
Cuando el usuario confirma la compra se comprueba:

- Si ya ha iniciado sesión y perfil es de tipo 1 , si es así se efectúa la compra,
- En caso de que el perfil sea 2 administrador se le muestra un mensaje de perfil no autorizado ha hacer compras.
- En caso que el usuario de sesión sea anónimo se le presenta la pantalla de login y una vez debidamente identificado se tramita la compra.

También se ha decidido de añadir una opción más en el menú “ver pedidos pendientes”, de esta manera el usuario puede consultar los pedidos que todavía no se le han servido.

En cuanto al menú lateral de búsqueda “Temas” en la fase de análisis estaba pensado que fuera estático, se cargaban los temas existentes a la tabla correspondientes. Pero me encontré con el problema que cuando el administrador añadía un tema no aparecida en el menú al acceder a cualquier pagina con lo que he decidido de implementarlo de manera dinámica.

La clase de uso modificación materia finalmente no se implementara ya que solo tenemos el atributo nombre, entonces en la modificación se dará de baja y si es necesario se crea una nueva.

Para la capa de datos he hecho algunos cambios para simplificar y mejorar la estructura. La tabla super_usuario no es necesaria, ya que el usuario administrador se crea con un script en el momento de crear toda la estructura, y se guarda en la tabla usuarios con un perfil (“2”) distinto al resto de usuarios (“1”). De esta manera cuando se registra el usuario se comprueba el perfil y si es igual a 2 se le presenta la pantalla de administrador.

También se ha añadido la tabla provincias para facilitar el alta y modificación de usuarios, ya que de esta manera al presentar el formulario de alta / modificación al usuario se cargan en una combo todas las provincias.

Las tablas existentes en la base de datos son:

Tabla Usuario		
nif	String	Clave primaria
nombre	String	
Apellido1	String	
Apellido2	strnig	
Usuario	string	
password	string	
perfil	integer	

Tabla Usuarios_registrados entidad subclase de usuario		
nif	String	Clave primaria, clave foránea a Usuario
direccion		
portal	String	
piso	string	
urbanizacion	string	
Localidad	string	
Cp	String	
Poblacion	String	
Provincia	String	
Telefono	String	
Email	String	
cuenta	string	
Entidad	String	
Oficina	Sring	
Dig_control	String	
Estado	string	
Telefono2	String	
Fecha_nac	String	

Tabla Productos		
Id_producto	int	Clave primaria
autor	string	
titulo	string	
editorial	String	
isbn	String	
idioma	String	
descuento	float	
stock	int	
precio	Int	
Id_materia	Int	Referencia a materia
novedades	boolean	
descripcion	string	

Tabla materia		
Id_materia	Int	Clave primaria
nombre	string	

Tabla pedidos		
Id_pedido	int	Clave primaria
importe	float	
Estado	string	Por defecto pendiente
Log_usuario	String	Clave foranea a usuarioregistrado

Tabla linea de venta		
Id_linea	int	Clave primaria
Cantidad	int	
Id_producto	int	Clave foranea a producto
Id_pedido	int	Clave foránea a pedido
Nom_producto	String	
Precio	float	

Tabla provincias		
Cod_provincia	String	
Nom_provincia	String	

4.4.3 Resultado de la implementación

CAPA CLIENTE	**VISTA**	**JSP**: contiene las paginas jsp, tecnología basada en lenguaje java que permite incorporar contenido dinámico a las páginas web.

<table>
<tr><td rowspan="4">CAPA NEGOCIO</td><td>CONTROLADOR</td><td>ActionForm conjunto de clases que heredan del actionForm de Struts y que encapsulan los datos obtenidos en el formulario, también contienen código de validación para comprobar el formato de los datos.
Las clases Action, conjunto de clases que heredan del actino.class d’estruts y que despachan cada una de las peticiones realizadas por el usuario. Mediante el struts-config.xml Struts determina el action destinatario de cada acción y una vez ejecutada muestra la vista (jsp) correspondiente.
La clase ActionBaseIni es el padre de todas, es decir se ejecuta para cualquier petición de usuario, se encarga de controlar la sessión.</td></tr>
<tr><td rowspan="3">MODELO</td><td>FACADE: Conjunto de clases que representan el patrón Façade. Su función es integrar les funcionalidades del software, delegando en otras clases, de manera que podemos disponer de un conjunto de procesos accediendo a una única clase.</td></tr>
<tr><td>DAO:
Classes que cumplen el patrón data access object, encapsulando todos los accesos a los orígenes de datos. Este se encarga de conseguir i librar conexiones a la base de datos, a si como de obtener o registrar datos a la base de datos.</td></tr>
<tr><td>VO: classes Value Object
Classes que implementan el patrón Transfer Object i que transportan los datos entre los diferentes clases. Se llenan con el contenido proporcionado por los ActionForms o con los datos generados por los DAOs.</td></tr>
<tr><td colspan="2">CAPA DE DATOS</td><td>Base de datos: contiene la información persistente del sistema.</td></tr>
</table>

UTILS	En este paquete se incluyen utilidades diversas: Configuraciones y constantes del software. Administrador del pool de conexiones. Funciones de validaciones.

La finalidad básica de esta arquitectura es la de tener aislada los componentes y las funcionalidades para poder localizar errores y/o aumentar las funcionalidades rápidamente.

4.5 Requisitos de software

4.5.1 Configuración e instalación

Para un correcto despliegue de la aplicación newBooks es necesario tener configuradas algunas variables de entorno y software especifico.

Las explicaciones que detallare a continuación están hechas en base a un equipo con sistema operativo Windows xp y con las aplicaciones Tomcat, Mysql, y JDK 1.5.

4.5.1.1 Variables de entorno

Para un correcto funcionamiento de Java, se necesita tener configuradas algunas variables de entorno de Windows xp:

- Variables de entorno-> variables del sistema añadimos las siguiente variables (es un ejemplo de cómo están en mi equipo)
- **CATALINA_HOME**: "ponemos la ruta donde esta instalada el tomcat" C:\Archivos de programa\Java\Tomcat 5.5
- **JAVA_HOME** "ponemos la ruta donde esta instalada el JDK" :C:\Archivos de programa\Java\jdk1.5.0_04
- **PATH**: se añade al final de la línea la ruta donde este instalado el JDK apuntando a la carpeta bin C:\Archivos de programa\Java\jdk1.5.0_04\bin

4.5.1.2 Configuración Tomcat

Para una mayor seguridad se configura el tomcat para que se pueda acceder a la web empleando http sobre ssl.
Consiste en utilizar la herramienta keytool, distribuida con el kit de desarrollo de Java (jdk1.5.0_04), para generar un certificado autofirmado del servidor. También será necesario cambiar la configuración del fichero server.xml para activar un conector que permitirá acceder a Tomcat a través de https.

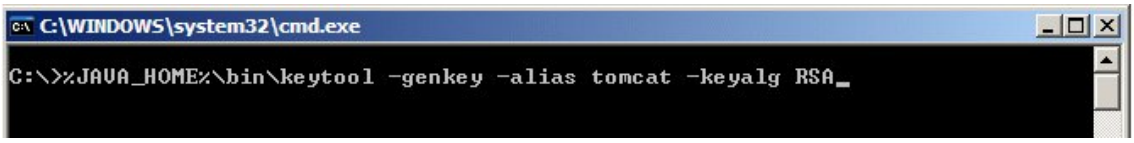
- **Configuranción del server.xml:** editamos el server.xml que esta en C:\Archivos de programa\Java\Tomcat 5.5\conf\server.xml Lo único que tenemos que hacer es descomentar la parte del siguiente elemento Connector:

<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->

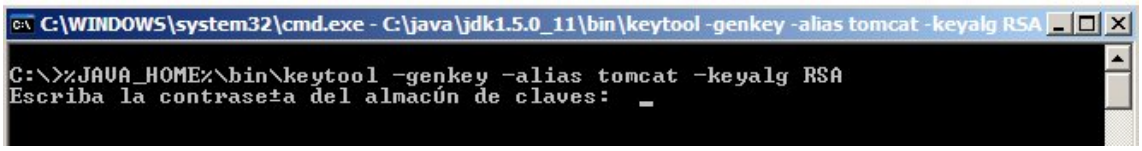
<!--
<Connector port="8443" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
-->
Con esto conseguimos activar el uso de SSL en el puerto 8443. No se solicita certificado al cliente durante el establecimiento de la conexión SSL (clientAuth = “false”), por tanto, no es necesario cambiar nada en el navegador web que usemos, aunque si hay que crear el almacén de claves en el servidor.

- **Creando el almacén de claves y un certificado autofirmado:** abrimos una consola y ejecutamos:

  %JAVA_HOME%\bin\keytool –genkey –alias tomcat –keyalg RSA



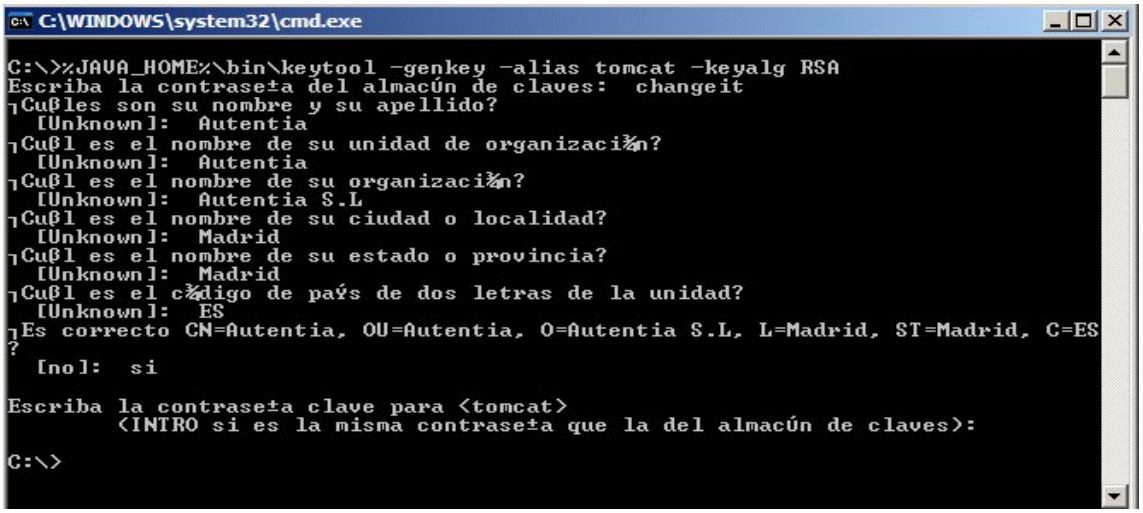
Nos pedirá una serie de datos para configurar el certificado. Lo primero que nos pide es la contraseña para el almacén de claves:



La clave por defecto utilizada por Tomcat es “changeit” y es la que introduciremos. Si queremos usar cualquier otra clave, lo único que tenemos que hacer es añadir al Connector el parámetro keystorePass especificando la clave a usar.

A continuación nos pide nuestro nombre, el nombre de nuestra unidad de organización, el nombre de nuestra organización, la ciudad o localidad, el estado o provincia, y el código del país. Nos pregunta si los datos introducidos son correctos e introducimos si.

A continuación nos pide la contraseña clave para <tomcat>. Aunque nos permite introducir una distinta, siempre debemos poner la misma contraseña aquí que la que tenemos en nuestro almacén de claves (Nos da la opción de pulsar INTRO para este efecto).



Y ya hemos terminado, el proceso nos ha creado un archivo .keystore en el home del usuario:

C:\Documents and Settings\marisol en mi caso con sistemas Windows XP

Tomcat utiliza por defecto este .keystore para buscar los certificados (el que se encuentra en el directorio home del usuario). Si quisieramos utilizar un almacén situado en otra ubicación, unicamente debemos añadir al Connector definido anteriormente en el server.xml el parámetro keystoreFile especificando la ruta absoluta al fichero,

keystoreFile="C:\keyStores\.mykeystore"

4.5.1.3 Instalación del producto

El producto vendrá en un fichero llamado mrodriguezgoy_producto.zip. Lo primero que se ha de hacer es descomprimir el contenido de este fichero, quedando una carpeta newBooks que contendrá la aplicación.
Una vez descomprimida esta carpeta, para que la aplicación funcione, se ha de desplegar en el servidor de aplicaciones, a si como preparar la base de datos MySql realizando los siguientes pasos:

- para empezar, es necesario crear la estructura de datos necesaria en MySql. Para hacerlo ejecutaremos un script de creación de base de datos y las tablas, el cual encontraremos en la carpeta newBooks obtenida al descomprimir el fichero .zip antes mencionado, este escript se llama newBooks.sql. lo copiaremos en **c:** , y para ejecutarlo se abre el MySQL para cargar el script de la BBDD dede inici->programas->MySQL Server 5.0-> MySQL Command Line Client entonces ya nos situamos en el directorio mysql> y ecribimos:
  mysql> **\. c:\newBooks.sql**
  La base de datos de la aplicación newBooks se cargara, este script también creara el usuario administrador de la aplicación.

- Nos situamos en la carpeta descomprimida anteriormente, y encontraremos un fichero llamado new_Books.war que se dejara en el

servidor de aplicaciones C: \Archivos de programa\Java \Tomcat 5.5\webapps

- También cogeremos el fichero mysql-connector-java-5.0.3-bin.jar y lo copiaremos en la carpeta C:\ Archivos de programa\Java\Tomcat 5.5\common\lib

Si tenemos el servidor parado hace falta arrancarlo.
Desde un navegador web se puede acceder a la siguiente url a través de una conexión segura:
https://localhost:8443/new_books/
Si el proceso de instalación es correcto debería aparecer la página web de inicio y poder navegar.

Para que el administrador pueda acceder a la parte correspondiente de administración de la aplicación se ha de registrar como cualquier usuario, o sea acceder por el menú superior "Registro" -> Identificación de usuarios registrados e introducir el usuario y password correspondiente del administrador, que en este caso es usuario "cristina", password "cristina" y si es debidamente identificado le aparecerá el menú del administrador.

4.6 Valoraciones finales

4.6.1 Conclusiones

La experiencia del TFC ha sido positiva y ha cubierto mucho más de lo esperado las expectativas que había puesto de antemano en está asignatura.

He adquirido un conocimiento más que suficiente en la arquitectura MVC, que me ha parecido muy interesante y portable a cualquier plataforma.

He valorado las ventajas de utilizar un lenguaje OO con la facilidad de incorporar componentes y la reutilización de las clases.

Por primera vez he trabajado con el servidor web (Tomcat) que he tenido que configurar y sobre el que he desplegado la aplicación.

Por último, es el primer trabajo en el que realizo completamente el ciclo de vida de un proyecto. Para su realización me han sido de ayuda la experiencia adquirida en las asignaturas de Ingeniería del Software I y la de Técnicas de Desarrollo del Software. También me ha ayudado el haber abordado un proyecto cuyo contenido y requerimientos ya estaban conceptualmente definidos en mi mente.

4.6.2 Mejoras para próximas versiones

La aplicación está sujeta a muchas posibles ampliaciones y mejoras, muy probablemente el propio uso que se haga de ella por parte de los usuarios finales muestren otras que inicialmente no se habían contemplado.
Una vez realizado el desarrollo aparecen las siguientes posibilidades:

- Notificación de compra, generar un correo destinado al cliente cuando este hace una compra, notificando de esa manera el nº de pedido y la compra realizada.

- Cambiar el acceso a la base de datos utilizando HIBERNATE, de esta manera cualquier modificación en la base de datos no afectara a las consultas realizadas en el código java.

- En el formulario de alta usuario para provincias y poblaciones hacerlo utilizando ajax, o sea cuando seleccionamos una provincia que solo salgan en una combo las poblaciones correspondiente a esa provincia, ya que ahora la población se introduce manualmente.

- Seria interesante hacer alguna promoción para los clientes registrados en las base de datos enviando e-mails.

- Hacer unas estadísticas de los libros más vendidos y una exportación al Excel.

- Establecer más facilidades de pago de los pedidos, con cheque, tarjeta de crédito o través de servicios de pago electrónico.

5. Glosario

CSS: Son las hojas de estilo en cascada (Cascading Style Sheets, CSS). Es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

ER: Diagramas de Entidad Relación. Es un modelo conceptual para la representación gráfica del diseño de las bases de datos.

MVC: Modelo, Vista y Controlador. Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz gráfica y la lógica de control.

OO: Orientación a Objetos. Es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado, comportamiento e identidad.

Open source: Código abierto (open source en inglés).

TILES: es un framework que se utiliza para simplificar el desarrollo de una aplicación web y el uso de interfaces.

J2EE: arquitectura completa de n-capas basada en la plataforma Java.
Java: lenguaje para la elaboración de aplicaciones exportables a la red y capaz de operar sobre cualquier plataforma.

JDBC: acrónimo de Java Data Base Connectivity, permite operaciones sobre base de datos desde el lenguaje de programación java, independientemente del sistema de operación donde se ejecute o la base de datos a la cual se quiera acceder utilizando el dialecto SQL del modelo de base de datos que se utilice.

JSPs: extensión de los servlets que permite de forma fácil la fusión del código con paginas HTML estándares.

Login: Momento de autentificación para acceder a un servicio.

MySQL: sistema de gestión de base de datos multiusuario.

Patrones de diseño: son la base para la búsqueda de soluciones a problemas comunes en el desarrollo del software y en otros ámbitos referentes al diseño de iteración o de interfícies .

RUP: siglas de Rational Unified Process.

Servlets: es una clase Java que puede cargarse dinámicamente para extender la funcionalidad del servidor web.

SGBDs: relacionales: implementación de base de datos relacionales que soporta todos los aspectos del modelo relacional incluidos los dominios y las dos reglas generales de integridad.

Struts: herramienta de soporte para el desarrollo de aplicaciones Web con el patrón MVC bajo la plataforma J2EE.

UML: siglas del lenguaje Unificado del modelo, lenguaje grafico para visualizar, especificar, construir y documentar un sistema de software.

Usuario: cualquier persona que se conecta a través de Internet a la página web de newbooks.

Usuario Registrado: usuario que ha rellenado el formulario de registro y se ha identificado en el sistema.

6. Bibliografía y referencias

En la bibliografía se incluye, además del material de consulta utilizado, referencias artículos de Internet que son de inclusión necesaria ya que se han utilizado en el desarrollo del proyecto.

Falkner Jayson, Galbraith Ben, Irani Romin, Kochmer Casey, Narayan Sathya, Perrumal Krishnaraj, Timmey John, Moidoo Meeraj. (2001). *Fundamentos Desarrollo Web con JSP*. Madrid: Ediciones Anaya Multimedia

Eckel Bruce, (2002) *Piensa en Java*.(segunda edición) Madrid: Prentice Hall

Bernaus Albert, Blanco Jaume. (1996). *Diseño de Paginas Web*. Barcelona:

Inforbook's Hall Marty (1998). *Core Servlets and JavaServer Pages.*
http://csajsp-chapters.corewebprogramming.com/Core-Servlets-and-JSP.pdf

MpCon. *Instalación y configuración Tomcat y MySQL.*
http://mpcon.org/apacheguide/

MySQL AB. *Tutorial básico de MySQL*
http://www.mysql-hispano.org/page.php?id=6&pag=1

Universidad de las Palmas de Gran Canaria. *Tutorial de JavaScript*.
http://www.ulpgc.es/otros/tutoriales/JavaScript/index.htm

TagLibs y JSPs.
http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=taglibs
http://struts.apache.org/1.x/struts-taglib/tlddoc/index.html

DesarrolloWeb.com "*CSS, hojas de estilos*
http://www.desarrolloweb.com/manuales/2/
http://www.free-css-templates.com

Jakarta.apache.org. *The Apache Jakarta Tomcat 5 Servlet/JSP Container*.
http://tomcat.apache.org/tomcat-5.0-doc/jndi-datasource-examples-howto.html

Instalación de Tomcat 5 y Pool de Conexiones.
http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=tomcat5

Programacion.com. *J2EE.*
http://www.programacion.com/java/articulos/J2EE/

Merelo J., *Programando con JSP*.
http://kal-el.ugr.es/~jmerelo/JSP/

Introducción a JSP.
http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSP.pdf

Control de navegación en Servlets.
http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=servletsbasico

Servlet Context
http://www.linti.unlp.edu.ar/tiki-download_file.php?fileId=347.

Java filters
http://mipagina.cantv.net/igvir/blog/Filters_en_la_Practica.pdf

7. Anexos

7.1 Preparación del entorno de desarrollo

Para la preparación del entorno de desarrollo seguiremos los siguientes pasos

- Crear un directorio en **c:\TFC**
- Abrir el eclipse y crearemos el workSpace en la carpeta creada **c:\TFC**
- Ahora tenemos que crear la estructura del proyecto. Tenemos 2 proyectos uno para las tareas ant (new_books_utils) y otro que es el proyecto (new_books), para crearlos haremos lo siguiente:

    - Para crear un proyecto web desde el eclipse:
      File-> new ->project->web->Dynamic web project-> next ahora nos pedirá el nombre del proyecto teclearemos **new_books**
    - Para crear un proyecto J2EE desde el eclipse:
      File-> new ->project->J2EE->Enterprise Application project-> next ahora nos pedirá el nombre del proyecto teclearemos **new_books_utils**

    - A continuación cogemos los ficheros fuentes de los 2 proyectos, que están en mrodriguezgoy_producto.zip (new_books_utils y new_books y los copiamos en el directorio creado en el primer punto c:\TFC. Como las carpetas new_books_utils y new_books ya existen nos dirá de sobrescribir , le diremos que si, de esta manera ya tenemos nuestra estructura y código fuente listo.
    - Ahora nos vamos al eclipse y pulsaremos F5 para actualizar todo los proyectos. En esta punto ya tenemos el proyecto completo con todas las librerías, en el caso de faltar alguna tendríamos que importarlas.
    - Para compilar utilizamos el Ant que esta en new_books_utils. Añadiremos una vista Ant y dentro de esta vista añadimos el fichero build.xml (tareas ant creadas) que esta en new_books_utils dentro de la carpeta ant. Ahora ya podemos compilar haciendo doble clic en el en new_books de la vista ant. Esta tarea nos desplegara el new_books.war en el tomcat. Si necesitamos cambiar alguna ruta de acceso de la tarea ant tenemos el fichero build.properties en el que podemos especificar la ruta deseada.