Desarrollo de una tienda virtual.

Rafael Doña Gil

Enginyeria Tècnica en Informàtica de Sistemes

Consultor: **Jose Juan Rodríguez**

14 de Enero de 2013

DEDICATORIA Y AGRADECIMIENTOS

Lo que en principio me planteé como un puro trámite para obtener una titulación necesaria para conseguir otro objetivo, me ha permitido que descubra el apasionante mundo de la informática. Por tanto, con la finalización de este trabajo no he conseguido solo el objetivo con el que inicie hace 3 años mi aventura en la UOC, sino que he descubierto un sector en el cual me gustaría desarrollar mi vida profesional.

Quiero agradecer el apoyo de toda mi familia, que ha estado siempre a mi lado y me ayuda a seguir adelante, en especial a dos personitas que me contagian de su energía día a día. También agradecer la atención de mis compañeros en los foros de todas las aulas por las que he pasado, con los que compartimos tantas dudas y en los que me vi tantas veces reflejado.

Por último agradecer la labor de todo el profesorado por asistirme a los largo de estos últimos 3 años y ayudar a hacerme entender este apasionante mundo.

RESUMEN.

El siguiente documento corresponde a la memoria de un proyecto de final de carrera, englobado dentro del área J2EE. Desde hace unos años la programación orientada a objetos para el desarrollo de programas está cogiendo más fuerza. Este hecho, junto con que cada vez existan más empresas que utilicen la red para desarrollar su negocio o parte de él, es el motivo por el que me he decidió a desarrollar un proyecto dentro de esta área temática.

El proyecto consiste en el desarrollo de una tienda virtual dedicada a la venta de vino ecológico. El objetivo a grandes rasgos de la tienda virtual no debe ser diferente al de una tienda física. La aplicación deberá mostrar a los clientes los productos de los que dispone la tienda virtual y les deberá permitir seleccionar los que deseen. Una vez seleccionados, el cliente deberá "pasar por caja" para pagar su compra.

Para el desarrollo de la aplicación se ha utilizado la tecnología J2EE (Java Enterprise Edition), que es una plataforma Java para el desarrollo de aplicaciones empresariales. Además de la propia tecnología J2EE, se han utilizado patrones de diseño (MVC), frameworks (struts2, javaEmail), contenido web (jsp, css), tratamientos de bases de datos (SQL). Por tanto el producto resultante es una aplicación web, donde los usuarios podrán acceder desde cualquier lugar con una conexión Internet.

Esta memoria intenta resumir el trabajo realizado durante el desarrollo del proyecto, así como las experiencias y conocimientos adquiridos durante la realización del mismo.

Palabras clave: J2EE, struts2, MVC, javaEmail.

INDICE

1. Introducción…… 6
1.1 Justificación del TFC…… 6
1.2 Objetivo…… 6
1.3 Enfoque y método a seguir…… 7
1.4 Planificación del proyecto…… 9
1.5 Productos obtenidos…… 10
1.6 Descripción de los capítulos de la memoria…… 10
2. Análisis funcional…… 11
2.1 Actores…… 11
2.2 Casos de uso…… 12
2.2.1 Casos de uso del actor Visitante…… 12
2.2.2 Casos de uso del actor Usuario…… 18
2.2.3 Casos de uso del actor Administrador…… 21
2.2.4 Casos de uso de toda la aplicación…… 31
3. Diseño Técnico…… 32
3.1 Arquitectura…… 32
3.1.1 Patrón MVC…… 33
3.1.2 Capa Vista…… 35
3.1.2.1 JSP…… 35
3.1.2.2 Biblioteca Jquery…… 36
3.1.3 Capa Controlador…… 36
3.1.3.1 Struts 2…… 36
3.1.3.2 JavaMail…… 39
3.1.4 Capa Modelo…… 39
3.1.4.1 Data Access Object (DAO)…… 39
3.1.5 IDE. Netbeans…… 41
3.2 Diagrama de clases…… 42
3.3 Diseño de la base de datos…… 43
3.5 Diseño interficie gráfica…… 44
4. Implementación…… 49
4.1 Requisitos de software…… 49
4.2 Preparación del entorno de trabajo…… 50
4.3 Librerías…… 50
4.4 Seguridad…… 51
4.5 Estructura de la aplicación…… 51
4.6 Ficheros de configuración…… 54
5. Conclusiones…… 57
6. Bibliografía…… 59

INDICE DE FIGURAS

Fig. 1. Etapas del modelo en cascada …… 8
Fig. 2. Diagrama de Gantt…….. 9
Fig. 3. Relación entre actores…….. 11
Fig. 4. Casos de uso actor Visitante…… 12
Fig. 5. Casos de uso actor Usuario…… 18
Fig. 6. Casos de uso actor Administrador…… 21
Fig. 7. Casos de uso actor de la aplicación…….. 31
Fig. 8. Esquema general de aplicación web…….. 32
Fig. 9. Esquema patrón MVC…….. 34
Fig. 10. Relación J2EE y MVC…….. 35
Fig. 11. Esquema funcionamiento struts2…… 38
Fig. 12. Estructura capa Modelo utilizando DAO…….. 40
Fig. 13. Estructura patrón de diseño DAO…… 41
Fig. 14. Diagrama de Clases…… 42
Fig. 15. Diagrama de E/R…….. 43
Fig. 16. Estructura general de la Interficie gráfica. …… 45
Fig. 17. Estructura página registro…… 46
Fig. 18. Estructura página listado de productos…… 47
Fig. 19. Estructura página zona administración…… 47
Fig. 20. Estructura de los ficheros de la aplicación…… 53
Fig. 21. Fichero de configuración web.xml…….. 54
Fig. 22. Fichero de configuración context.xml…… 55
Fig. 23. Fichero de configuración struts.xml…… 55
Fig. 24. Fichero de configuración registerAction-validation.xml…… 56

1. Introducción.

El trabajo de final de carrera (TFC) está pensado para realizar un trabajo de síntesis de conocimiento adquiridos en otras asignaturas con el fin de ponerlos en práctica en un proyecto concreto. El TFC que se presenta a continuación consiste en desarrollar una tienda virtual dedicada a la venta de vino ecológico. El objetivo a grandes rasgos de la tienda virtual no es diferente al de una tienda física. La aplicación deberá mostrar a los clientes los productos de los que dispone la tienda virtual y les deberá permitir seleccionar los que deseen. Una vez seleccionados, el cliente deberá "pasar por caja" para pagar su compra.

El desarrollo del proyecto se ha dividido en varias etapas. La primera etapa ha consistido en definir un Plan de Trabajo donde se ha realizado una descripción del TFC, se han definido los objetivos generales y específicos y se ha planificado temporalmente la realización del proyecto. La segunda etapa ha consistido en la realización del análisis funcional y del diseño de la arquitectura de la aplicación. Una vez definido el análisis funcional se ha procedido a la etapa de implementación, donde se ha dado forma a la idea del TFC. Una vez implementada la aplicación se ha procedido a la etapa de pruebas, donde se ha depurado la aplicación de errores en etapas anteriores. Finalmente con la realización de esta memoria y una presentación se pretende sintetizar el desarrollo de este proyecto.

1.1. Justificación del TFC.

Este proyecto está enmarcado dentro del área J2EE. Desde hace unos años la programación orientada a objetos para el desarrollo de programas está cogiendo más fuerza. J2EE (Java Enterprise Edition), es una plataforma Java para el desarrollo de aplicaciones empresariales y se ha convertido en un estándar en el mundo de desarrollo distribuido de aplicaciones empresariales por internet. Este hecho, junto con que cada vez existan más empresas que utilicen la red para desarrollar su negocio o parte de él, es el motivo por el que me he decidió a desarrollar un proyecto dentro de esta área temática.

1.2. Objetivo

El objetivo principal de este proyecto es profundizar en el uso de la tecnología java, ya vista en asignaturas anteriores, y en este caso conocer la arquitectura J2EE, a partir del análisis, diseño e implementación de una aplicación basada en esta arquitectura. J2EE (también conocida como JEE) es una arquitectura que permite desarrollar aplicaciones distribuidas para internet utilizando el lenguaje de programación java (POO).

Además de la propia tecnología J2EE, para el desarrollo de este proyecto se han utilizado patrones de diseño (MVC), frameworks (struts, javaEmail), contenido web (jsp, css), tratamientos de bases de datos (SQL),….

Todas estas tecnologías son las que nos han permitido llevar a cabo la obtención la aplicación desarrollada que permite:

- Gestionar una base de datos con todos los productos que existen a la tienda, usuarios registrados, datos de las compras, etc…
- Permite que un cliente pueda registrarse en la aplicación y guardar sus datos en la base de datos.
- Avisa al administrador cuando un cliente se ha registrado a través de e-mail.
- Controla si durante el inicio de una sesión como usuario registrado, se está accediendo a la tienda como un cliente o un administrador, y proporciona para cada caso sus respectivas herramientas:

  Cliente:

  - Busca los productos agrupados por familias, ofertas, más comprados,…
  - Gestión del carrito de la compra (ver su contenido, añadir un producto, eliminar un producto, modificar la cantidad de artículos)
  - Gestión de la compra (tramitar el pedido, indicar datos de facturación, envío, forma de pago, etc…)
  - Modificar datos de usuario registrado.
  - Realizar consultas al administrador.
  - etc..

  Administrador:

  - Gestión de productos (añadir, eliminar, modificar, indicar ofertas, …)
  - Gestión de clientes (ver listado clientes registrados, eliminar cliente, …)
  - Gestión compras (ver listado compras, generar facturas, estadísticas …)
  - Registrar más administradores de la aplicación.
  - etc..

1.3. Enfoque y método a seguir

Para el desarrollo de este proyecto se ha seguido las siguientes etapas:

- **Plan de trabajo.** En esta etapa se han definido los objetivos generales y específicos y se ha planificado temporalmente la realización del proyecto

- **Análisis funcional del proyecto.** Define los requerimientos específicos del proyecto y como se producirá la interacción de los diferentes actores con el producto final.
- **Diseño de la arquitectura.** Define los requisitos de diseño de la arquitectura que tendrá la aplicación.
- **Implementación.** Es esta etapa es donde se ha desarrollado la aplicación. Está implementación se ha realizado de forma iterativa.
- **Pruebas.** En esta etapa se ha depurado la aplicación de posibles errores en etapas anteriores.
- **Memoria y presentación:** Finalmente se ha documentado el desarrollo del proyecto.

Estas etapas coinciden con las etapas del modelo en cascada. Este modelo define los estados a través de los cuales se mueve un proyecto de desarrollo de software. El final de una etapa indica el inicio de la siguiente. El modelo en cascada puro no se suele utilizar tal cual, ya que implicaría un previo y absoluto conocimiento de los requisitos y que las etapas estén libres de errores. Como podemos pensar, esto es utópico ya que las aplicaciones software son cambiantes y difícilmente libres de errores. Por tanto, permitiremos una realimentación entre etapas, permitiendo retroceder de una a la anterior (e incluso poder saltar a varias anteriores) si es necesario.

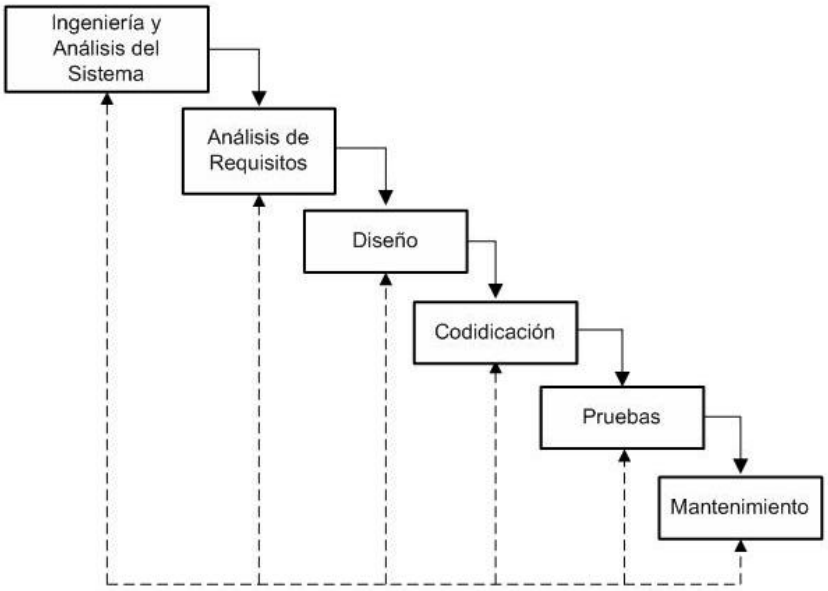


Fig 1. Etapas del modelo en cascada [19]

1.4 Planificación del proyecto.

La planificación del proyecto tiene que contemplar las fechas límites de entrega de las diferentes tareas parciales de las que está formado el modelo de evaluación continuada de la UOC. Estas tareas parciales vienen definidas por:

Entrega	**Fecha**
PAC1. Plan de Trabajo	03/10/2012
PAC2. Análisis Funcional y Diseño de la Arquitectura	08/11/2012
PAC3. Implementación	17/12/2012
Memoria y presentación	14/01/2013

A partir de estas entregas definidas he realizado una descomposición de tareas, que me ha permitido realizar los diferentes informes de entrega de cada una de las PACs. Esta descomposición de tareas la representaré en un diagrama de Gantt, donde se define la planificación temporal de cada una de las tareas.

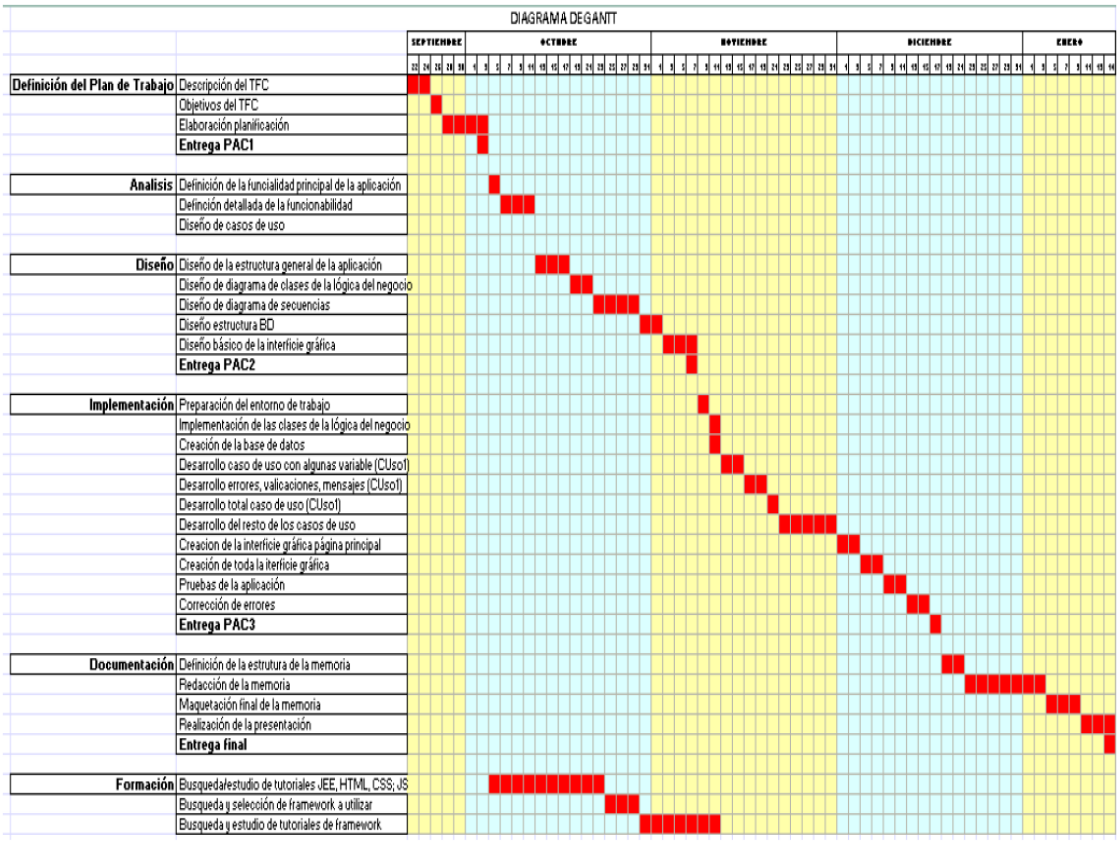


Fig 2. Diagrama de Gantt

1.5. Productos obtenidos.

El desarrollo de este proyecto ha generado los siguientes productos:

- **Una aplicación web:** Esta aplicación web se entrega en formato .war siguiendo el estándar J2EE. Este fichero contendrá todas las clases, ficheros de configuración xml, paginas jsp, hojas de estilos css, ficheros con javaScript, librerías…. Por tanto a partir de este fichero podremos ejecutar la aplicación tal y como se explicará en los siguientes apartados.
- **Memoria:** Documento donde se sintetiza todo el proceso de desarrollo del proyecto.
- **Presentación:** Presentación en PowerPoint con las conceptos que se explican en la memoria pero aun más sintetizados.

1.6. Descripción de los capítulos de la memoria.

La memoria está dividida en 4 capítulos principales.

- **Análisis funcional:** En este capítulo se definirá las especificaciones de la aplicación. Para definir estas especificaciones estudiaremos los diferentes actores que interactuarán con la aplicación y como será esta interactuación. Los casos de usos nos permitirán definir las especificaciones que debe cumplir nuestra especificación de una forma detallada.

- **Diseño de la arquitectura.** Es este capítulo se mostrará el diagrama de clases de la lógica del negocio, diseño de la estructura de la base de datos, la definición de la arquitectura de la aplicación (IDE, frameworks, SGBD, bibliotecas, ….) y se mostrará la interficie gráfica principal de la aplicación.

- **Implementación.** En este capítulo se explicarán las decisiones de diseño que se han llevado a cabo en el desarrollo de la aplicación, el software utilizado, la estructura de la aplicación y una detalla descripción para el despliegue de la aplicación.

- **Conclusiones.** Es este último capítulo se describen cuales con las conclusiones extraídas de la realización del proyecto.

2. Análisis funcional

En este apartado se definirá el comportamiento que debe tener la aplicación. Para definir este comportamiento estudiaremos los diferentes actores que interactuarán con la aplicación y como será esta interactuación. Para realizar el análisis funcional de la aplicación se realizará a través de los casos de uso. En primer lugar definiremos los actores.

2.1. Actores.

Un Actor es un rol que un usuario juega con respecto al sistema. Podemos distinguir tres tipos de actores diferentes que interactuarán con nuestra aplicación. Cada uno podrá realizar las operaciones permitidas para su perfil, por lo que se tendrá que crear un sistema de login para que la aplicación pueda saber el tipo de usuario que está interactuando. Los tres tipos de actores son:

Visitante. Será un usuario del cual no conoceremos ningún dato (será anónimo). A este tipo de usuario se le permitirá realizar una gran cantidad de operaciones, exceptuando la tramitación de un pedido, autentificarse,...

Usuario registrado. Este usuario, será una generalización de Visitante (también podrá realizar todas las operaciones que puede realizar Visitante). Este tipo de usuario podrá realizar todas las operaciones que se le permiten a los usuarios clientes.

Administrador. Este usuario es el encargado de hacer las tareas de administrador de la aplicación (gestión de artículos, clientes, compras, estadísticas,). Es una generalización de Usuario registrado, por lo que además podrá realizar todas las tareas que puede realizar un usuario cliente.

A continuación se muestra como es relación que existe entre los 3 tipos de actores:

Fig. 3. Relación entre actores

2.2. Casos de uso.

Los diagramas de casos de uso documentan el comportamiento del sistema por parte de un usuario. Nos servirán, pues, para mostrar las funcionalidades que debe tener el sistema desde el punto de vista de sus interacciones con el exterior [1]. Por supuesto, no entraremos en una descripción detallada de que algoritmos que necesitaremos para resolver la funcionalidad.

Para facilitar la comprensión se mostraran los diagramas de casos de uso separados por actores y seguidamente la descripción de cada uno de los casos de uso asociados a cada actor.

2.2.1 Casos de uso del actor Visitante

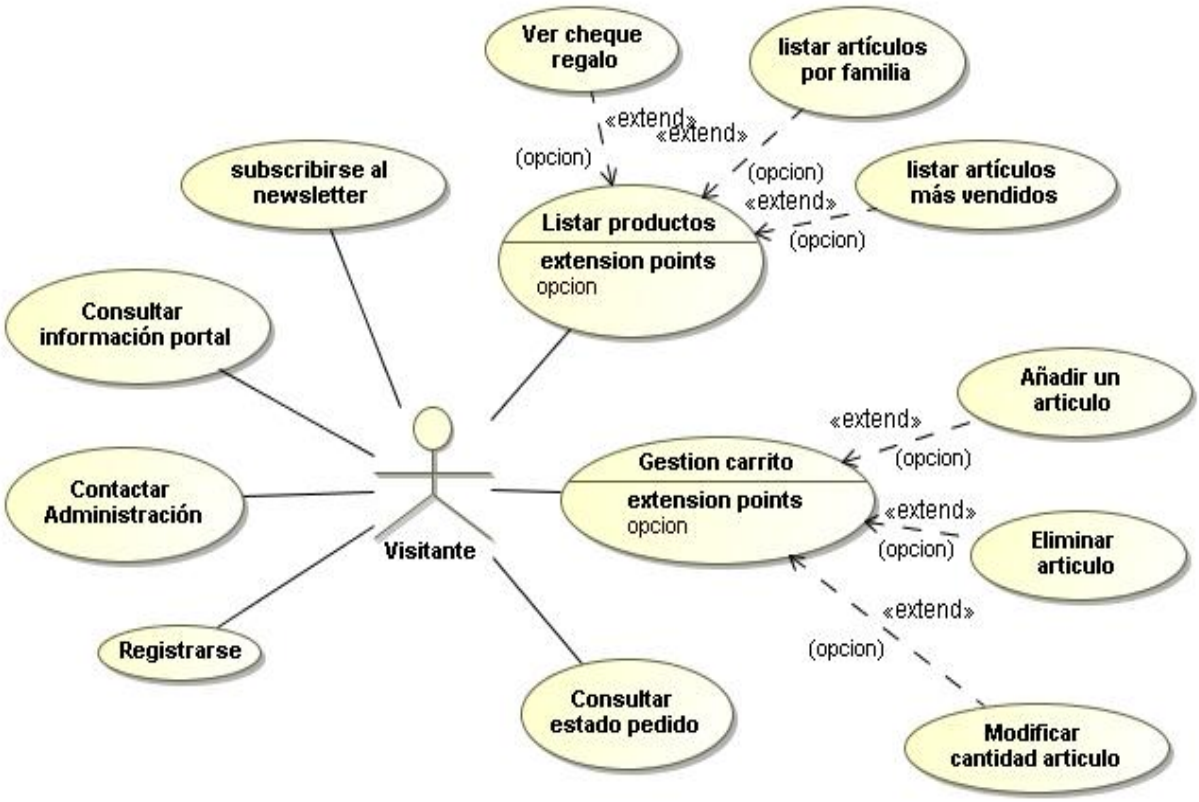


Fig. 4. Casos de uso actor Visitante

Caso de Uso	**Gestión carrito**
Resumen de la funcionalidad	Mostrará un listado con el contenido del carrito de la compra
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Se mostrará el contenido del carrito.
Proceso normal principal	El usuario clicará el botón “Mi carrito”. El sistema redireccionará al usuario a la página correspondiente, donde se mostrará toda la información del contenido del carrito (artículos, coste,..)
Alternativas de proceso y excepciones	Si el carrito no tiene productos se mostrará el mensaje correspondiente.

Caso de Uso	**Modificar cantidad artículo**
Resumen de la funcionalidad	Modificará la cantidad de unidades compradas de un artículo específico.
Actores	Visitante, Usuario y Administrador
Precondición	El usuario a ejecutado el caso de uso Gestión carrito
Poscondición	Se mostrará el contenido del carrito actualizado.
Proceso normal principal	Desde el listado del contenido del artículo, el usuario clicará el botón “Unidades” y seleccionará un nuevo número de unidades desea. Una vez seleccionada clicará el botón “Modificar”. El modificará la cantidad de unidades del artículo y mostrará el contenido de carrito actualizado.
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Eliminar artículo**
Resumen de la funcionalidad	Eliminará el artículo deseado del carrito de la compra.
Actores	Visitante, Usuario y Administrador
Precondición	El usuario a ejecutado el caso de uso gestión carrito
Poscondición	Se mostrará el contenido del carrito actualizado.

Proceso normal principal	Desde el listado del contenido del artículo, el usuario clicará el botón "eliminar" del artículo que desea borrar del carrito. El sistema eliminará el artículo del carrito de la compra y mostrará su contenido actualizado
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Añadir artículo**
Resumen de la funcionalidad	Añadirá un artículo al carrito de la compra.
Actores	Visitante, Usuario y Administrador
Precondición	El usuario a ejecutado el caso de uso listar productos
Poscondición	Se mostrará el contenido del carrito con el nuevo producto añadido.
Proceso normal principal	El Usuario clicará el botón "comprar" del articulo deseado. El sistema añadirá el artículo al carrito de la compra y redireccionará al usuario a la página correspondiente, donde se mostrará toda la información del contenido del carrito.
Alternativas de proceso y excepciones	Si el producto ya existe en el carrito se mostrará el mensaje correspondiente al usuario

Caso de Uso	**Listar ofertas**
Resumen de la funcionalidad	Se encarga de mostrar un listado de los productos en oferta.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Se mostrará un listado de los productos más vendidos
Proceso normal principal	El Usuario clicará el botón "nuestras ofertas". El sistema mostrará el listado de productos en oferta que están situados en la tabla "Ofertas".
Alternativas de proceso y excepciones	Si no hay productos en la tabla "Ofertas" se mostrará el mensaje correspondiente al usuario

Caso de Uso	**Listar más vendidos**
Resumen de la funcionalidad	Se encarga de mostrar un listado de los productos más vendidos.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Se mostrará un listado de los productos más vendidos
Proceso normal principal	El Usuario clicará el botón "productos más vendidos". El sistema mostrará el listado de productos más vendidos que están situados en la tabla "masVendidos".
Alternativas de proceso y excepciones	Si no hay productos en la tabla "masVendidos" se mostrará el mensaje correspondiente al usuario

Caso de Uso	**Listar por familia**
Resumen de la funcionalidad	Se encarga de mostrar un listado de los productos que corresponden a la familia seleccionada.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Se mostrará un listado de los productos de una familia
Proceso normal principal	El Usuario clicará el botón de la familia la cual desea ver un listado de sus productos (tinto, blanco, rosado o cava). El sistema mostrará el listado correspondiente.
Alternativas de proceso y excepciones	Si la familia seleccionada no tiene ningún producto se indicará el mensaje correspondiente al usuario

Caso de Uso	**Ver cheque regalo**
Resumen de la funcionalidad	Se encarga de redireccionar a la página donde está la información del cheque regalo.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Se mostrará la página con la información del cheque regalo
Proceso normal principal	El Usuario clicará el botón "cheque regalo" y el sistema redireccionará al usuario a la página correspondiente
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Listar productos**
Resumen de la funcionalidad	Se encarga de listar los productos que existen en la base de datos en función de alguna característica.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Se mostrará el listado de los productos seleccionados
Proceso normal principal	El Usuario ejecuta el caso de uso cheque regalo, listar por familia, listar más vendidos o listar ofertas
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Contactar con administración**
Resumen de la funcionalidad	Se encarga de enviar una consulta por parte de un usuario al e-mail de administración.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Si el formato de los datos introducidos es correcto se envía una consulta a administración y una copia al usuario.
Proceso normal principal	El Usuario clicará en el botón “contacta” y será redirigido a la página correspondiente donde cumplimentará los datos del formulario y clicar el botón “enviar”. El sistema enviará un e-mail con los datos de la consulta al e-mail de administración y una copia del mismo e-mail al usuario.
Alternativas de proceso y excepciones	Si el formato de los textos no es correcto se mostrará el mensaje de error correspondiente.

Caso de Uso	**Subscribirse al newsletter**
Resumen de la funcionalidad	Se encarga de añadir el e-mail facilitado por el usuario a la tabla newsletter, para que el usuario reciba el newsletter por e-mail
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Si el formato del e-mail es correcto, se añadirá a la tabla newsletter
Proceso normal principal	El Usuario introducirá su e-mail en la caja destinada a ello, y clicará el botón enviar. El sistema añadirá el e-mail a la tabla newsletter

Alternativas de proceso y excepciones	Si el formato de los textos no es correcto se mostrará el mensaje de error correspondiente. Si el e-mail ya existe en la tabla newsletter se mostrará un mensaje de error en la misma página.

Caso de Uso	**Registrarse**
Resumen de la funcionalidad	Permitirá a un visitante poder registrarse para poder abrir una sesión como usuario registrado.
Actores	Visitante, Usuario y Administrador
Precondición	ninguna
Poscondición	Si los datos son correctos se añadirá un nuevo usuario en la base de datos
Proceso normal principal	El usuario clicará el botón “Registrarse”. El sistema redireccionará al usuario a la página correspondiente, donde se mostrará un formulario con los datos necesarios para poder registrarse. Una vez introducidos los datos el usuario clicará en el boto “Registrarse”. La aplicación añadirá un nuevo usuario a la tabla “usuarios” de la base de datos.
Alternativas de proceso y excepciones	Si el e-mail indicado corresponde con el de un usuario ya registrado se mostrará el mensaje de error correspondiente. Si el formato de los textos no es correcto se mostrará el mensaje de error correspondiente. el carrito no tiene productos se mostrará el mensaje correspondiente.

Caso de Uso	**Contactar estado del pedido**
Resumen de la funcionalidad	Mostrará al usuario en qué estado se encuentra su pedido.
Actores	Visitante, Usuario y Administrador
Precondición	El usuario se ha autentificado en el sistema y a realizado un pedido.
Poscondición	Se muestra el estado del pedido.
Proceso normal principal	El Usuario introducirá el número de pedido en la caja destinada a ello y clicará el botón “Consultar”. El sistema comprobará el estado del pedido en la tabla “Pedido” y mostrará el resultado al usuario.
Alternativas de proceso y	Si el usuario no se ha autentificado en el sistema se redireccionará a la página de login.

excepciones	Si el número de pedido no corresponde con ningún número de la base de datos se mostrará un mensaje de error correspondiente.

2.2.2 Casos de uso del actor Usuario.

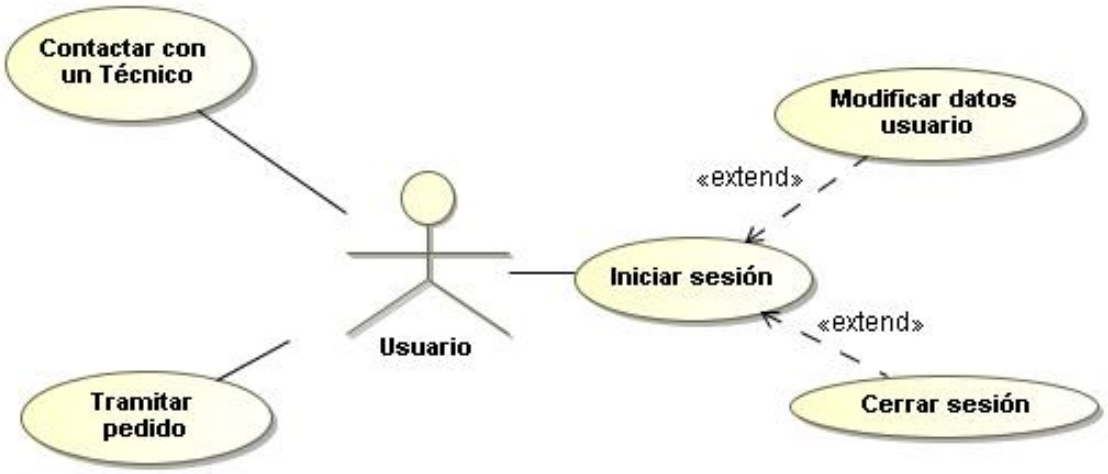


Fig. 5. Casos de uso actor Usuario

Caso de Uso	**Tramitar pedido**
Resumen de la funcionalidad	Se encargará de tramitar el pedido.
Actores	Usuario y Administrador
Precondición	Haber ejecutado el caso de uso “Gestión carrito”
Poscondición	Se enviará el pedido al administrador y se añadirá el pedido a la tabla “Pedidos”
Proceso normal principal	El Usuario clicará en el botón “Tramitar pedido” desde “Mi carrito”. El sistema mostrará: PASO 1. Datos usuario: El cliente deberá elegir la opción de identificación (cliente nuevo o cliente registrado). En función de la opción elegida el sistema mostrar el formulario correspondiente. El usuario cumplimentará el formulario y clicará el boto “Siguiente”. PASO 2. Facturación y envío. El cliente deberá elegir la opción de

	facturación (particular o empresa). En función de la opción elegida el sistema mostrar el formulario correspondiente. El usuario cumplimentará el formulario y clicará el boto “Siguiente”. PASO 3. Forma de pago. El cliente deberá elegir la opción de forma de pago (transferencia, contrareenbolso o tarjeta de crédito). En función de la opción elegida el sistema mostrar el formulario correspondiente. El usuario cumplimentará el formulario y clicará el boto “Siguiente”. PASO 4. Confirmación del pedido. El sistema mostrará todos los datos referentes al pedido. El cliente podrá modificar los datos que desee y una vez son correctos clicará el botón “Aceptar”. El sistema redireccionará al cliente a una pasarela de pago (TPV virtual), donde una entidad externa llevará a cabo la gestión del pago y nos retornará si el pago a sido satisfactorio. En el caso de pago satisfactorio, el sistema guardará la información de la compra en la base de datos en la tabla “Pedidos”, y mostrará al usuario el número de pedido.
Alternativas de proceso y excepciones	Si en algunos de los formularios el formato del los datos es incorrecto se mostrará un mensaje de error en la misma página. Si en el paso 1 de autentificación, no existe ningún usuario registrado con los datos facilitados se mostrará el mensaje de error correspondiente.

Caso de Uso	**Modificar datos de usuario**
Resumen de la funcionalidad	Se encargará de modificar los datos de un usuario.
Actores	Usuario y Administrador
Precondición	Haber ejecutado el caso de uso “Iniciar sesión”
Poscondición	Se modificarán los datos del usuario.
Proceso normal principal	El Usuario clicará en el botón “Mi perfil”. El sistema mostrará los datos del usuario. El usuario podrá modificar los datos que desee y clicar el botón “Modificar”. El sistema modificará los datos de la tabla “Usuarios” y mostrará los datos de “Mi perfil” actualizados.
Alternativas de proceso y excepciones	Si el formato del los datos es incorrecto se mostrará un mensaje de error en la misma página.

Caso de Uso	**Cerrar sesión**
Resumen de la funcionalidad	Se encargará de limpiar la sesión del usuario.
Actores	Usuario y Administrador
Precondición	Haber ejecutado el caso de uso “Iniciar sesión”
Poscondición	Se cerrará la sesión del usuario
Proceso normal principal	El Usuario clicará en el botón “Cerrar sesión”. El sistema limpiará todos los datos que contiene la sesión.
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Iniciar sesión**
Resumen de la funcionalidad	Se encargará de autentificar a un usuario
Actores	Usuario y Administrador
Precondición	Ninguna
Poscondición	Si el usuario se ha validado con éxito, se redireccionará a la página correspondiente
Proceso normal principal	El Usuario introducirá el e-mail y contraseña en las cajas destinada a ello y clicará el botón “Entrar”. El sistema buscará en la tabla “Usuarios” si existe un usuario que se corresponda con los datos facilitados. Según si el rol del usuario es “Usuario” o “Administrador” se redireccionará a una página u otra.
Alternativas de proceso y excepciones	Si el formato del los datos es incorrecto se mostrará un mensaje de error en la misma página. Si no existe ningún usuario registrado con los datos facilitados se mostrará el mensaje de error correspondiente.

Caso de Uso	**Contactar con técnico**
Resumen de la funcionalidad	Se encarga de enviar una consulta por parte de un usuario al e-mail de los técnicos.
Actores	Usuario y Administrador
Precondición	El usuario se ha identificado en el sistema

Poscondición	Si el formato de los datos introducidos es correcto se envía una consulta a administración y una copia al usuario.
Proceso normal principal	El Usuario clicará en el botón "Consulta a nuestro especialistas" y será redirigido a la página correspondiente donde deberá cumplimentar los datos del formulario y clicar el botón "enviar". El sistema enviará un e-mail con los datos de la consulta al e-mail de los técnicos y una copia del mismo e-mail al usuario.
Alternativas de proceso y excepciones	Si el formato del los datos es incorrecto se mostrará un mensaje de error en la misma página.

2.2.3 Casos de uso del actor Administrador.

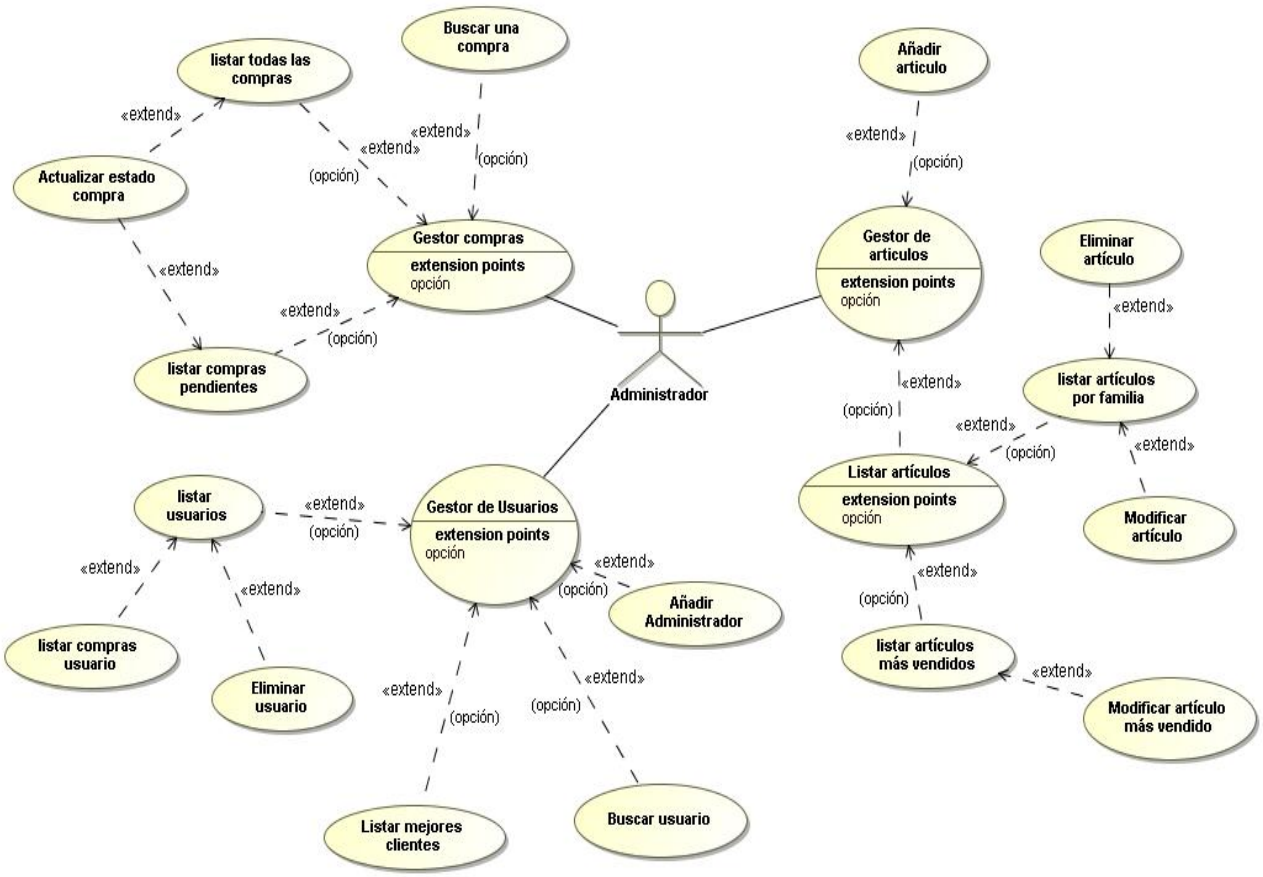


Fig. 6. Casos de uso actor Administrador

Caso de Uso	**Gestor de compras**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la interfaz del gestor de compras
Actores	Administrador
Precondición	El usuario se ha identificado en el sistema como administrador
Poscondición	Se muestra al usuario la interfaz de gestor de compras.
Proceso normal principal	El usuario clicará en el botón “Gestor de Compras”. El sistema mostrará una pantalla donde el usuario pueda ejecutar el caso de uso “Listar compras” o “Listar compras pendientes” o “Buscar una compra”
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Listar compras**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la información de todas las compras de la base de datos
Actores	Administrador
Precondición	El usuario se ha ejecutado el caso de uso “Gestor de Compras”
Poscondición	Se muestra al usuario la interfaz correspondiente
Proceso normal principal	El usuario clicará en el botón “Listar compras”. El sistema mostrará un listado con los datos de todas las compras guardadas en la tabla “compras” de la base de datos.
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Buscar una compra**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la información de una compra específica
Actores	Administrador
Precondición	El usuario se ha ejecutado el caso de uso “Gestor Compras”
Poscondición	Si los datos son correctos se muestra al información de una compra
Proceso normal	El usuario clicará en el botón “Buscar compras”. El sistema

principal	mostrará un formulario donde se pedirá al usuario que indique el número de compra. Cuando el usuario haya introducido el número de compra clicará el botón "Buscar". El sistema mostrará los datos de la compra
Alternativas de proceso y excepciones	Si el número de compra introducido no corresponde con ninguna compra de la base de datos se mostrará el mensaje de error correspondiente

Caso de Uso	**Listar compras pendientes**
sResumen de la funcionalidad	Se encarga de mostrar al usuario un listado con las compras pendientes de gestionar
Actores	Administrador
Precondición	El usuario se ha ejecutado el caso de uso "Gestor Compras"
Poscondición	Se mostrará al usuario un listado con las compras pendientes de gestionar
Proceso normal principal	El usuario clicará en el botón "Listar compras pendientes". El sistema mostrará un listado con los datos de todas las compras guardadas en la tabla "compras" de la base de datos pendientes de gestionar
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Actualizar estado compra**
sResumen de la funcionalidad	Se encarga de modificar el estado de una compra
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso "Listar compras pendientes" o "Listar compras"
Poscondición	Se modifica el estado de una compra
Proceso normal principal	El usuario clicará en el botón "Actualizar estado" en la compra que desea actualizar. El sistema mostrará un formulario donde el usuario deberá indicar el nuevo estado. Una vez introducido el nuevo estado el usuario clicará el botón "Actualizar". El sistema modifica el registro de la tabla "compras" de la base de datos
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Gestor de usuarios**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la interfaz del gestor de usuarios
Actores	Administrador
Precondición	El usuario se ha identificado en el sistema como administrador
Poscondición	Se muestra al usuario la interfaz de gestor de usuarios
Proceso normal principal	El usuario clicará en el botón “Gestor de Usuarios”. El sistema mostrará una pantalla donde el usuario pueda ejecutar el caso de uso “añadir Administrador” o “Listar usuarios” o “Listar Mejores Clientes” o “Buscar usuarios”.
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Añadir administrador**
Resumen de la funcionalidad	Se encarga de añadir un nuevo administrador
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Gestor de Usuarios”
Poscondición	Si los datos son correctos se añadirá un administrador a la tabla “usuarios” de la base de datos
Proceso normal principal	El usuario clicará en el botón “Añadir Administrador”. El sistema mostrará un formulario con los datos que identifican un administrador. El usuario introducirá los dantos y clicará el botón “Añadir”. El sistema añadirá un administrador a la tabla “usuarios” de la base de datos
Alternativas de proceso y excepciones	Si ya existe un usuario con e-mail indicado se mostrará un mensaje de error.

Caso de Uso	**Listar mejores clientes**
Resumen de la funcionalidad	Se encarga de listar los usuarios de la tabla “usuarios” de la base de datos ordenados por el importe de las compras realizadas.
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Gestor de usuarios”
Poscondición	Se mostrará un listado de todos los usuarios ordenados por el

	importe de las compras realizadas
Proceso normal principal	El usuario clicará el botón “Listar mejores clientes”. El sistema mostrará listado de todos los usuarios de la tabla “usuarios” de la base de datos ordenados por el importe de las compras realizadas.
Alternativas de proceso y excepciones	

Caso de Uso	**Buscar un usuario**
Resumen de la funcionalidad	Se encarga de buscar un usuario de la tabla “usuarios” de la base de datos
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Gestor de usuarios”
Poscondición	Si los datos introducidos son correctos se mostrará los datos un usuario
Proceso normal principal	El usuario clicará el botón “Buscar usuario”. El sistema mostrará un formulario donde se pedirá el e-mail del usuario a buscar. El usuario introducirá el e-mail y clicará el botón “Buscar”. El sistema mostrará los datos del usuario indicado.
Alternativas de proceso y excepciones	Si el e-mail indicado no se corresponde con ningún e-mail de ningún usuario se indicará el mensaje de error correspondiente.

Caso de Uso	**Listar usuarios**
Resumen de la funcionalidad	Se encarga de listar los usuarios de la tabla “usuarios” de la base de datos
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Gestor de usuarios”
Poscondición	Se mostrará un listado de todos los usuarios
Proceso normal principal	El usuario clicará el botón “Listar usuarios”. El sistema mostrará listado de todos los usuarios de la tabla “usuarios” de la base de datos
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Eliminar un usuario**
Resumen de la funcionalidad	Se encarga de eliminar un usuario de la tabla “usuarios” de la base de datos
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Listar Usuarios”
Poscondición	Si los datos introducidos son correctos se eliminará un usuario de la base de datos
Proceso normal principal	El usuario clicará el botón “Elimina usuario” del usuario que desea eliminar. El sistema cambiará el estado del usuario a “baja” de la tabla “usuarios” de la base de datos.
Alternativas de proceso y excepciones	Si el usuario a eliminar es un administrador y este es el último administrador de la base de datos se mostrará el mensaje de error correspondiente.

Caso de Uso	**Listar compras usuario**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la información de las compras que ha realizado un usuario concreto
Actores	Administrador
Precondición	El usuario se ha ejecutado el caso de uso “Listar Usuarios”
Poscondición	Si los datos son correctos se muestra al información de las compras que ha realizado un usuario
Proceso normal principal	El usuario clicará en el botón “Listar compras usuario” del usuario el cual quiere ver sus compras. El sistema mostrará los datos de las compras que ha realizado el usuario deseado.
Alternativas de proceso y excepciones	Si el usuario no ha realizado ninguna compra no se mostrarán ningún dato.

Caso de Uso	**Gestor de artículos**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la interfaz del gestor de artículos
Actores	Administrador
Precondición	El usuario se ha identificado en el sistema como administrador
Poscondición	Se muestra al usuario la interfaz de gestor de artículos

Proceso normal principal	El usuario clicará en el botón “Gestor Artículos”. El sistema mostrará una pantalla donde el usuario pueda ejecutar el caso de uso “Listar Artículos” o “Añadir artículo”
Alternativas de proceso y excepciones	Ninguna

aso de Uso	**Añadir artículo**
Resumen de la funcionalidad	Se encarga de añadir un artículo a la base de datos
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Gestor de Artículos”
Poscondición	Si el formato y datos son correctos se añade un artículo a la base de datos
Proceso normal principal	El usuario clicará en el botón “Añadir artículo”. El sistema mostrará un formulario con el código de artículo (variable que identifica un artículo) y el resto de información que definen un artículo. Cuando se ha cumplimentado el formulario el usuario clicará el botón “Añadir”. El sistema añadirá el artículo a la base de datos.
Alternativas de proceso y excepciones	Si ya existe un artículo con el código de artículo indicado, se indicará un mensaje de error.

Caso de Uso	**Listar artículos**
Resumen de la funcionalidad	Se encarga de mostrar al usuario la interfaz del listado de artículos
Actores	Administrador
Precondición	El usuario se ha ejecutado el caso de uso “Gestor de Articulos”
Poscondición	Se muestra al usuario el listado seleccionado
Proceso normal principal	El sistema mostrará los tipos de opciones que dispone la aplicación de listado de artículos. Para cada tipo el sistema mostrará un botón y se ejecutará el caso de uso “Listar artículos por familia” o “Listar artículos más vendidos”.
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Listar artículos por familia**
Resumen de la funcionalidad	Se encarga de mostrar al usuario un listado de los productos de una familia
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Listar Artículos”
Poscondición	Se muestra al usuario el listado de los productos de la familia seleccionada
Proceso normal principal	El sistema mostrará botones para cada una de las familias de artículos que existen en la base de datos. El usuario clicará en el botón de una de las familias. El sistema mostrará un listado con todos los productos de la familia seleccionada
Alternativas de proceso y excepciones	Si la familia seleccionada no dispone de ningún producto se mostrará el mensaje correspondiente

Caso de Uso	**Listar artículos más vendidos**
Resumen de la funcionalidad	Se encarga de mostrar al usuario un listado de los productos de una familia
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Listar Artículos”
Poscondición	Se muestra al usuario el listado de los productos más vendidos
Proceso normal principal	El usuario clicará en el botón “Más vendidos”. El sistema mostrará un listado con los productos más vendidos.
Alternativas de proceso y excepciones	Si no hay ningún producto en la categoría más vendidos se mostrará el mensaje correspondiente

Caso de Uso	**Modificar artículo**
Resumen de la funcionalidad	Se encarga de modificar un artículo ya existente en la base de datos
Actores	Administrador

Precondición	El usuario ha ejecutado el caso de uso “listar artículos por familia”
Poscondición	Si el formato y datos son correctos se modifica un artículo de la base de datos
Proceso normal principal	El usuario clicará en el botón “Modificar” situado al lado del nombre del artículo dentro del listado “Listar artículos por familia”. El sistema mostrará un formulario con los datos actuales del artículo seleccionado. Cuando se ha modificado los datos deseados el usuario clicará el botón “Modificar”. El sistema modificará el artículo de la base de datos.
Alternativas de proceso y excepciones	Si ya existe un artículo con el código de artículo indicado, se indicará un mensaje de error.

Caso de Uso	**Eliminar artículo**
Resumen de la funcionalidad	Se encarga de eliminar un artículo de la base de datos
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “listar artículos por familia”
Poscondición	Se eliminará un artículo de la base de datos.
Proceso normal principal	El usuario clicará en el botón “Eliminar artículo” ” situado al lado del nombre del artículo dentro del listado “Listar artículos por familia”. El cambiará el estado del artículo seleccionada a “baja”.
Alternativas de proceso y excepciones	Ninguna

Caso de Uso	**Modificar artículo más vendidos**
Resumen de la funcionalidad	Se encarga de añadir un artículo a la tabla “masVendidos” de la base de datos
Actores	Administrador
Precondición	El usuario ha ejecutado el caso de uso “Listar más vendidos”
Poscondición	Si el formato y datos son correctos se añadirá un artículo de la tabla “masVendidos” de la base de datos
Proceso normal principal	El usuario modificará el código del producto del listado, que desea remplazar, indicado el código del producto que quiere incorporar a la tabla “masVendidos” de la base de datos. Una vez modificado

	clicará el botón “Modificar”. El sistema modificará el contenido de la tabla “masVendidos” de la base de datos.
Alternativas de proceso y excepciones	Si no existe ningún artículo con el código de artículo introducido se indicará un mensaje de error.

2.2.4. Casos de uso de toda la aplicación.

El siguiente diagrama de de casos de uso corresponde con todos los casos de usos de la aplicación:

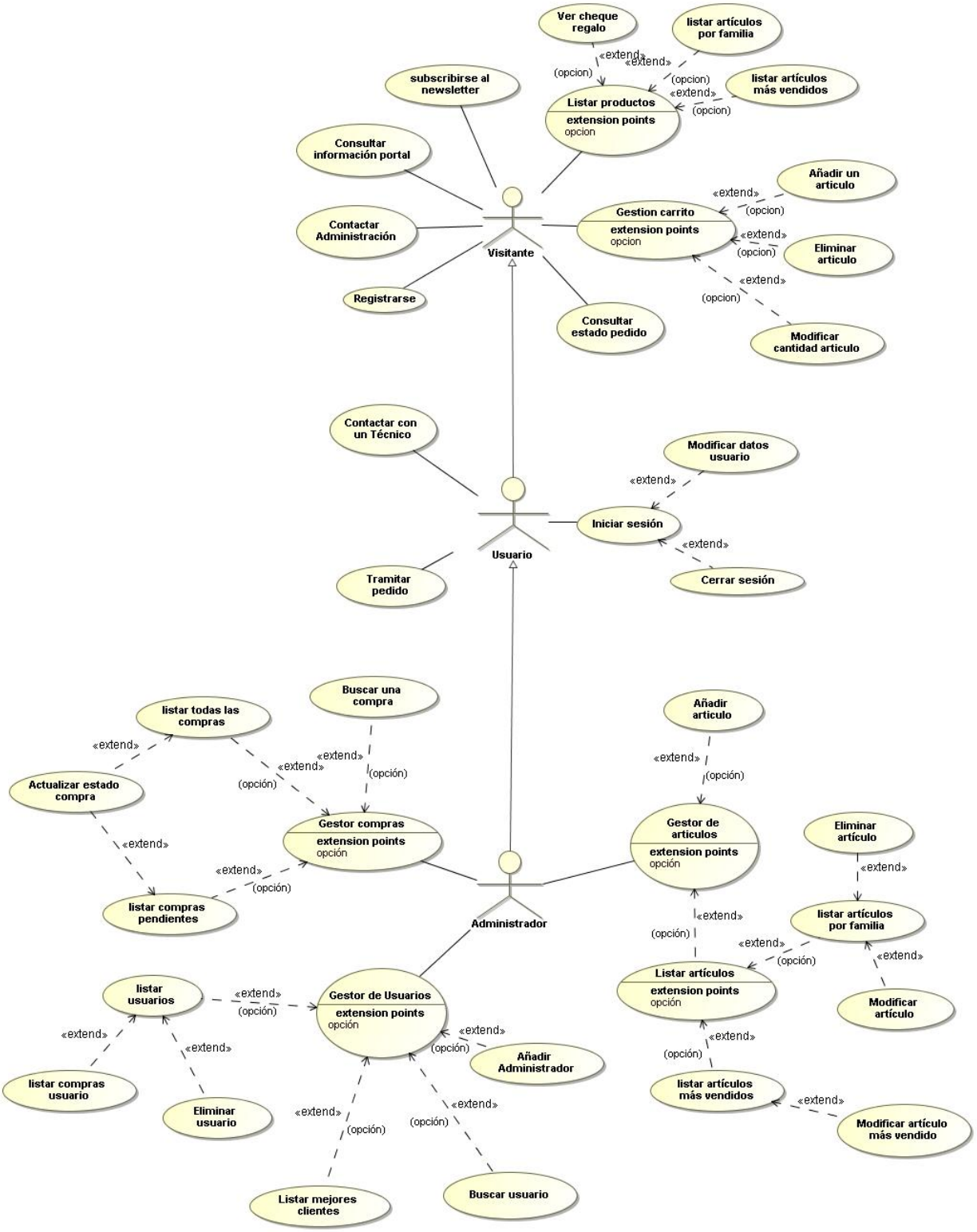


Fig. 7. Casos de uso actor de la aplicación

3. Diseño técnico

Una vez realizado el análisis funcional del proyecto ya conocemos que debe hacer nuestra aplicación. Por lo tanto ahora deberemos definir el diseño técnico necesario para poder llevar a cabo la funcionabilidad de la aplicación.

Este diseño técnico estará formado por la definición de la arquitectura de la aplicación (IDE, frameworks, SGBD, bibliotecas,….) y de los diagramas necesarios (de clases o de Entidad-Relación)

3.1. Arquitectura

El proyecto que se presenta, como ya hemos definido, se trata de una aplicación web, donde los usuarios podrán acceder desde cualquier lugar con una conexión Internet. El esquema general de la aplicación será:

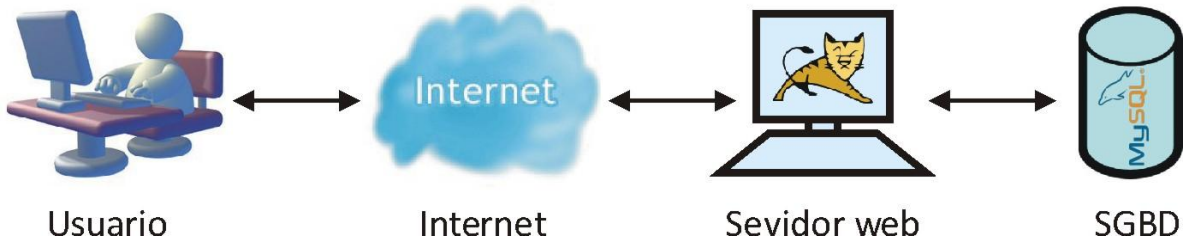


Fig. 8. Esquema general de aplicación web

- **Usuario:** Será la persona que utilice el sistema. Tal y como hemos comentado en el punto anterior podrá ser de 3 tipos: visitante, usuario registrado y administrador. Cada uno de ellos podrá tener acceso a partes de la aplicación específicas.

- **Servidor web:** En el servidor de aplicaciones web es donde se guardará la aplicación para que pueda ser utilizada a través de una conexión de internet. En nuestro caso utilizaremos el servidor de aplicaciones Apache Tomcat.

  Apache Tomcat funciona como un contenedor de servlets desarrollado por Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Tomcat puede funcionar como servidor web por sí mismo, pero que habitualmente se integra

con el propio servidor Apache. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. [2].

- **Sistema gestor de base de datos (SGBD):** Su función será la de almacenar los datos de nuestra aplicación. En nuestro caso utilizaremos como sistema gestor de base de datos MySQL.

  MySQL es un motor de base de datos relacional, multihilo y multiusuario que permite soportar una gran carga de forma muy eficiente, ampliamente utilizado no sólo en proyectos de software libre sino también en multitud de desarrollos comerciales ya que dispone de dos versiones, una comercial y otra libre con licencia GPL.
  [3]

  Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. [4]

Como ya hemos comentado para el desarrollo de la aplicación utilizaremos la tecnología J2EE (Java Enterprise Edition), que es una plataforma Java para el desarrollo de aplicaciones empresariales. Java EE ofrece un framework para el desarrollo de aplicaciones distribuidas multicapa basadas en Web [5]. Es por tanto que he optado por implementar el patrón arquitectónico Modelo Vista Controlador (MVC).

3.1.1. Patrón MVC

Un patrón arquitectónico es un patrón de alto nivel que fija la arquitectura global de una aplicación [6]. De esta forma la aplicación empresarial será más mantenible y podrá contener partes reutilizables, que es uno de los objetivos de la programación orientada a objetos, en la cual se basa la tecnología J2EE. En este caso, el patrón MVC realiza una separación clara entre el modelo (lógica del negocio), la vista (interfaz gráfica) y el controlador:

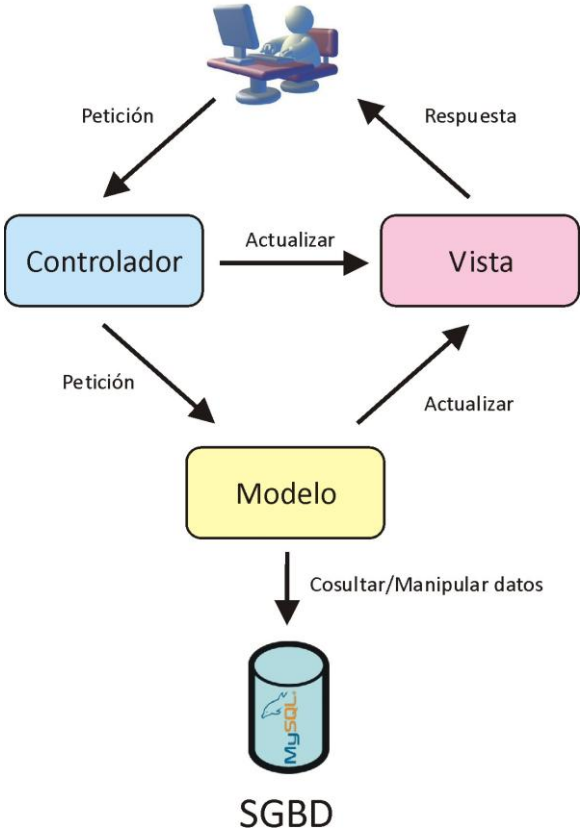


Fig. 9. Esquema patrón MVC

- La **vista** define la interficie del usuario. En la vista, un usuario hace una request (pulsa un enlace, envía un formulario…), nuestra aplicación procesa esa petición y envía de vuelta al navegador los datos necesarios para generar la respuesta, de nuevo en la vista.

- El **modelo** sería el componente encargado del acceso al SGBD (Sistema Gestor de Base de Datos) y de la lógica de negocio (la funcionalidad propia de la aplicación, el procesamiento de los datos).

- El **controlador** se encarga de recibir la petición desde la vista, acceder al modelo y/o actualizarlo y devolver el control a la vista para desplegar la interfaz de usuario.

Tal y como se puede observar en el siguiente gráfico, J2EE ofrece soluciones para cada una de las vistas:

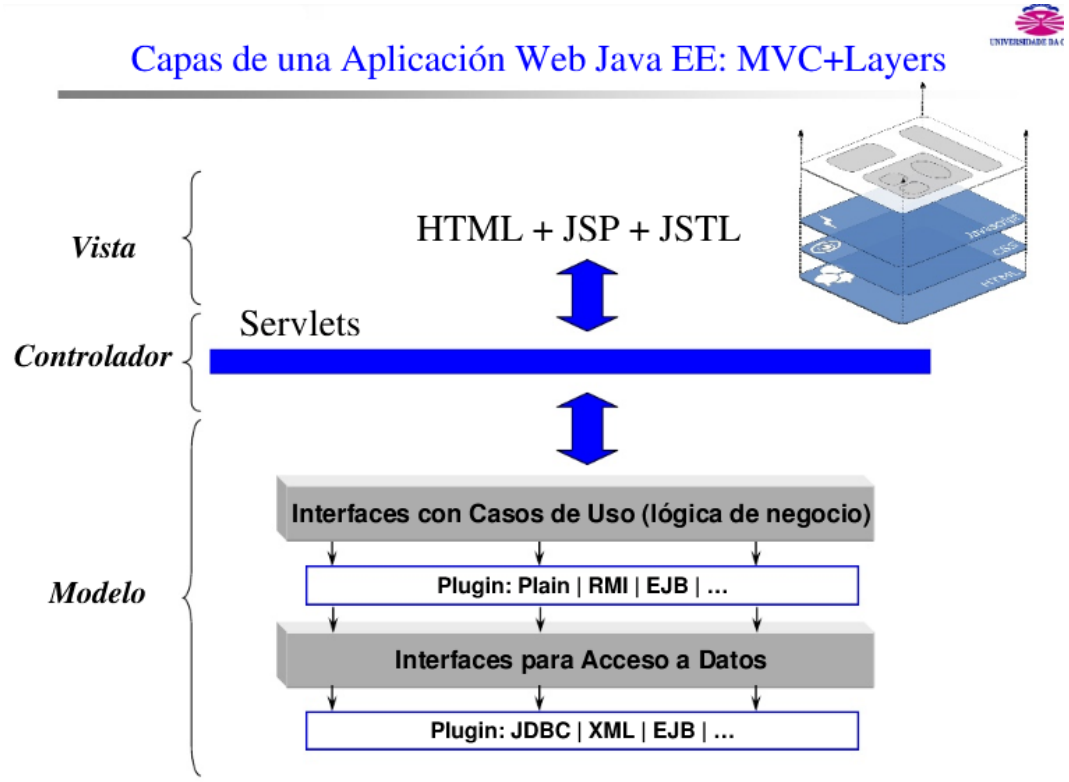


Fig. 10. Relación J2EE y MVC [6]

La utilización del patrón arquitectónico MVC y los Patrones de Diseño J2EE, dotarán a nuestras aplicaciones de gran escalabilidad facilitando posibles futuras ampliaciones, migraciones y cambios en general. [7]

A continuación se definirán los diferentes frameworks o bibliotecas que se utilizan en la aplicación web para cada una de las capas.

3.1.2. Capa Vista

3.1.2.1 JSP

Para generar la vista de la aplicación se utilizarán principalmente páginas JSP. La tecnología JSP se crea especialmente para producir contenido dinámico. Una página JSP es un documento que contiene una plantilla de texto fija, junto a unas etiquetas específicas para incluir texto dinámico o ejecutar la lógica del negocio. Puesto que es un lenguaje de servidor, una página JSP es siempre servida al cliente como código HTML tradicional. Las páginas JSP no deben utilizarse como controladores, ya que con

esta funcionalidad se obtiene una mezcla de etiquetas y código que las hace difícil de leer y mantener. Y es por esta razón que utilizamos el patrón MVC. [7]

La tecnología JSP es compatible con una amplia gama de herramientas de diseño ya que al igual que HTML se basa en un lenguaje de etiquetas. Las etiquetas de una página JSP pueden ser de tres tipos: directivas, script, o etiquetas personalizadas. Las directivas son evaluadas en tiempo de compilación y se identifican por los delimitadores <%@ y %>. Los scripts son bloques de código Java embebido en la página JSP entre los delimitadores <% y %> que generan contenido dinámico que se incluye en la respuesta cuando la página es servida. [7]

En estas páginas JSP utilizaremos algunos componentes para la capa de vista que proporciona struts 2, un framework que veremos más adelante, en la Capa Controlador.

3.1.2.2. Biblioteca Jquery.

jQuery es una biblioteca de JavaScript, de software libre y de código abierto, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. [8]

En este proyecto se utilizará la biblioteca Jquery para la creación de algunos efectos de animación en la capa vista, para mejorar la navegabilidad de la aplicación.

3.1.3 Capa Controlador

3.1.3.1 Struts 2

Struts 2 es un framework para el desarrollo de aplicaciones web realizadas con java EE que tiene como objetivo facilitar y agilizar la implementación de las mismas. Dentro del patrón de diseño MV, Struts 2 implementa el controlador, y además proporciona algunos componentes para la capa de vista. [9]

Componentes.

Struts 2 está formado por varios componentes, siendo el principal un filtro, conocido como el **FilterDispatcher**. A través de este filtro se lanza la ejecución de todas las peticiones relacionadas con el framework. Las principales responsabilidades del FilterDispatcher son:

- Ejecutar los Actions, que son los manejadores de las peticiones.
- Comenzar la ejecución de la cadena de interceptores.
- Limpiar el "ActionContext", para evitar fugas de memoria.

Struts 2 procesa las peticiones usando tres componentes principales:

- **Interceptores:** Estos permiten que se implementen funcionalidades que afectan a uno o para varios Actions, pero que se ejecuten fuera del Action (por ejemplo validaciones de datos, conversiones de tipos, etc.). Los interceptores realizan tareas antes y después de la ejecución de un Action e incluso pueden evitar que un Action se ejecute (por ejemplo si estamos haciendo alguna validación que no se ha cumplido).
- **Acciones:** Las acciones o Actions son clases encargadas de realizar la lógica para servir una petición. Cada URL es mapeada a una acción específica, la cual esta implementada para poder servir a la petición.
- **Resultados:** Después que un Action ha sido procesado se debe enviar la respuesta de regreso al usuario, esto se realiza usando Results. Este resultado determinará qué es lo que se debe hacer. Por ejemplo un tipo de Result puede enviar al usuario de vuelta una JSP, otro puede redirigirlo hacia otro sitio, e incluso otro puede enviarlo a otro action.

Funcionamiento.

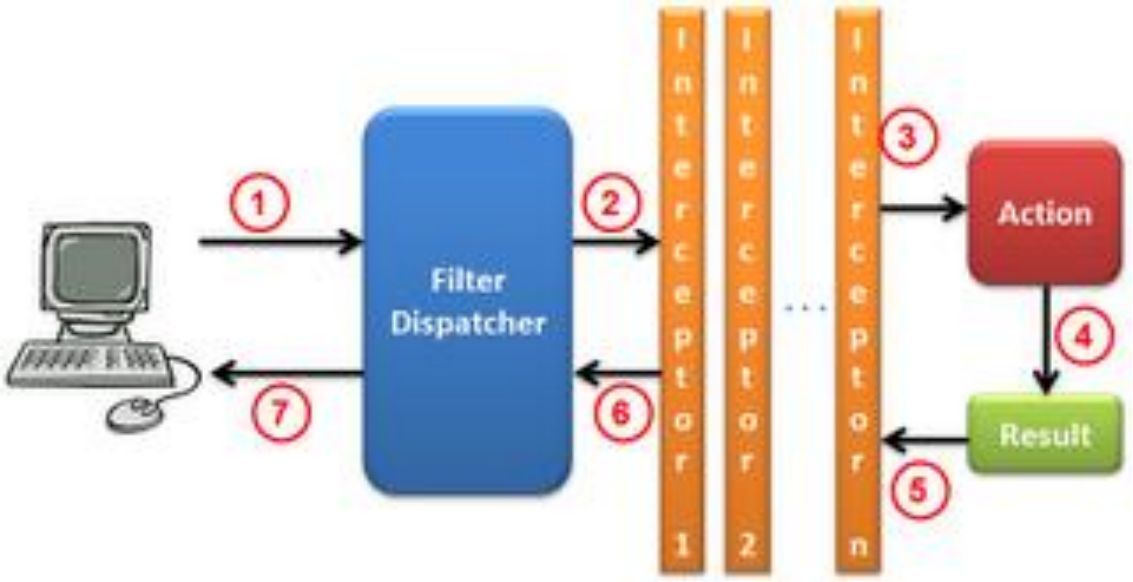


Fig. 11. Esquema funcionamiento struts2 [10]

Los pasos que sigue una petición son:

1. El navegador web hace una petición para un recurso de la aplicación. El filtro de Struts (FilterDispatcher) revisa la petición y determina el Action apropiado para servirla.
2. Se aplican los interceptores, los cuales realizan algunas funciones como validaciones, flujos de trabajo, manejo de la subida de archivos, etc.
3. Se ejecuta el método adecuado del Action (por default el método "execute"), este método usualmente almacena y/o regresa alguna información referente al proceso.
4. El Action indica cuál result debe ser aplicado. El result genera la salida apropiada dependiendo del resultado del proceso.
5. Se aplican al resultado los mismos interceptores que se aplicaron a la petición, pero en orden inverso.

6. El resultado vuelve a pasar por el FilterDispatcher aunque este ya no hace ningún proceso sobre el resultado (por definición de la especificación de Servlets, si una petición pasa por un filtro, su respuesta asociada pasa también por el mismo filtro).

7. El resultado es enviado al usuario y este lo visualiza.

3.1.3.2 JavaMail.

El API JavaMail [11] es un paquete opcional para leer, componer y enviar mensajes electrónicos. Se usa este paquete para crear programas similares a Eudora y Microsoft Outlook, por ejemplo. Su propósito principal es leer y escribir *e-mails* e interaccionar con los programas que se encargan del envío de estos mensajes usando el lenguaje de programación Java. [12]

En nuestro caso, utilizaremos este API solo para enviar mensajes de forma automática desde la aplicación, tanto a los usurarios como administradores. Para ello utilizaremos el servidor SMPT de Gmail. [13]

3.1.3 Capa Modelo

3.1.3.1 Data Access Object (DAO)

La mayoría de las aplicaciones, tienen que persistir datos en algún momento, y nuestra aplicación no es una excepción. Para ello, nuestra aplicación deberá interactuar con una base de datos. Crearemos una capa de persistencia dentro de la capa Modelo, que será la encargada de interactuar con la base de datos. Habiendo explicado esto, podemos decir que DAO es un patrón de diseño utilizado para crear esta capa de persistencia. [14]

El patrón de diseño DAO nos permitirá poder cambiar, por ejemplo el motor de la base de datos (por ejemplo de MySQL a PostgreSQL), modificando solo la capa de persistencia, sin tener que modificar nada de la lógica del negocio.

Como dijimos antes, DAO encapsula el acceso a la base de datos. Cuando la capa de lógica de negocio necesite interactuar con la base de datos, va a hacerlo a través de la API que le ofrece DAO. Generalmente esta API consiste en creación de métodos CRUD (Create, Read, Update y Delete). Por ejemplo cuando la capa de lógica de negocio necesite guardar un dato en la base de datos, va a llamar a un método create(), sin tener en cuenta lo que haga este método, esto es la tarea de DAO y depende de cómo DAO implemente el método create(), que puede que lo implemente de manera que los

datos se almacenen en una base de datos relacional como puede que lo implemente de manera que los datos se almacenen en ficheros de texto. Lo importante es que la capa de lógica de negocio no tiene porque saberlo, lo único que sabe es que el método create() va a guardar los datos, así como el método delete() va a eliminarlos, el método update() actualizarlos, etc. [14]

En una aplicación, podemos tener tantos DAOs como modelos (1 DAO por tabla) o utilizar un solo DAO para todos los modelos (1 DAO para todas las tablas de la BD), tal y como he hecho yo en este proyecto. Todo dependerá del tamaño de la aplicación.

Este patrón es muy utilizado en el contexto JDBC, que será el que usemos en nuestra aplicación. La capa Modelo tendrá una estructura del tipo:

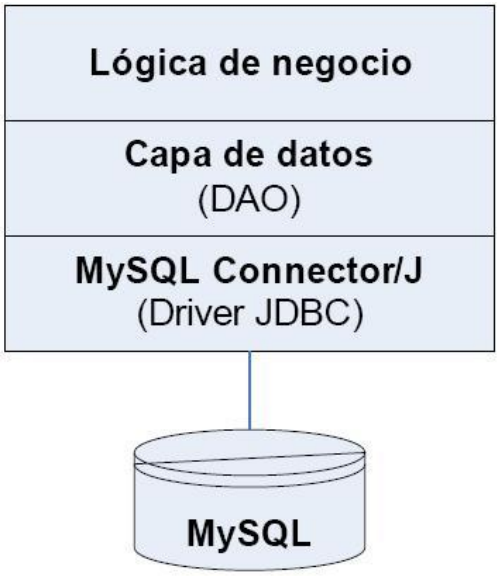


Fig. 12. Estructura capa Modelo utilizando DAO [15]

El controlador JDBC ofrece la conexión a la base de datos e implementa el protocolo para la transferencia de las consultas y resultados entre el cliente (aplicación) y la base de datos. Será pues el objeto DAO el que maneja la conexión con la fuente de datos para obtener y modificar datos implementando los mecanismos de acceso requeridos para trabajar con la fuente de datos seleccionada.

Para acceder a datos resistentes en bases de datos relacionales, el API JDBC, que pertenece a la librería java: java.sql.*, proporciona acceso y manipulación de datos relacional utilizando sentencias SQL. [16]

Estructura DAO

A continuación se representa la estructura que tiene el patrón de diseño DAO:

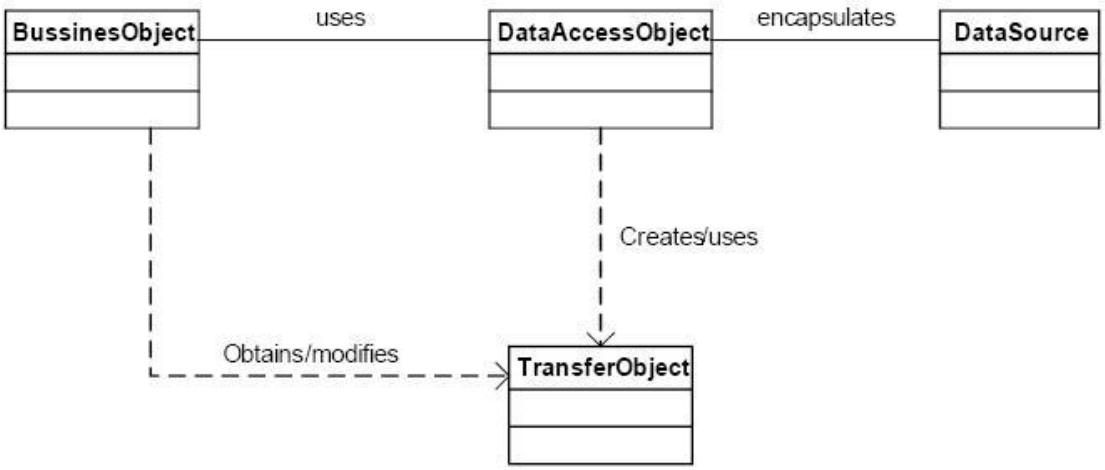


Fig. 13. Estructura patrón de diseño DAO [17]

donde:

- **BuisnessObject:** Este objeto representa el que quiere acceder a la fuente de datos para poder almacenar o consultar datos.
- **DataAccesObject:** Es el objeto que maneja la conexión con la fuente de datos para obtener y modificar datos, abstrayendo los detalles de la gestión a BussinesObject.
- **DataSource:** Representa la implementación de la fuente de datos. En nuestro caso tendremos una base de datos relacional con MySQL.
- **TransferObject:** Es un objeto intermedio entre BuisnessObject y el DataAcessObject, y encapsulará los datos referentes a una tabla de la base de datos. Permitirá manejar los resultados y entradas en la base de datos.

3.1.5 IDE: Netbeans

Para finalizar, decir que toda la aplicación la vamos a desarrollar bajo en entorno de desarrollo integrado NetBeans.

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Es un producto libre y gratuito sin restricciones de uso.

El proyecto NetBeans consiste en un IDE de código abierto y una plataforma que permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software [18]

3.2. Diagrama de clases

A continuación se presenta el diagrama de clases de la aplicación:

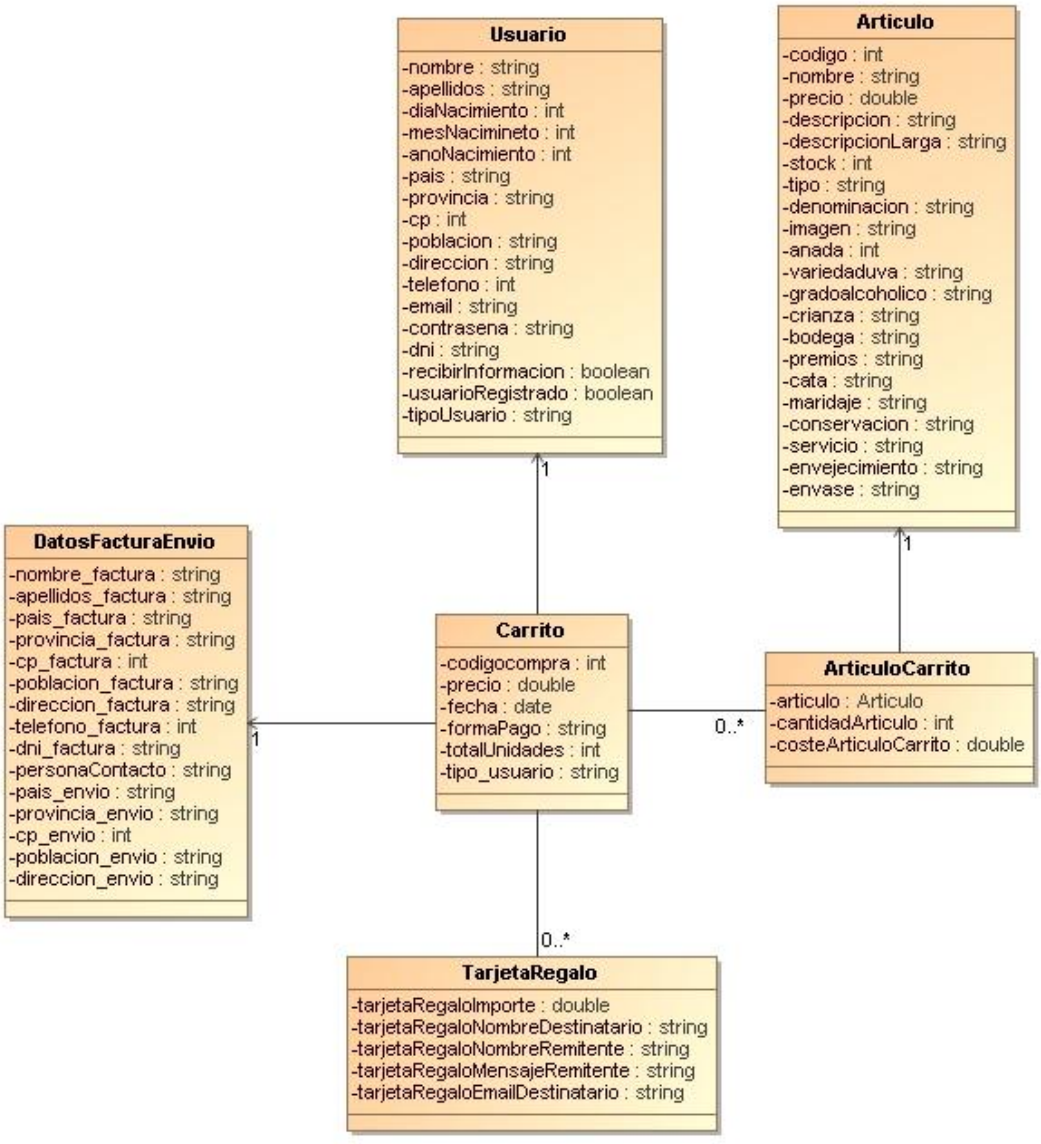


Fig. 14. Diagrama de Clases.

3.3. Diseño de la base de Datos. Model E/R

En este apartado presentaremos como el diseño de la base de datos que utilizamos en nuestra aplicación.

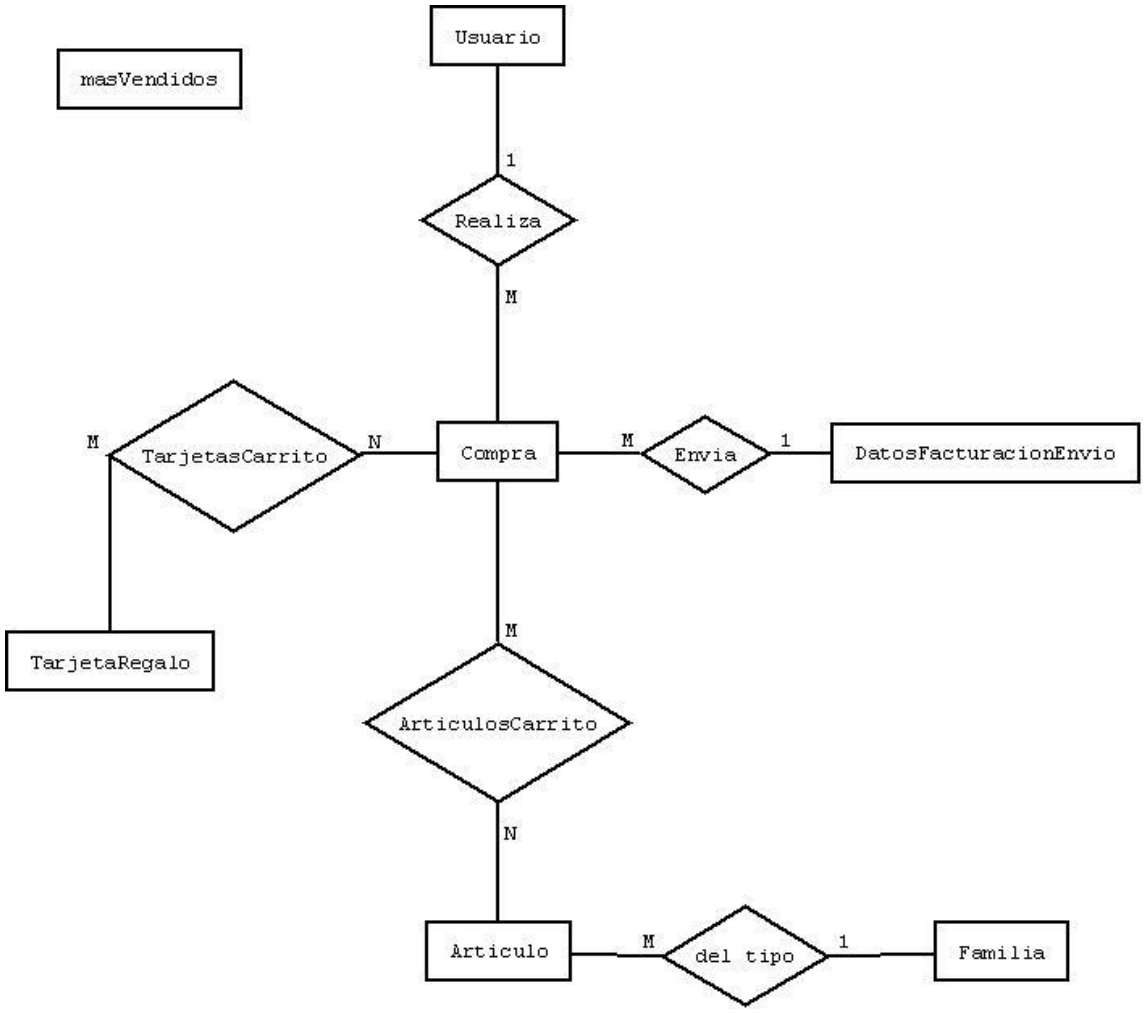


Fig. 15. Diagrama de E/R.

Transformación al modelo relacional (claves primarias subrayadas)

Usuario

email, nombre, apellidos, diaNacimiento, mesNacimiento, anoNacimiento, país, provincia, cp, población, dirección, teléfono, contraseña, dni, recibirInformacion, usuarioRegistrado, tipoUsuario.

Articulo

codigo, nombre, precio, descripción, descripcionLarga, stock, tipo, denominación, imagen, anada, variedaduva, gradoalcoholico, crianza, bodega, premios, cata, maridaje, conservación, servicio, envejecimiento, envase.
{ tipo } clave foránea hacia Familia (tipo)

DatosFacturaEnvio

codigoCompra, nombre_factura, apellidos_factura, pais_factura, provincia_factura, cp_factura, poblacion_factura, direccion_factura, telefono_factura, dni_factura, personaContacto_envio, pais_envio, provincia_envio, cp_envio, poblacion_envio, direccion_envio.

TarjetaRegalo

tarjetaRegaloCodigo, tarjetaRegaloImporte, tarjetaRegaloNombreDestinatario, tarjetaRegaloEmailDestinatario, tarjetaRegaloNombreRemitente, tarjetaRegaloMensajeRemitente.

MasVendidos

codigo

Compra

codigoCompra, precio, fecha, formaPago, totalUnidades, tipo_usuario, emailUsuario.
{ emailUsuario } clave foránea hacia Usuario (email)
{ codigoCompra } clave foránea hacia DatosFacturaEnvio (codigoCompra)

ArticulosCarrito

codigoCompra, codigoArticulo, cantidad.
{ codigoArticulo } clave foránea hacia Articulo (codigo)
{ codigoCompra } clave foránea hacia Compra (codigoCompra)

TarjetasCarrito

codigoCompra, tarjetaRegaloCodigo.
{tarjetaRegaloCodigo} clave foránea hacia TarjetaRegalo (tarjetaRegaloCodigo)
{ codigoCompra } clave foránea hacia Compra (codigoCompra)

3.4. Diseño interficie gráfica

En este apartado se presenta la interficie gráfica de la aplicación. El objetivo de este apartado no es el de definir con todo detalle cómo es la interficie, sino indicar a rasgos generales las partes que forman esta interficie.

La estructura general de la interficie gráfica de la aplicación es la siguiente:

Fig. 16. Estructura general de la Interficie gráfica.

Donde podemos diferenciar 4 zonas diferentes:

- **Imagen general:** En primer lugar tenemos una imagen con el logotipo de la aplicación web y una textura de fondo, que ocupará parte de la interficie y que estará siempre visible.

- **Menú:** Los menús de la aplicación también estarán siempre visibles, serán estáticos e iguales para todas las páginas jsp a las que puede acceder un visitante o usuario registrado. Por otro lado tendremos menús diferentes pero también estáticos para todas las páginas jsp a las que puede acceder un administrador. Tendremos tres zonas dedicadas a estos menús que permitirán interactuar con la aplicación (ver listados, carrito de la compra, etc…)

- **Pie de página.** Esta zona también será estática y estándar para todas las páginas jsp, y contendrá entre otra información legal, de contacto, de condiciones de compra, etc...

- **Presentación de la información.** Esta es la zona donde la aplicación presentará todos los resultados de las acciones requeridas por el usuario (sea del tipo que sea). Esta, es pues, la zona donde generaremos html dinámico.

Por ejemplo, si un usuario clica sobre el link “registrarse” la aplicación web presentará una interficie gráfica tipo a:

Fig. 17. Estructura página registro

Otro ejemplo sería la presentación del listado productos de una familia, la cual tendrá una interficie gráfica del tipo:

Fig. 18. Estructura página listado de productos

Tal y como he indicado el actor administrador puede hacer tareas de administración de los datos de la aplicación (gestión de artículos, usuarios, compras,…). Para poder realizar estas tareas se ha creado un menú de exclusivo acceso para los administradores. Desde este menú se podrán realizar todas las tareas de administración. En la siguiente figura se muestra el contenido de este menú:

Fig. 19. Estructura página zona administración

Como podemos ver, desde este menú podemos ejecutar todas las funcionalidades definidas en los casos de usos para el actor administrador.

4. Implementación.

Una vez definido el análisis funcional y el diseño de la arquitectura ya sabemos que tiene que hacer la aplicación y como lo tiene que hacer. Por tanto, deberemos diseñar el código necesario para que la implementación cumpla con los requisitos establecidos.

En este apartado definiremos los requisitos de software para poder ejecutar la aplicación. También definiremos las decisiones de diseño e implementación que se han llevado a cabo durante la implementación de la aplicación. Un aspecto a considerar es el hecho de que la aplicación está configurada para trabajar con el contexto vacío.

4.1. Requisitos de software.

El desarrollo de la aplicación se ha realizado bajo el siguiente software:

- **Sistema Operativo: Windows 7 home edition**
- Diagramas de casos de uso, de secuencia, de clase, etc…: **MagicDraw UML.**
- Prototipos y retoque de imágenes: **Adobe Photoshop y Illustrator**.

- Entorno de desarrollo integrado: **Netbeans IDE 7.2.1**
- Gestor de bases de datos: **MySQL** a través del paquete **XAMPP Control Panel v3.0.12**.
- Servidor web: **Apache Tomcat v7.0.22**

- Frameworks: **Struts2 v2.3.4**
  Javamail v1.4.5

- Para las pruebas de la aplicación se han utilizado diferentes navegadores: **Mozilla Firefox, Google Chrome , Opera** e **Internet Explorer 9.**

4.2. Preparación del entorno de trabajo

Para la poder ejecutar la aplicación correctamente se deberán seguir los pasos que se describen en el fichero "Instrucciones de Instalacion.pdf" que se facilita con el proyecto. Siguiendo los pasos de forma correcta se podrá ejecutar la aplicación teniendo ya unos datos para poder interactuar. Con el proyecto se entrega dos scripts, uno para crear las tablas de la bases de datos (Creacion_BD.sql) y otro para introducir algunos datos de ejemplo como usuarios, artículos, compras,… (Inserccion_datos.sql). Con estos datos crearemos los siguientes usuarios:

Usuario -> e-mail: **rafa@gmail.com** contraseña: **usuario**
Usuario -> e-mail: **juan@gmail.com** contraseña: **usuario**
Administrador -> e-mail: **administrador@solovinoecologico.com** contraseña: **admin1**

4.3. Librerías.

Este proyecto utiliza dos librerías: librerías de Struts2 y JavaMail.

Referente a struts2, no es necesario instalar todas las librerías de Struts2. En el proyecto hemos utilizado las siguientes librerías, que se encuentran dentro de la carpeta \lib:

- asm-3.3.jar
- asm-commons-3.3.jar
- commons-fileupload-1.2.2.jar
- commons-io-2.0.1.jar
- commons-lang-2.4.jar
- freemarker-2.3.19.jar
- javassist-3.11.0.GA.jar
- ognl-3.0.5.jar
- struts2-core-2.2.4.jar
- xwork-core-2.2.4.jar
- struts2-convention-plugin-2.2.4.jar

Referente a la libreríua de JavaMail, esta está formada por un único fichero (mail.jar), por lo que para instarla solo tendremos que seleccionar este fichero.

4.4. Seguridad.

La aplicación se ha desarrollado considerando algunos mecanismos de seguridad.

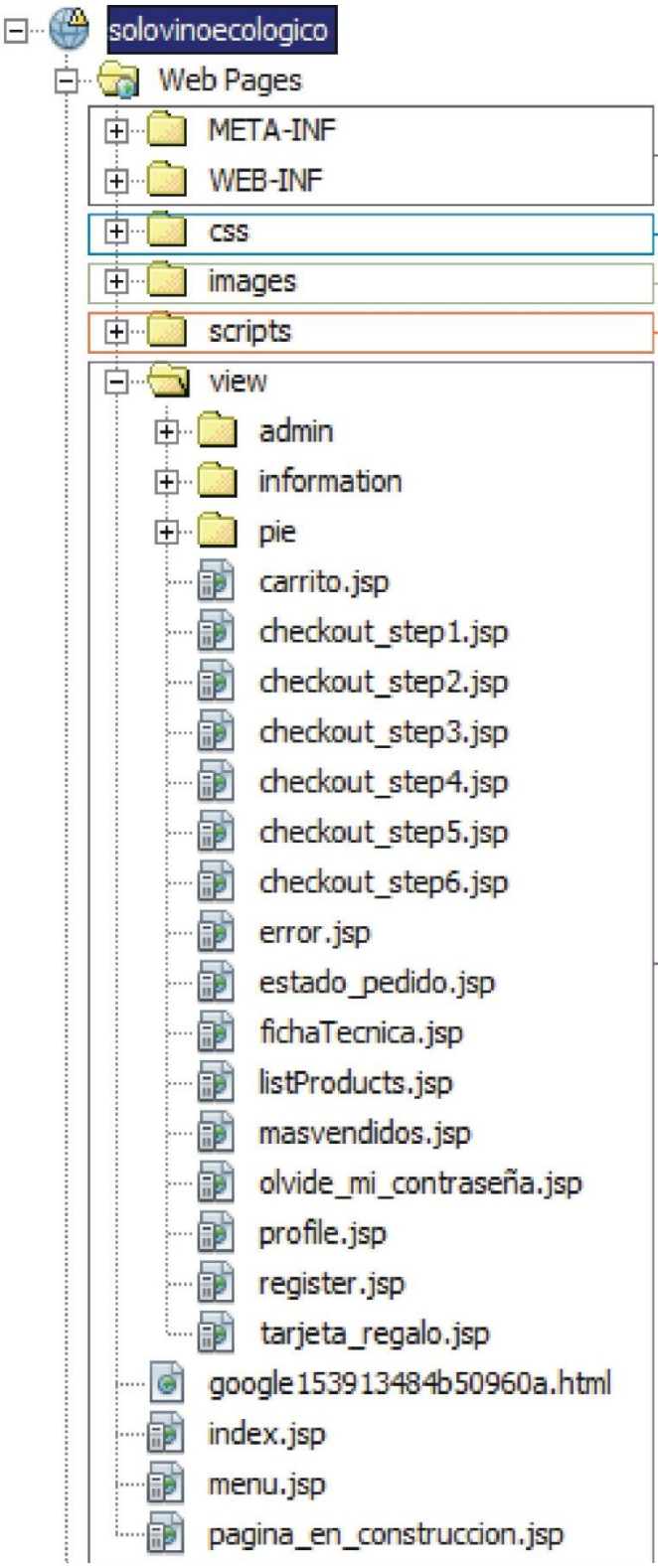
Uno de ellos es la encriptación de las contraseñas. Cuando un visitante se registra para pasar a ser un usuario, deberá introducir una contraseña que será la que utilizará para iniciar una sesión como usuario registrado. Una vez aceptado el proceso de registro la aplicación calculará el hash de la contraseña indicada utilizando el algoritmo de encriptación MD5 y será el valor de este hash el que guardaremos en la base de datos. Por tanto, la base de datos no tendrá ninguna contraseña, sino el hash de las mismas. De esta forma, cuando un usuario quiera iniciar una sesión introducirá el email y la contraseña. La aplicación calculará el hash del valor de la contraseña introducida y lo comparará con los hash de la base de datos. Este sistema permite que si un atacante se hace de los datos de la base de datos no podrá conocer la contraseña de ningún usuario.

Todos los datos que introducen los usuarios y los actores son verificados para que tengan el formato correcto. Para ello se utilizan ficheros XML de validación de datos mediante el mecanismo que proporciona struts2. En estos ficheros (que veremos más adelante), se verifica, por ejemplo, que el formato de los email sea correcto, longitud de contraseñas, entradas que deben contener solo números que no contengan otros tipos de caracteres, etc.. También dentro de algunos Actión se ha sobrescrito el método valídate() donde se verifican otras características, como que no existan usuarios con el mismo e-mail, que los valores introducidos correspondan con valores de las bases de datos…

No se ha configurado la aplicación para poder prevenir ataques, como por ejemplo inyección de código SQL. Sería interesante poder evitar estos ataques a partir de alguna librería como ESAPI [25]. Enterprise Security API se trata de una librería de código que maneja objetos y métodos que permiten validaciones y controles eficientes para evitar XSS, CSRF y muchos otros tipos de ataques web. De esta forma aumentaríamos la seguridad de nuestra aplicación contra ataques externos.

4.5. Estructura de la aplicación.

A continuación se presenta la estructura utilizada para la aplicación, es decir, sistema de directorios, ficheros de configuración, páginas .jsp, Como ya hemos explicado, estamos utilizando el el modelo MVC para el desarrollo de la aplicación. Por tanto la estructura deberá estar de forma que tengamos los ficheros correspondientes al modelo, vista y controlador de forma claramente separada. Si abrimos la aplicación en Netbeans, podemos ver la siguiente estructura:



Carpeta donde se encuentran los ficheros de configuración.

Ficheros de estilos CSS

Imágenes utilizadas en el proyecto

JavaScripts utilizados

Paginas jsp de la aplicación Están divididas en función de diversos factores (páginas del actor Administrador, páginas referentes a la información general del portal, paginas utilizadas para el pie del portal y el resto páginas de los actores Visitante y Usuario)

VISTA

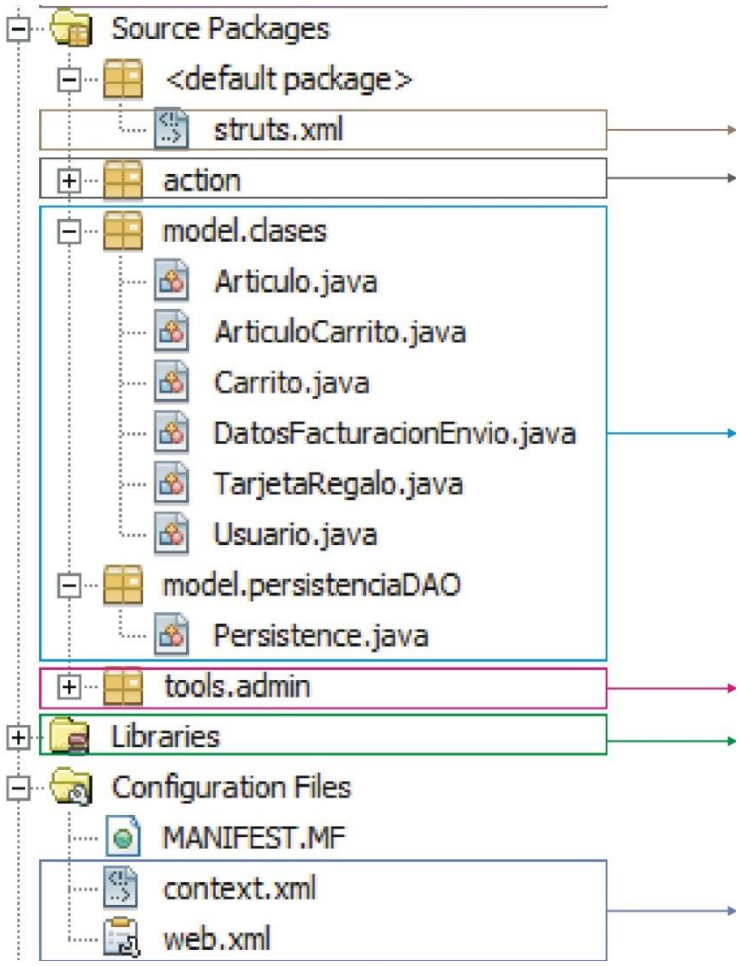


Fig. 20. Estructura de los ficheros de la aplicación

Como podemos ver, tenemos bien diferenciado la parte del Modelo y vista y Controlador.

4.6. Ficheros de configuración.

En este apartado se presentarán los ficheros de configuración que se han utilizado para el desarrollo de la aplicación.

- **web.xml:** Este es el fichero de configuración de la aplicación. En el definimos el tiempo establecido para la sesió, la página que se muestra al inicio de la aplicación, el filtro que se ejecuta y a que peticiones se ejecuta (es el filtro de struts2 y se ejecuta en todas las peticiones) y se hace una referencia al pool de conexiones.

web.xml

Source | General | Servlets | Filters | Pages | References | Security | History

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns
3      <session-config>
4          <session-timeout>
5              30
6          </session-timeout>
7      </session-config>
8      <welcome-file-list>
9          <welcome-file>index.jsp</welcome-file>
10     </welcome-file-list>
11
12     <filter>
13         <filter-name>struts2</filter-name>
14         <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPr
15     </filter>
16
17     <filter-mapping>
18         <filter-name>struts2</filter-name>
19         <url-pattern>/*</url-pattern>
20     </filter-mapping>
21
22     <resource-ref>
23         <description>Referencia al pool de conexiones</description>
24         <res-ref-name>jdbc/solovinoecologicoMySql</res-ref-name>
25         <res-type>javax.sql.DataSource</res-type>
26         <res-auth>Container</res-auth>
27     </resource-ref>
28
29 </web-app>
```

Fig. 21. Fichero de configuración web.xml

context.xml: En este fichero configuramos el pool de conexiones (número de conexiones, usuario, contraseña, nombre del jdbc, etc…).

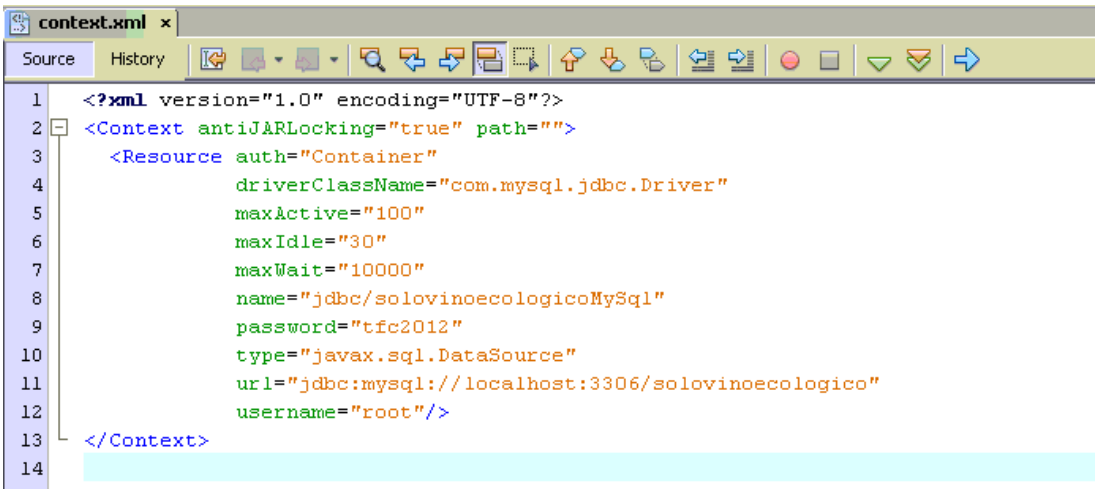


Fig. 22. Fichero de configuración context.xml

- **struts.xml:** este es el fichero de configuración de struts. En él se especificarán los action, con lo que trabaja la aplicación, con sus respectivos resultados, etc… Hay que decir que no todos los Actions se han configurado mediante XML. Muchos de ellos están configurados utilizando el mecanismo de anotaciones en el mismo action.

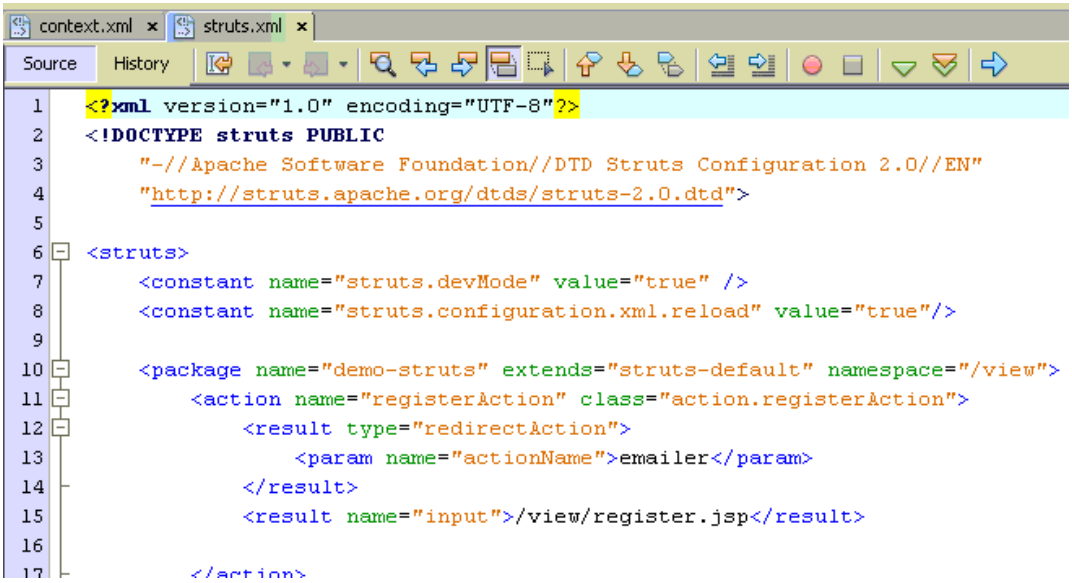


Fig. 23. Fichero de configuración struts.xml

ficheros de validación: estos ficheros del tipo .xml, están asociados a un action y en ellos se especificarán las comprobaciones que se realizan a los datos que introduce el usuario. El formato del nombre estos ficheros será, el nombre del actión + -validatión. Por ejemplo para el action "registerAction" el nombre del fichero de validación será "registerAction-validation.xml". Por tanto, para cada action que tengamos que verificar datos de entrada, tendremos un fichero asociado de validación.

context.xml × struts.xml × registerAction-validation.xml ×

Source History

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN" "htt
3  <validators>
4      <field name="nombre">
5          <field-validator type="requiredstring">
6              <param name="trim">true</param>
7              <message>El nombre del usuario es un campo obligatorio.</message>
8          </field-validator>
9      </field>
10
11     <field name="apellidos">
12         <field-validator type="requiredstring">
13             <param name="trim">true</param>
14             <message>Los apellidos del usuario es un campo obligatorio.</message>
15         </field-validator>
16     </field>
17
18     <field name="pais">
19         <field-validator type="requiredstring">
20             <param name="trim">true</param>
21             <message>El Pais es un campo obligatorio.</message>
22         </field-validator>
23     </field>
24
25     <field name="provincia">
26         <field-validator type="requiredstring">
27             <param name="trim">true</param>
28             <message>La Provincia es un campo obligatorio.</message>
29         </field-validator>
30     </field>
```

Fig. 24. Fichero de configuración registerAction-validation.xml

5. Conclusiones

La realización de este proyecto me ha permitido profundizar el desarrollo de aplicaciones utilizando la tecnología java y en particular J2EE. No cabe duda de que la tecnología es apasionante y con muchas posibilidades pero a la vez muy extensa (por lo menos para el tiempo que se dispone para la realización del proyecto). Lo primero que percibí al comenzar el proyecto es la extensión del tema, ya que no solo es la arquitectura J2EE, sino que esta arquitectura está rodeada de una gran cantidad de frameworks, por lo que a medida que avanzaba un paso durante mi aprendizaje se me abrían nuevas puertas para explorar.

He de decir que mi conocimiento sobre la tecnología J2EE era nulo, por lo que he tenido que realizar un gran esfuerzo en el aprendizaje de esta tecnología y frameworks que la rodean. Y es que no solo he tenido que formarme en J2EE, sino que el lenguaje HTML, CSS, JavaScript para la creación de las páginas jsp también ha absorbido gran parte del tiempo dedicado al aprendizaje.

Una de las principales experiencias que me llevo con la realización de este proyecto y que pienso que puede ser de gran valor en mi vida laboral es haber aprendido a reconocer la importancia de las primeras fases de la realización de un proyecto de estas características. Y es que creo que uno de los errores que me han hecho perder más tiempo, es no haber definido de forma más precisa toda la parte de la lógica de aplicación (diagrama de clases, diagramas E/R,…). El hecho de redefinir las clases de la estructura de la aplicación hace que se tenga que realizar un trabajo extra que se puede evitar dedicando más tiempo a las primeras fases.

Otra experiencia importante que he obtenido de este proyecto es la utilización de patrones de diseño. Durante mi paso por la UOC, aunque se había hecho mención, no habíamos trabajado con ninguno de ellos. En este proyecto se utilizan dos patrones de diseño (MVC y DAO). Ahora entiendo la importancia de ellos, ya que cuando se hacen aplicaciones de gran dimensiones pienso es necesario tener todo lo más estandarizado posible.

El haber realizado todo el ciclo de vida de una aplicación, desde la elaboración del Plan de trabajo hasta la implementación y pruebas de la aplicación, creo que ha sido una

experiencia muy positiva, ya que hasta ahora había realizado parte del ciclo de vida por separado en diferentes asignaturas. Es ahora cuando he realizado todo el ciclo de vida de un mismo proyecto que me dado cuenta de la importancia de cada una de las fases.

Una vez finalizado el proyecto puedo decir que me siento satisfecho, ya que he podido crear un producto, el cual podría estar comercializado (por supuesto, con mejoras en el apartado de seguridad, tema que no he podido abarcar como me hubiera gustado). Se podría decir que han cobrado forma conocimientos que he ido adquiriendo de diferentes asignaturas, dando como resultado la aplicación presentada.

6. Bibliografía.

[1] http://www2.uah.es/jcaceres/capsulas/DiagramaCasosDeUso.pdf

[2] http://es.wikipedia.org/wiki/Tomcat

[3 https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&ved=0CDwQFjAB&url=https%3A%2F%2Fforja.rediris.es%2Fdocman%2Fview.php%2F282%2F558%2FmanualDesarrollador.pdf&ei=V8PqUIb9GpKShgeQ9YHgBQ&usg=AFQjCNGNkoeo9LR7EcvtzZ57AXmDWz4i3A&bvm=bv.1355534169,d.ZG4

[4] http://www.danielpecos.com/docs/mysql_postgres/x57.html

[5] http://ccia.ei.uvigo.es/docencia/SCS/1112/transparencias/Tema5-1.pdf

[6] http://es.scribd.com/doc/81350989/45/Patrones-arquitectonicos-MVC-y-Layers-1

[7] http://bibing.us.es/proyectos/abreproy/11270/fichero/MEMORIA%252FCAPITULO_05.pdf

[8] http://es.wikipedia.org/wiki/JQuery

[9] http://struts.apache.org/2.x/

[10] http://www.javatutoriales.com/2011/06/struts-2-parte-1-configuracion.html

[11] http://www.oracle.com/technetwork/java/javamail/index.html

[12] http://zarza.usal.es/~fgarcia/docencia/poo/02-03/trabajos/S2T8.pdf

[13] http://www.mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/

[14] http://www.javamexico.org/blogs/richardmx/que_es_data_access_object

[15] http://isg3.pbworks.com/w/page/7624463/Dise%C3%B1o%20de%20la%20capa%20de%20datos

[16] http://www.oracle.com/technetwork/java/javase/jdbc/index.html

[17] http://isg3.pbworks.com/w/page/7624463/Dise%C3%B1o%20de%20la%20capa%20de%20datos

[18] http://netbeans.org/

[19] http://ruizji.blogspot.com.es/2012/02/ciclo-de-vida-del-software.html

[20] http://www.javahispano.org/portada/2011/7/28/tutorial-basico-de-java-ee-por-abraham-otero.html

[21] Struts 2: El Framework De Desarrollo De Aplicaciones Java EE - Jerome Lafosse

[23] http://es.scribd.com/doc/70705402/2/EL-PATRON-MVC

[24] http://www.lcc.uma.es/~galvez/ftp/libros/JavaMail.pdf

[25] http://www.e-securing.com/novedad.aspx?id=68

Material de las asignaturas de la UOC de Bases de Datos, Ingeniería de software, Programación Orientada a Objetos, Estructura de la Información.