TFC - J2EE

Proyecto Tienda Virtual:

"Uniformes Escolares"

Alumno: Rubén Teja Rubio
Titulación: Ingeniería Técnica en Informática de Gestión
Tutor: Salvador Campo Mazarico
Fecha: Enero 2016

Dedicatoria y agradecimientos

La realización tanto de este documento como del resto de la asignatura y demás asignaturas de la carrera no hubiera sido posible sin el apoyo incondicional de muchas personas que de una forma u otra me han ayudado, animado, sostenido y a veces hasta levantado a lo largo de estos años.

Mi principal agradecimiento es para Charo, mi mujer, que durante muchos años, no solo durante mi estancia en la UOC, ha sido mi apoyo para poder sacar adelante los estudios. También para mis dos pequeñas mellizas Lorena y Rocío que ya han ido comprendiendo que su papá también estudia y hace exámenes como ellas y me daban las más cariñosas regañinas cuando 'no hacía los deberes'. Espero poder devolverlas pronto todo el tiempo que he dejado de dedicarlas.

También mis recuerdo va para el resto de mi familia, mis padres Pepe y Pili y mis suegros Antonio y Rosario, que me han ayudado y han sabido comprender que a estas alturas, con el trabajo, con los hijos y otras responsabilidades, me haya dedicado a estudiar una carrera. Y por supuesto para mi hermano mayor, José Luis, del que probablemente me venga la vocación informática desde aquel día en que trajo a casa el Spectrum de 48K.

Por último me gustaría también agradecer a los profesores y tutores de la UOC con los que he coincidido durante la realización de las asignaturas. En una universidad como la UOC donde habitualmente no tratas mucho con otros compañeros, su soporte y apoyo ha sido muy importantes para superar cada uno de los peldaños hasta el final de la carrera.

Muchas gracias a todos.

Resumen

Mediante el presente documento se ha realizado la Memoria de la asignatura Trabajo Fin de Carrera en el área de J2EE donde se detalla el proceso de creación de una aplicación web usando el lenguaje de programación Java y tecnologías y herramientas afines a este lenguaje.

El proyecto que he realizado, *Uniformes Escolares*, consiste en la implementación de un tienda online que gestione la venta de las prendas de vestir de los uniformes de un colegio. El sistema será privado y para acceder a la aplicación es necesario realizar previamente una autentificación como usuario autorizado. Dependiendo del perfil del usuario se accederá a distintos módulos o subsistemas. Los usuarios administradores podrán acceder al módulo de mantenimiento con posibilidad de realizar tareas de alta/baja/modificación de elementos del sistema. Los usuarios con perfil de cliente accederán al módulo de ventas donde podrán seleccionar productos y realizar un proceso de compra online.

Además de las características funcionales propias de una web de comercio online, la aplicación se ha implementado de modo que cuente con funcionalidades como la autentificación de seguridad y encapsulado de módulos en base al perfil y selección multi-idioma con cambio de lenguaje inmediato.

Para la realización de la aplicación se han utilizados frameworks y tecnologías integrables con J2EE como Hibernate, JPA, Spring MVC y JSP. Como base de datos se ha hecho uso de MySQL y como servidor web se ha utilizado Apache Tomcat. Estas tecnologías y su uso en la aplicación se comentarán con más detalle en los siguientes capítulos.

Índice de contenidos

Índice de figuras

Capitulo 1. Introducción.

1.1 Justificación del TFC y contexto en el que se desarrolla: punto de partida y aportación del TFC.

La realización de una asignatura como TFC me ha permitido tener la posibilidad de poner en práctica los conocimientos y técnicas adquiridas al cursar las distintas asignaturas de la carrera, en especial las relacionadas con la programación y la gestión de proyectos.

La decisión de optar por un proyecto del área J2EE fue sencilla ya que me siento cómodo trabajando con la plataforma Java y tengo experiencia profesional en el desarrollo de aplicaciones web con este lenguaje y con frameworks de desarrollo.

En cuanto a la idea de realizar una tienda virtual para la venta de uniformes escolares surge de una experiencia personal propia donde mientras esperaba una larga cola para comprar en una tienda física dichos uniformes imagine lo sencillo y cómodo que sería si se pudiera realizar dicha compra de forma online.

El punto de partida del proyecto ha sido probablemente uno de los momentos más complicados porque hay que tomar la decisión de qué tipo de aplicación vamos a realizar y que tecnologías vamos a utilizar. Además una vez hecha esta última elección, en algunos casos nos enfrentamos a una amplia curva de aprendizaje de alguna de las tecnologías que nos puede llevar un tiempo del que en ocasiones no disponemos. A cambio esta circunstancia me ha servido para aumentar de forma sustancial mis conocimientos en herramientas y software que no conocía y una vez superados los problemas me ha dado seguridad a la hora de afrontar las siguientes fases del proyecto.

1.2 Objetivos del TFC.

Los objetivos que he pretendido conseguir en la realización del TFC son:

- Diseño de un proyecto web siguiendo las fases definidas por la Ingeniería del Software
- Utilización de tecnologías Java EE aplicadas al desarrollo de aplicaciones web.
- Integración de distintos marcos de desarrollo (frameworks) con el lenguaje Java.
- Uso de patrones MVC (Modelo Vista Controlador).
- Búsqueda de información de referencia y su aplicación en el proyecto.
- Generación de documentación funcional, técnica y memoria del proyecto.

1.3 Enfoque y método seguido.

Para la creación del proyecto se ha seguido un modelo de ciclo de vida en cascada en que el que se ha ido completando los objetivos especificados en cada fase del proyecto de forma secuencial. Aunque una de las principales desventajas que tiene este modelo es la rigidez, hay que tener en cuenta que estamos hablando de un proyecto académico y que viene marcado por el cumplimiento de unos hitos ya predeterminados, las PECs, que hay que completar en el tiempo requerido.

Las etapas que se han completado durante el proyecto son las siguientes:

- Planificación del proyecto
- Definición funcional
- Análisis
- Diseño
- Implementación
- Pruebas y Documentación

1.4 Planificación del Proyecto.

1.4.1 Planificación general

En el siguiente cuadro se muestra la planificación de actividades de las cuatro pruebas de evaluación continua en que se ha divido el TFC. Además se presenta las fecha de comienzo y final de cada actividad así como la etapa o etapas con las que se corresponde:

Actividad	Etapa	Fecha Inicio	Fecha Fin
PEC1	Plan de trabajo	16/09/2015	30/09/2015
PEC2	Análisis de requerimientos y diseño de arquitectura	01/10/2015	04/11/2015
PEC3	Implementación	05/11/2015	18/12/2015
Entrega Final	Entrega final del Software, Memoria y Presentación Virtual	19/12/2015	11/01/2016

1.4.2 Planificación por etapas

Plan de trabajo (PEC1)

Plan de Trabajo	**16/09/15**	**30/09/15**	11 días
Lectura y comprensión de tareas	17/09/15	21/09/15	3 días
Redacción del plan de trabajo	22/09/15	29/09/15	6 días
Entrega PEC1 - Plan de trabajo	30/09/15	30/09/15	1 día

Análisis de requerimientos y diseño de arquitectura (PEC2)

Análisis de requerimientos y Diseño de arquitectura	**01/10/15**	**04/11/15**	**25 días**
Comprensión de requerimientos	01/10/15	04/10/15	3 días
Diseño de clases	05/10/15	11/10/15	6 días
Diseño de casos de uso	12/10/15	18/10/15	6 días
Diseño de modelo de datos	19/10/15	25/10/15	6 días
Implementación de entorno de desarrollo	26/10/15	30/10/15	5 días
Generación de documento de entrega PEC2	31/10/15	03/11/15	3 días
Entrega PEC2	04/11/15	04/11/15	1 día

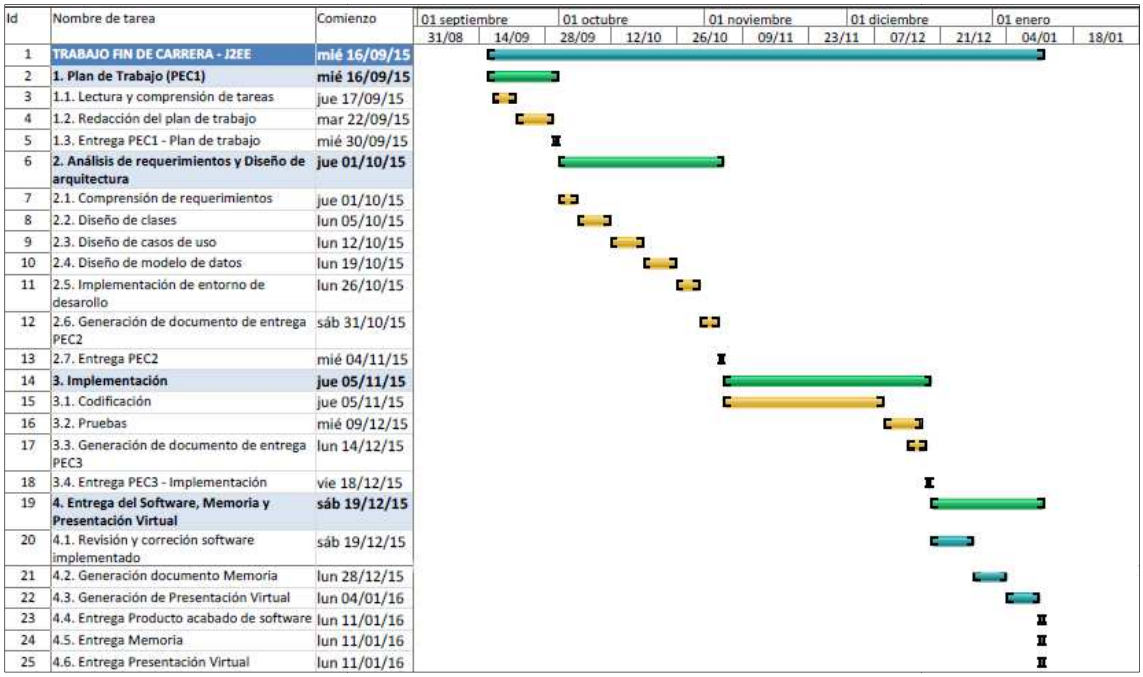
Implementación (PEC3)

Implementación	**05/11/15**	**18/12/15**	**32 días**
Codificación	05/11/15	08/12/15	24 días
Pruebas	09/12/15	16/12/15	6 días
Generación de documento de entrega PEC3	14/12/15	17/12/15	4 días
Entrega PEC3 - Implementación	18/12/15	18/12/15	1 día

Entrega final del Software, Memoria y Presentación Virtual

Entrega del Software, Memoria y Presentación Virtual	**19/12/15**	**11/01/16**	**17 días**
Revisión y corrección software implementado	19/12/15	27/12/15	7 días
Generación documento Memoria	28/12/15	03/01/16	6 días
Generación de Presentación Virtual	04/01/16	10/01/16	6 días
Entrega Producto acabado de software	11/01/16	11/01/16	1 día
Entrega Memoria	11/01/16	11/01/16	1 día
Entrega Presentación Virtual	11/01/16	11/01/16	1 día

En la siguiente figura de muestra un diagrama de Gantt con la planificación global del proyecto:



Id	Nombre de tarea	Comienzo
1	**TRABAJO FIN DE CARRERA - J2EE**	**mié 16/09/15**
2	**1. Plan de Trabajo (PEC1)**	**mié 16/09/15**
3	1.1. Lectura y comprensión de tareas	jue 17/09/15
4	1.2. Redacción del plan de trabajo	mar 22/09/15
5	1.3. Entrega PEC1 - Plan de trabajo	mié 30/09/15
6	**2. Análisis de requerimientos y Diseño de arquitectura**	**jue 01/10/15**
7	2.1. Comprensión de requerimientos	jue 01/10/15
8	2.2. Diseño de clases	lun 05/10/15
9	2.3. Diseño de casos de uso	lun 12/10/15
10	2.4. Diseño de modelo de datos	lun 19/10/15
11	2.5. Implementación de entorno de desarollo	lun 26/10/15
12	2.6. Generación de documento de entrega PEC2	sáb 31/10/15
13	2.7. Entrega PEC2	mié 04/11/15
14	**3. Implementación**	**jue 05/11/15**
15	3.1. Codificación	jue 05/11/15
16	3.2. Pruebas	mié 09/12/15
17	3.3. Generación de documento de entrega PEC3	lun 14/12/15
18	3.4. Entrega PEC3 - Implementación	vie 18/12/15
19	**4. Entrega del Software, Memoria y Presentación Virtual**	**sáb 19/12/15**
20	4.1. Revisión y correción software implementado	sáb 19/12/15
21	4.2. Generación documento Memoria	lun 28/12/15
22	4.3. Generación de Presentación Virtual	lun 04/01/16
23	4.4. Entrega Producto acabado de software	lun 11/01/16
24	4.5. Entrega Memoria	lun 11/01/16
25	4.6. Entrega Presentación Virtual	lun 11/01/16

Figura 1. Planificación global del proyecto.

1.5 Productos obtenidos.

Tras la realización del Trabajo fin de Carrera se han obtenido los siguientes productos:

Código fuente de la aplicación. Se aporta un fichero *.war* que puede ser directamente añadido y desplegado por el servidor de aplicaciones.

Script SQL. Genera el esquema de datos de la aplicación , las tablas y relaciones así como datos maestros y de prueba.

Presentación virtual. Extracto del contenido de la memoria.

Memoria del proyecto. Es el presente documento.

1.6 Resumen de otros capítulos.

En los siguientes capítulos de este documento de Memoria se describirán los siguientes aspectos:

Capitulo 2. Análisis funcional, donde se hace una descripción de la aplicación y se detallan los requerimientos funcionales de la misma.

Capitulo 3. Diseño, donde se especifican la arquitectura y el diseño técnico de la aplicación junto con las tecnologías usadas.

Capitulo 4. Implementación, donde se explicarán los requerimientos de software y las decisiones tomadas a la hora de realizar el desarrollo de la aplicación.

Capítulo 5. Coste económico. Se intentará aproximar la cuantía económica de nuestro proyecto.

Capítulo 6. Conclusiones. Resumen final y reflexión tanto personal como acerca del proyecto.

Glosario. Descripción de términos que aparecen en este documento.

Bibliografía. Listado de libros y páginas webs de referencia en las que me he apoyado para realizar el proyecto.

Anexo I. Script de creación del esquema de datos.

Capitulo 2. Análisis funcional.

2.1 Descripción funcional.

La aplicación *Uniformes Escolares* se ha construido pensando en las funcionalidades clásicas de un portal de venta de productos online. Hay que especificar en este punto que las funcionalidades que aparecen en este tipo de aplicaciones están muy condicionadas por el tipo de producto que se quiere vender y normalmente se suelen añadir de forma modular según la necesidad.

En el caso de *Uniformes Escolares* se ha implementado una aplicación web que vende uniformes para un colegio. El tipo de producto no es especialmente variado ni cambia mucho en sus características por lo que las funcionalidades requeridas no son complejas y se corresponden sin problema a lo que se puede esperar una tienda virtual.

El programa se ha pensado de forma que quede muy claramente dividida en dos módulos o subsistemas independientes entre sí, aunque por supuesto relacionados, condicionando el acceso a cada uno de ellos en base al perfil que tenga asignado el usuario. Un tercer subsistema general es la parte de autentificación donde el usuario será redirigido al módulo correspondiente tras registrarse correctamente en el sistema.

2.2 Actores.

La aplicación se ha desarrollado para que pueda ser usada por dos tipo distintos de actores o usuarios:

Cliente: El usuario autentificado como cliente solo puede acceder al módulo de ventas. En este subsistema de la aplicación puede acceder a las siguientes funcionalidades:

- Consulta de productos: El cliente podrá visualizar listados de todos los productos o listados filtrados por categorías.
- Añadir productos al carrito: Puede seleccionar un producto y añadirlo a la cesta de la compra.
- Visualizar carrito: El cliente pude visualizar el contenido, cantidad y coste de los productos que ha añadido al carrito.
- Formalizar pedido: El cliente valida el pedido para inicializar el proceso de compra.
- Pago de pedido: El cliente finaliza el proceso de compra con el pago del pedido.
- Consulta de pedidos: El cliente puede consultar un listado con los pedidos que ha realizado y el detalle de estos.

Administrador: El usuario Administrador únicamente puede acceder al módulo de administración. En este módulo puede realizar las siguientes acciones:

- Consulta de tablas maestras: El administrador puede visualizar listados de Usuarios, Perfiles, Artículos, Categorías y, Atributos..
- Modificación de tablas maestras: El usuario administrador tiene permisos para crear, modificar y eliminar usuarios, perfiles, artículos, categorías y atributos.
- Consulta de pedidos: El administrador podrá visualizar un listado de todos los pedidos realizados.
- Actualización de pedidos: El administrador puede cambiar el estado de un pedido.

Además de estas funcionalidades, ambos actores pueden acceder a la pantalla de autentificación en el subsistema principal donde podrán registrar para ser re direccionados a su módulo correspondiente

En la siguiente figura se muestra un diagrama de caso de casos de uso y actores donde se refleja lo anteriormente descrito:

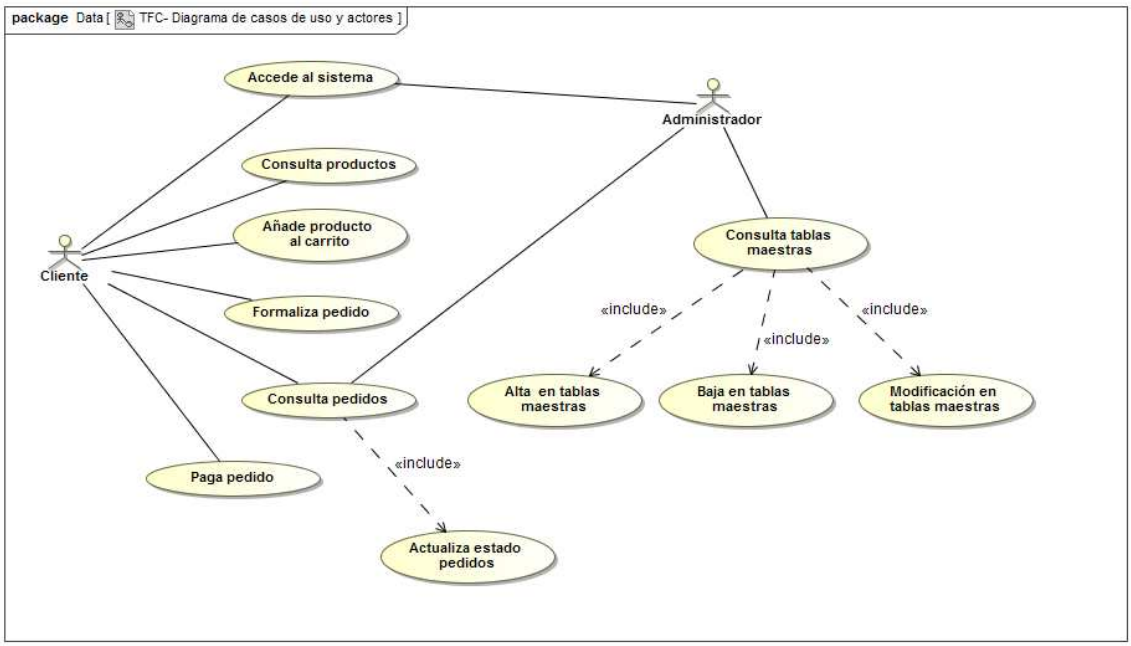


Figura 2. Diagrama de caso de uso y actores

2.3 Módulos de la aplicación.

Como se ha comentado en puntos anteriores, la aplicación se ha dividido en tres subsistemas diferenciados:

- Subsistema general o de usuarios.
- Subsistema de venta
- Subsistema de administración

2.3.1 Subsistema de usuarios

El módulo o subsistema de usuarios comprende la primera pantalla de la aplicación donde el usuario se autentificará contra el sistema. Dependiendo del perfil de dicho usuario se accederá a la pantalla de la tienda virtual donde se continuará con el proceso de compra, en el caso del usuario con perfil de cliente, o si se accede con perfil administrador, se pasará a la pantalla de mantenimiento de la tienda.

En el proceso de autentificación se gestionará tanto el perfil del usuario como la sesión que tendrá para el resto del proceso así como la seguridad del sistema.

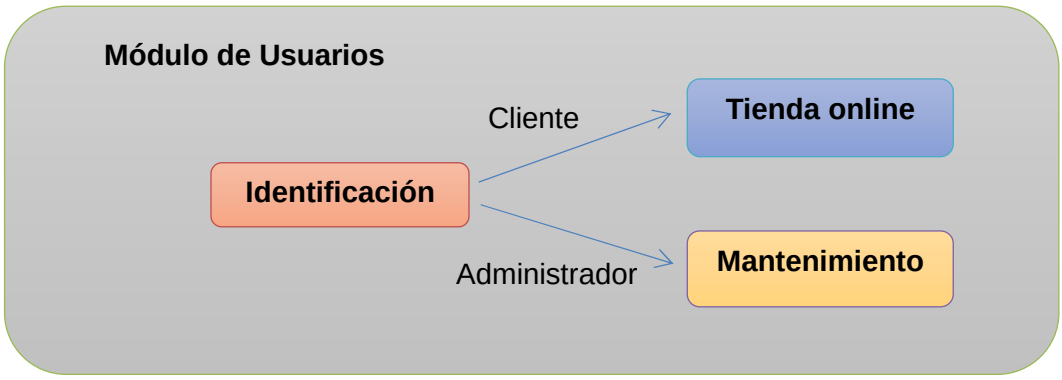


Figura 3. Esquema del módulo de usuarios.

La pantalla de autentificación de la aplicación consiste en un formulario con dos campos para introducir el usuario y contraseña. El sistema validará al usuario y si la autentificación no es correcta se mostrará un mensaje de error. Si el acceso es correcto se adjudica sesión al usuario que se mantendrá mientras interactué dentro de los otros subsistemas de la aplicación o cierre voluntariamente la sesión.

2.3.2 Subsistema de venta

El módulo de venta corresponde a la parte principal de la tienda online. El usuario cliente una vez autentificado en el sistema visualizará una serie de productos categorizados los cuales podrá ir seleccionando e ir agregando al carrito de la compra. Cuando el cliente decida validar el pedido realizado, procederá a finalizar el proceso de compra. En todo momento, el cliente puede consultar la información acerca de sus propios pedidos ya realizados.

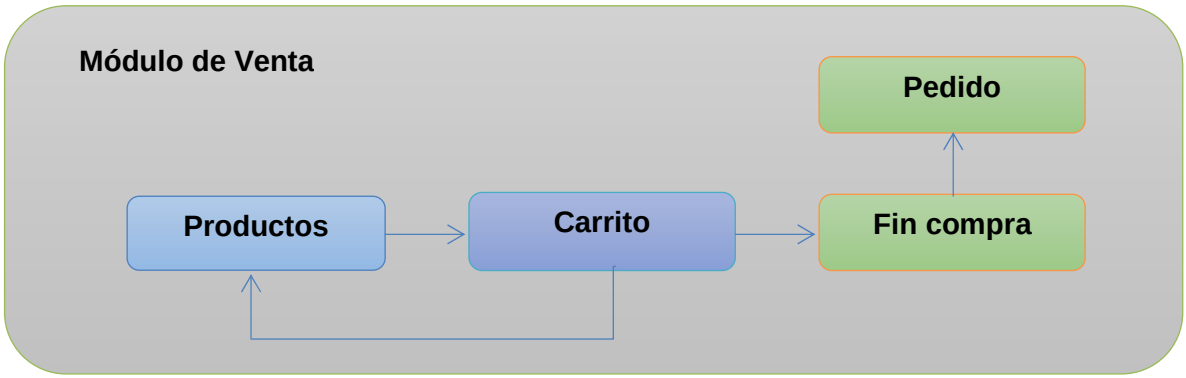


Figura 4. Esquema del módulo de venta.

En el modulo de ventas encontramos la pantalla divida en cuatro componentes con distintas funcionalidades:

- **Cabecera:** En la parte superior de la pantalla encontramos los iconos (banderas) para elegir cambiar a otro idioma los textos de la aplicación. En *Uniformes Escolares* se han incluido los textos en idioma Castellano, Inglés y Catalán.
- **Menú lateral:** El menú lateral izquierdo despliega un listado de las distintas categorías de productos por las que podemos filtrar. Además contiene un enlace al detalle de los pedidos realizados por el usuario.
- **Pantalla Principal:** Es donde se muestra el contenido principal de la aplicación.
- **Carrito de la compra:** Situado en la derecha de la pantalla principal, muestra los productos seleccionados por el cliente así como las cantidades y precio unitario y total. Si se pulsa el botón de validar pedido se comienza el proceso de validación del la compra.

2.3.3 Subsistema de administración

Al módulo de mantenimiento o administración solo podrán acceder una vez autentificados, los usuarios con perfil administrador. En estas pantallas se podrá visualizar los listados de usuarios, artículos, perfiles, atributos y categorías y se permitirá el alta, baja y modificación de elementos de estas entidades.
También se podrá en el caso de los pedidos, actualizar su estado.

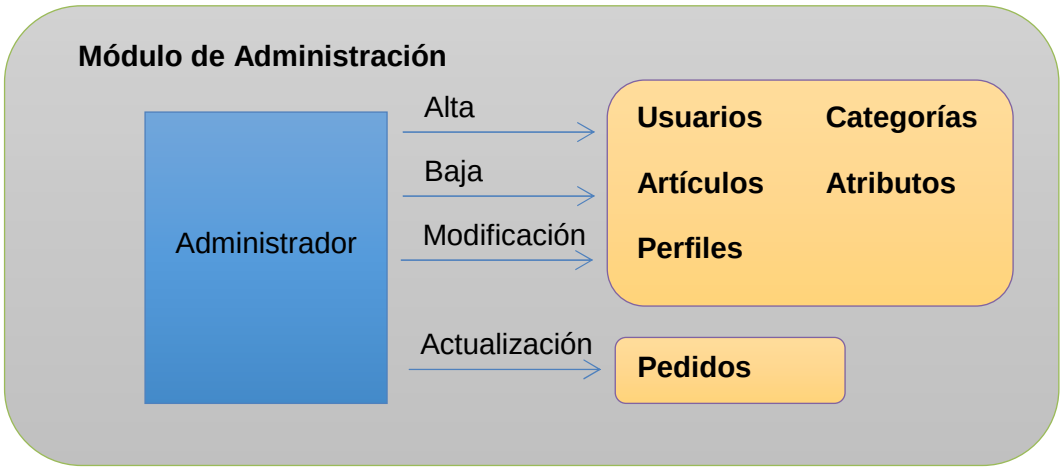


Figura 5. Esquema del módulo de administración.

En el modulo de administración encontramos la pantalla divida en tres componentes con distintas funcionalidades:

- **Cabecera:** En la parte superior de la pantalla encontramos también los iconos (banderas) para elegir cambiar a otro idioma los textos de la aplicación. Como en el módulo de ventas se podrá elegir entre idioma Castellano, Inglés y Catalán.
- **Menú lateral:** El menú lateral izquierdo despliega un listado de las distintas tablas maestras que podemos consultar. estas son Usuarios, Artículos, Pedidos, Categorías, Atributos y Perfiles.
- **Pantalla Principal:** Es donde se muestra el contenido principal de la aplicación. En este subsistema de mantenimiento, es necesario especificar tres tipos de pantallas:
  - Listados. Los listados nos muestran una tabla con la información almacenada en forma de registros. En esta pantalla podemos crear un nuevo registro con lo que se nos re direccionará a una nueva pantalla, eliminar un registro con lo que lo daremos de baja de la base de datos o modificar un registro con lo que también seremos re direccionados a una nueva pantalla.
  - Altas. Las pantallas de alta constan de un formulario con un número de campos que hay que completar para crear un nuevo registro. El formulario cuenta con un control de validación de campos por lo que si hay algo incorrecto se mostrará un mensaje específico informándolo. Si se valida correctamente el formulario se crea un nuevo registro. En el caso de la tabla de pedidos no es posible crear un nuevo elemento. Este proceso se realiza en el módulo de ventas.
  - Modificaciones. Las pantallas de modificación consisten en un formulario con determinados campos que solo se muestran habilitados si son susceptibles de ser modificados. Al igual que en el alta, existe un control de validación de campos que muestra un mensaje de error si dicha validación es incorrecta. Si la validación se hace correctamente el registro modificado aparece actualizado con el nuevo valor asignado.

2.4 Especificación de casos de uso.

En un punto anterior se ha mostrado un diagrama de casos de uso y actores para describir de forma general las funcionalidades a las que los distintos usuarios pueden acceder. Los casos de uso en su representación textual están relacionados con el diagrama de casos de uso pero son diferentes ya que dan una descripción a mayor nivel de detalle del comportamiento del sistema y de los actores implicados.

A continuación se muestran los casos de uso más relevantes que se han identificado en el proyecto:

Acceso al sistema

CASO DE USO: Acceso al sistema	
IDENTIFICACIÓN	CU-TFC1
DESCRIPCIÓN	Acceso al sistema
ACTORES	Cliente, Administrador
PRECONDICIONES	N/A
CASOS DE USO RELACIONADOS	N/A
ESCENARIO	• El usuario introduce identificador de usuario y contraseña. • El sistema navegara a la pantalla que corresponda a la funcionalidad del usuario.
ESCENARIO ALTERNATIVO	No existen el usuario o los datos introducidos no son correctos.
RESULTADO	El usuario accede a la pantalla correspondiente a su perfil
POSTCONDICIONES	N/A

Consulta de productos

CASO DE USO: Consulta de productos	
IDENTIFICACIÓN	CU-TFC2
DESCRIPCIÓN	Consulta de productos de la tienda virtual.
ACTORES	Cliente
PRECONDICIONES	El usuario debe estar identificado con perfil ‘Cliente’
CASOS DE USO RELACIONADOS	CU-TFC1
ESCENARIO	• Al acceder a la primera pantalla de la tienda se muestra un listado de productos • Al seleccionar una categoría se mostrará el listado de productos
ESCENARIO ALTERNATIVO	No existen productos que consultar
RESULTADO	Se listan los productos en pantalla
POSTCONDICIONES	N/A

Añade producto al carrito

CASO DE USO: Añade producto a carrito	
IDENTIFICACIÓN	CU-TFC3
DESCRIPCIÓN	Añade un producto al carrito de compra
ACTORES	Cliente
PRECONDICIONES	El usuario debe estar identificado con perfil 'Cliente'
CASOS DE USO RELACIONADOS	CU-TFC1
ESCENARIO	• El cliente pulsa el botón 'Añadir' de un producto. • El producto se añade al carrito de compra.
ESCENARIO ALTERNATIVO	N/A
RESULTADO	El producto se añade al carrito. Se muestra el producto añadido junto a los otros productos.
POSTCONDICIONES	N/A

Formaliza pedido

CASO DE USO: Formaliza pedido	
IDENTIFICACIÓN	CU-TFC4
DESCRIPCIÓN	Formalización de pedido con los artículos previamente añadidos al carrito.
ACTORES	Cliente
PRECONDICIONES	El usuario debe estar identificado con perfil 'Cliente'. El cliente debe haber añadido al menos un producto al carrito.
CASOS DE USO RELACIONADOS	CU-TFC1, CU-TFC3
ESCENARIO	• El cliente pulsa el botón de confirmar el carrito. • Se formaliza el carrito como pedido iniciándose el proceso de compra.
ESCENARIO ALTERNATIVO	N/A
RESULTADO	Se inicia el proceso de compra de un pedido.
POSTCONDICIONES	N/A

Consulta de pedidos

CASO DE USO: Consulta de pedidos	
IDENTIFICACIÓN	CU-TFC5
DESCRIPCIÓN	Consulta de pedidos existentes
ACTORES	Cliente, Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Cliente' o 'Administrador'
CASOS DE USO RELACIONADOS	CU-TFC1
ESCENARIO	• El usuario accede a la pantalla de consulta de pedidos • Al usuario Cliente solo se le mostrarán sus propios pedidos. El usuario Administrador podrá visualizar a los pedidos de todos los clientes.

ESCENARIO ALTERNATIVO	No existen pedidos a mostrar
RESULTADO	Se muestran los pedidos realizados por pantalla.
POSTCONDICIONES	N/A

Pago de pedido

CASO DE USO: Pago de pedido	
IDENTIFICACIÓN	CU-TFC6
DESCRIPCIÓN	Pago y finalización del proceso de compra.
ACTORES	Cliente, Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Cliente' o 'Administrador'. El cliente ha añadido productos al carrito y ha formalizado este como pedido.
CASOS DE USO RELACIONADOS	CU-TFC1, CU-TFC3, TFC4
ESCENARIO	• El cliente accede a la pantalla de pago • Se introducen los datos de pago en el TPV virtual. • Se devuelve la confirmación de compra • Se muestra un resumen del pedido realizado.
ESCENARIO ALTERNATIVO	No se finaliza el proceso de compra y no se crea el nuevo pedido.
RESULTADO	Se realiza el proceso de compra y se crea un nuevo pedido.
POSTCONDICIONES	Se produce un alta de pedido en base de datos.

Consulta de tablas maestras

CASO DE USO: Consulta de tablas maestras	
IDENTIFICACIÓN	CU-TFC7
DESCRIPCIÓN	Consulta de tablas maestras
ACTORES	Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Administrador'.
CASOS DE USO RELACIONADOS	CU-TFC1
ESCENARIO	• El administrador accede a la pantalla de consulta de alguna de las tablas maestras • Se muestra la lista de las tablas • El Administrador tendrá la opción de crear, modificar o eliminar registros de las tablas.
ESCENARIO ALTERNATIVO	N/A
RESULTADO	Se muestra una lista de las tabla maestra
POSTCONDICIONES	N/A

Alta en tablas maestras

CASO DE USO: Alta en tablas maestras	
IDENTIFICACIÓN	CU-TFC8
DESCRIPCIÓN	Alta de registros en tablas maestras
ACTORES	Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Administrador'. El administrador ha listado una tabla maestra y ha pulsado el botón de nuevo registro.
CASOS DE USO RELACIONADOS	CU-TFC1, CU-TFC7
ESCENARIO	• El administrador accede a la pantalla de alta de alguna de las tablas maestras • El administrador rellena el formulario con los datos requeridos. • El Administrador valida el nuevo registro.
ESCENARIO ALTERNATIVO	El nuevo registro no se valida correctamente y no se guarda.
RESULTADO	Se crea un nuevo registro de la tabla maestra.
POSTCONDICIONES	Se crea un nuevo registro en la tabla de base de datos.

Baja en tablas maestras

CASO DE USO: Baja en tablas maestras	
IDENTIFICACIÓN	CU-TFC9
DESCRIPCIÓN	Baja de registros en tablas maestras
ACTORES	Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Administrador'. El administrador ha listado una tabla maestra
CASOS DE USO RELACIONADOS	CU-TFC1, CU-TFC7
ESCENARIO	• El administrador selecciona un registro de la lista. • El administrador pulsa el botón de eliminar registro. • El Administrador valida la baja del registro.
ESCENARIO ALTERNATIVO	El sistema no valida la acción y no se produce la baja.
RESULTADO	Se elimina un registro de la tabla maestra.
POSTCONDICIONES	Se elimina un registro en la tabla de base de datos.

Modificación en tablas maestras

CASO DE USO: Modificación en tablas maestras	
IDENTIFICACIÓN	CU-TFC10
DESCRIPCIÓN	Modificación de registros en tablas maestras
ACTORES	Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Administrador'. El administrador ha listado una tabla maestra y pulsado el botón de modificar registro.
CASOS DE USO RELACIONADOS	CU-TFC1, CU-TFC7
ESCENARIO	• El administrador accede a la pantalla de modificación de alguna de las tablas maestras. • El administrador modifica los datos del registro. • El Administrador valida el cambio realizado.
ESCENARIO ALTERNATIVO	El sistema no valida la acción y no se produce la modificación.
RESULTADO	Se actualiza un registro de la tabla maestra.
POSTCONDICIONES	Se actualiza un registro en la tabla de base de datos.

Actualiza estado de pedidos

CASO DE USO: Actualiza estado pedidos	
IDENTIFICACIÓN	CU-TFC11
DESCRIPCIÓN	Actualización del estado de los pedidos
ACTORES	Administrador
PRECONDICIONES	El usuario debe estar identificado con perfil 'Administrador'. El administrador ha listado los pedidos y pulsado el botón de modificar registro.
CASOS DE USO RELACIONADOS	CU-TFC1, CU-TFC5
ESCENARIO	• El administrador accede a la pantalla de modificación de pedidos. • El administrador actualiza el estado del pedido. • El Administrador valida el cambio realizado.
ESCENARIO ALTERNATIVO	El sistema no valida la acción y no se produce la modificación.
RESULTADO	Se actualiza el estado de un pedido.
POSTCONDICIONES	Se actualiza el estado de un pedido en la base de datos.

2.5 Prototipado.

Como parte final de la fase de diseño se ha realizado la construcción de un prototipo donde se muestren de forma visual el aspecto de la aplicación y el diseño de las pantallas donde se encuentran las principales funcionalidades. Las pantallas que muestren funcionalidades repetidas se han omitido dentro de esta muestra.

Dado que como veremos más adelante, las funcionalidades de la aplicación se han completado al 100% al termino del proyecto, las siguientes figuras corresponden a capturas directamente tomadas de la propia aplicación.

Pantalla de acceso a la aplicación:

Figura 6. Acceso a la aplicación.

Pantalla de listado de tablas maestras:

Figura 7. Listado de tablas.

Pantalla de alta de registros:

Figura 8. Nuevo registro.

Pantalla de modificación de registros:

Figura 9. Modificación de registro.

Pantalla de listado de artículos:

Figura 10. Listado de artículos.

Pantalla de confirmación de compra:

Figura 11. Proceso de compra.

Pantalla de finalización de compra:

Figura 12. Finalización de compra.

Pantalla de detalle de pedidos:

Figura 13. Detalle de pedido.

Capitulo 3. Diseño.

3.1 Diagrama de clases.

A continuación se muestra el diagrama de clases de la aplicación. El diagrama de clases nos permite representar las principales entidades del sistema, así como las clases y las relaciones existentes entre ellas.

En el diagrama se muestra también los atributos de las entidades junto con el tipo de dato al que han sido instanciados.

Nótese como en el caso de clases relacionadas que contienen atributos que son objetos de otra clase, estos no aparecen incluidos en descripción de la entidad sino que están ubicados en la propia relación entre clases.

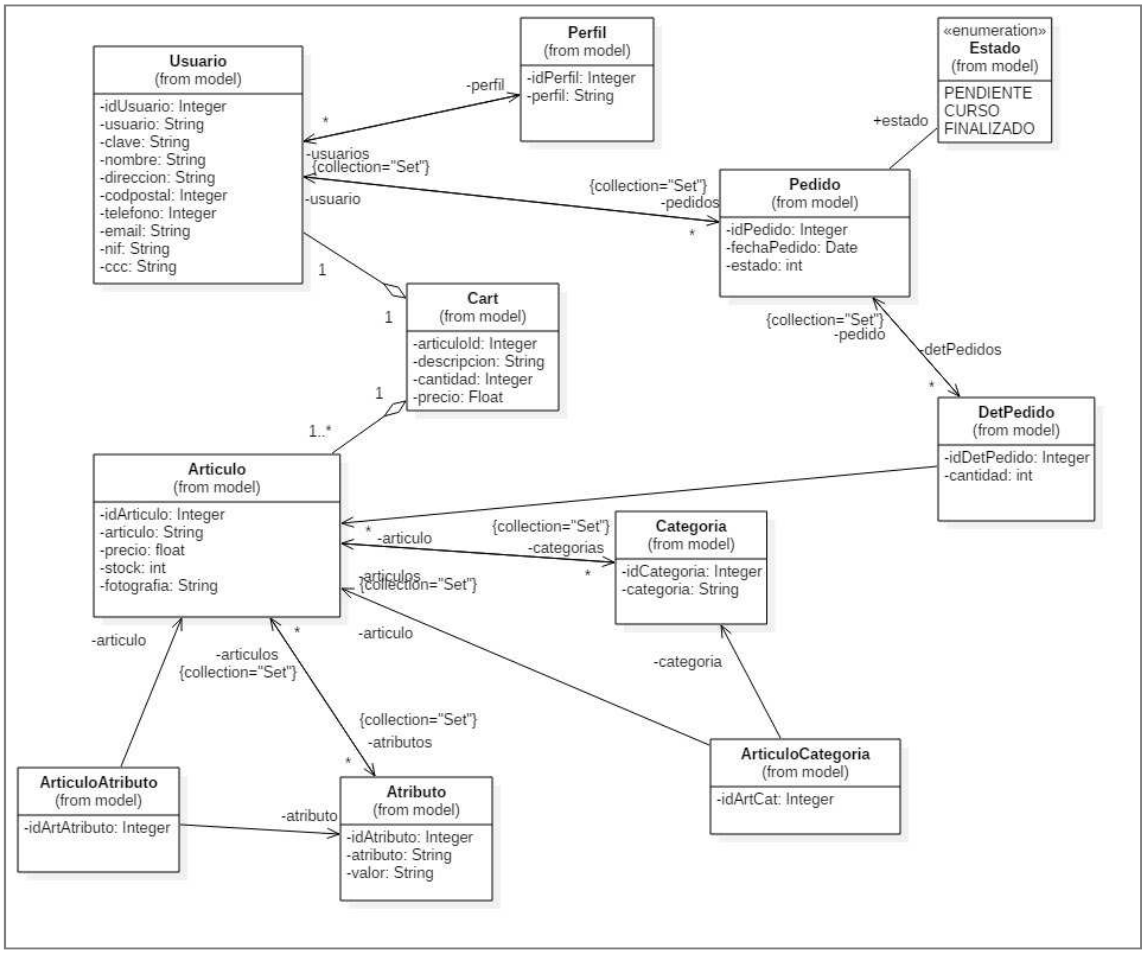


Figura 14. Diagrama de clases

3.2 Diagramas de secuencia.

En el capítulo anterior se incluyó una relación de casos de uso donde se detallaban de forma los escenarios posibles que describían las funcionalidades de la aplicación.

Mediante los diagramas de secuencia, podemos describir de forma visual, paso a paso la interacción entre los distintos elementos y actores a través de una línea de tiempo.

Dado que en ocasiones los escenarios son similares, se repiten o aportan poco valor, se han realizado tres diagramas de secuencia, uno por subsistema de la aplicación, donde se muestran los escenarios más significativos y que cubren las principales funcionalidades de dicho subsistema.

Autentificación de usuarios (Subsistema de Usuarios)

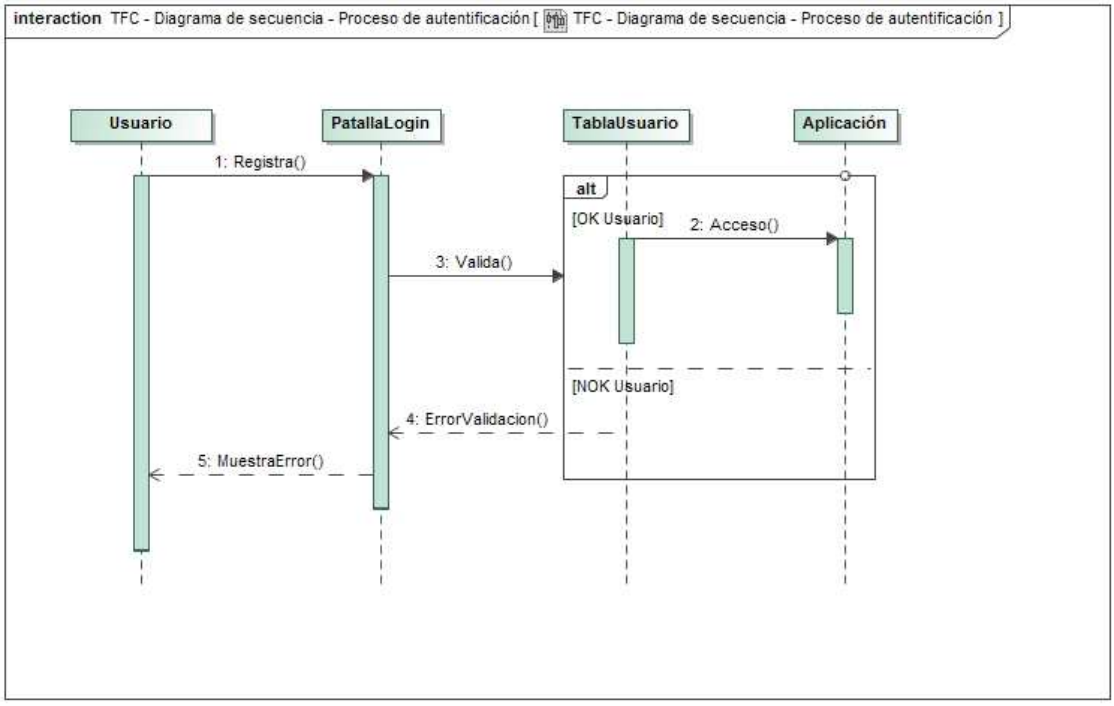


Figura 15. Diagrama de autentificación de usuarios.

Mantenimiento de tablas (Subsistema de Administración)

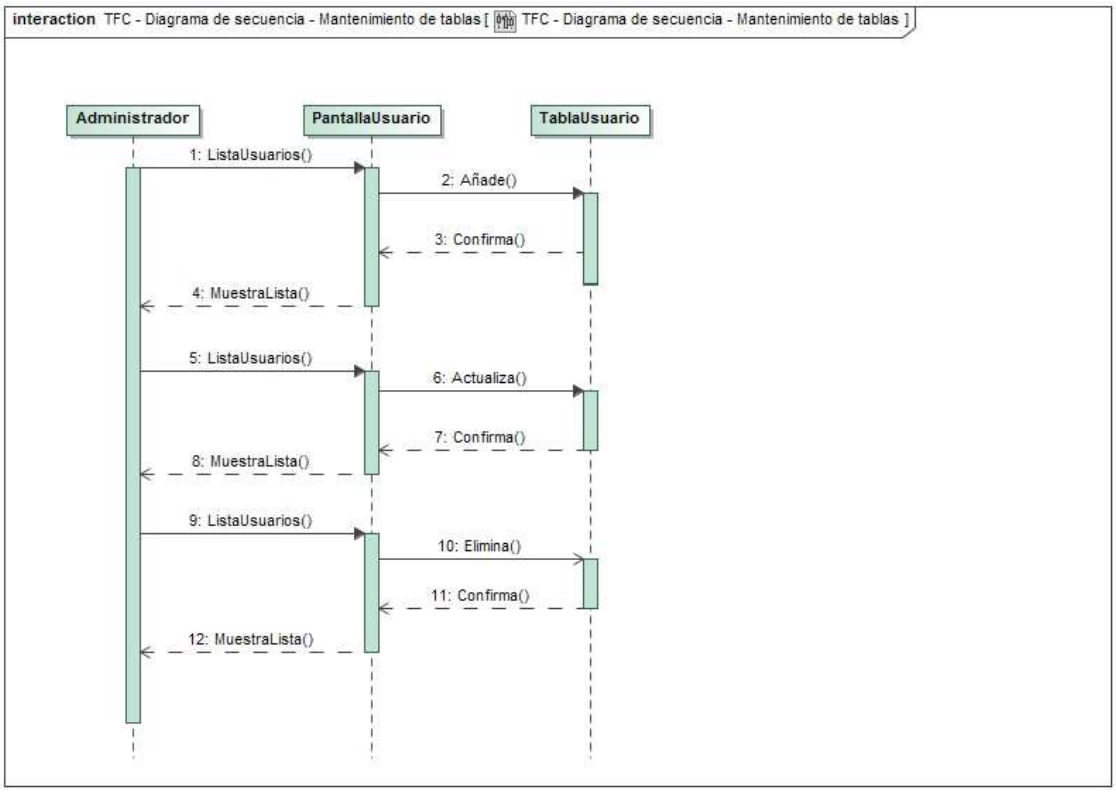


Figura 16. Diagrama de mantenimiento de tablas.

Proceso de venta (Subsistema de Venta)

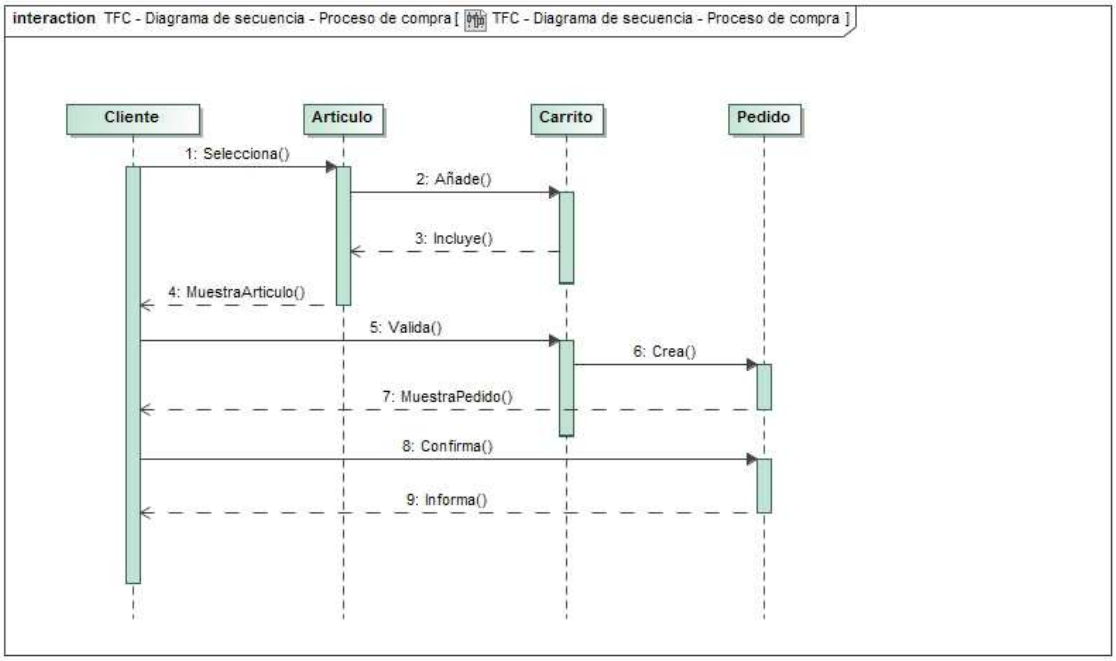


Figura 17. Diagrama del proceso de venta.

3.3 Modelo de datos

El diseño del modelo de datos con las tablas que componen la aplicación, los atributos de las tablas y las relaciones establecidas entre estas, son descritas en el siguiente diagrama de entidad/relación:

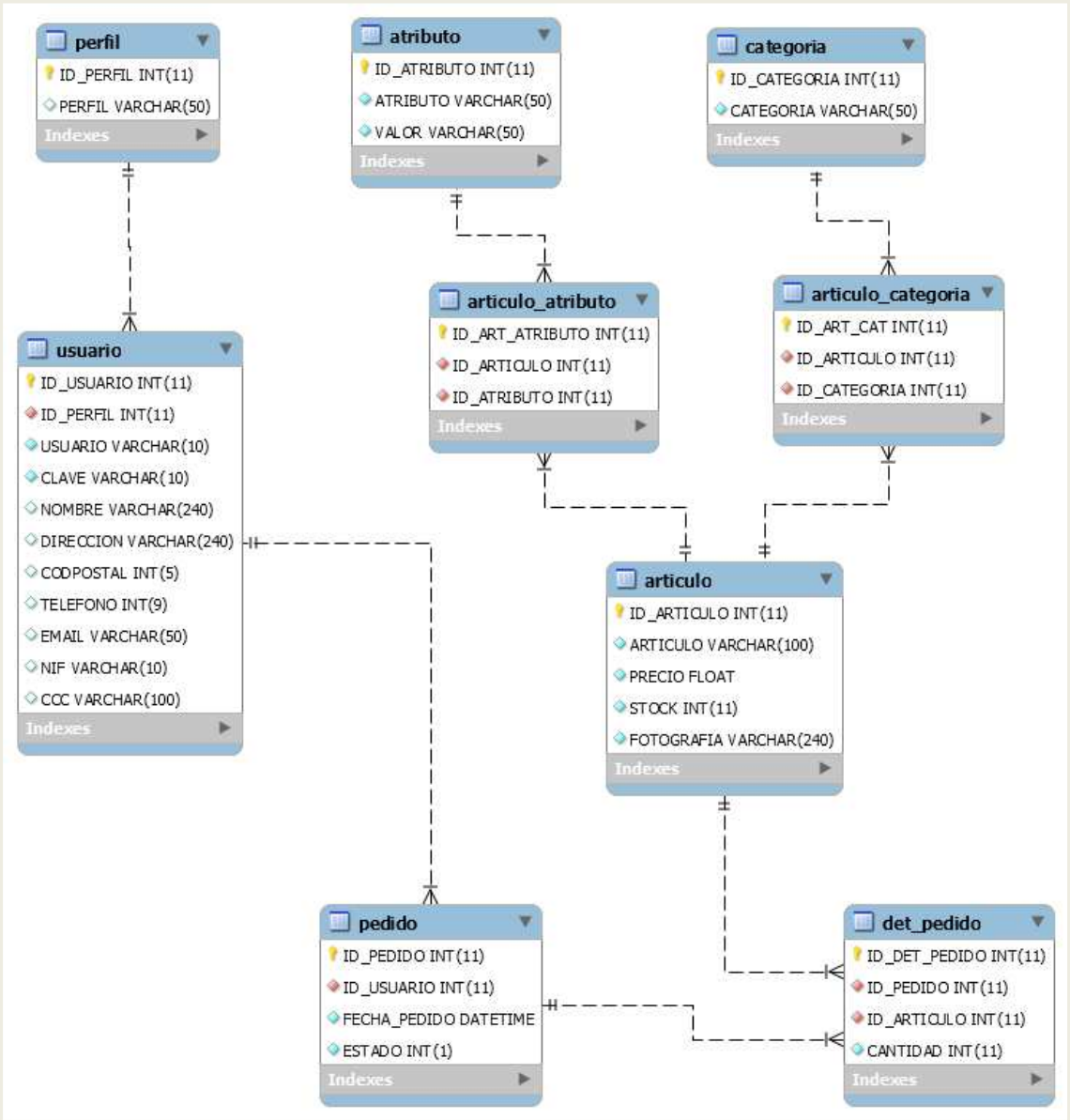


Figura 18. Diagrama modelo de datos E/R.

3.3.1 Descripción de tablas

Tabla USUARIO

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción

ID_USUARIO	NO	int	11	PK	Id. de usuario
ID_PERFIL	NO	int	11	FK	Id. de perfil
USUARIO	NO	varchar	10		Usuario de acceso
CLAVE	NO	varchar	10		Clave de acceso
NOMBRE	YES	varchar	240		Nombre y apellidos del usuario
DIRECCION	YES	varchar	240		Dirección del usuario
CODPOSTAL	YES	int	5		Código Postal
TELEFONO	YES	int	9		Teléfono
EMAIL	YES	varchar	50		Correo electrónico
NIF	YES	varchar	10		NIF
CCC	YES	varchar	100		Código Cuenta Corriente

Tabla PERFIL

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_PERFIL	NO	int	11	PK	Id. de perfil
PERFIL	NO	varchar	50		Perfil

Tabla ARTICULO

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_ARTICULO	NO	int	11	PK	Id. articulo
ARTICULO	NO	varchar	100		Articulo
PRECIO	NO	float	11		Precio
STOCK	NO	int	11		Cantidad en stock
FOTOGRAFIA	NO	varchar	240		URL de foto

Tabla CATEGORIA

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_CATEGORIA	NO	int	11	PK	Id. categoría
CATEGORIA	NO	varchar	50		Categoría

Tabla ATRIBUTO

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_ATRIBUTO	NO	int	11	PK	Id. de atributo

ATRIBUTO	NO	varchar	50		Atributo
VALOR	NO	varchar	50		Valor de atributo

Tabla ARTICULO_CATEGORIA

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_ART_CAT	NO	int	11	PK	ID. de articulo / categoría
ID_ARTICULO	NO	int	11	FK	Id. de articulo
ID_CATEGORIA	NO	int	11	FK	Id. de categoría

Tabla ARTICULO_ATRIBUTO

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_ART_ATRIBUTO	NO	int	11	PK	ID. de articulo /atributo
ID_ARTICULO	NO	int	11	FK	Id. de articulo
ID_ATRIBUTO	NO	int	11	FK	Id. de atributo

Tabla PEDIDO

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_PEDIDO	NO	int	11	PK	Id. de pedido
ID_USUARIO	NO	int	11	FK	Id. de usuario
FECHA_PEDIDO	NO	datetime			Fecha de pedido
ESTADO	NO	int	1		Estado del pedido

TABLA DET_PEDIDO

Campo	Nulo	Tipo de dato	Longitud	Key	Descripción
ID_DET_PEDIDO	NO	int	11	PK	Id. detalle de pedido
ID_PEDIDO	NO	int	11	FK	Id. del pedido
ID_ARTICULO	NO	int	11	FK	Id. del articulo
CANTIDAD	NO	int	11		Cantidad

3.4 Arquitectura

Uno de los objetivos del Trabajo Fin de Carrera es la realización de un proyecto basado en la plataforma J2EE de Java. Dado que dicha plataforma comprende un muy

extenso conjunto de componentes, servicios y marcos de desarrollo relacionados, es necesario que se realice una descripción, breve en todo caso, de las tecnologías que se han querido utilizar en la implementación del proyecto.

3.4.1 Plataforma JAVA EE

J2EE o Java 2 Platform Enterprise Edition (Java EE, Java Enterprise Edition a partir de la versión 1.4**)** es una plataforma de Java diseñada para el desarrollo de aplicaciones empresariales distribuidas, basada en una arquitectura multicapa, implementada en el lenguaje de programación Java y que se ejecuta en un servidor de aplicaciones.

La plataforma J2EE comprende un conjunto de servicios, APIs y protocolos que proporcionan la funcionalidad para desarrollar aplicaciones de varios niveles, basados en la web. Algunas de las características y servicios principales en la arquitectura de J2EE son:

- **El nivel de cliente**. Este consiste en clientes de aplicaciones que tienen acceso a un servidor Java EE y que se encuentran normalmente en un equipo diferente del servidor. Los clientes pueden ser un navegador web, una aplicación independiente de Java u otros servidores que se ejecuta en una máquina diferente a la del servidor Java EE.
- **En la capa web** se gestiona y da soporte a los componentes que implementan este nivel como Servlets, JSP (Java Server Pages) o JSF (Java Server Faces).
- **En el nivel de negocio**, EJB (Enterprise JavaBeans) proporciona otra capa donde se almacena la lógica de la plataforma. EJB es una tecnología basada en servidor que ofrece funciones tales como concurrencia, seguridad y gestión de memoria.
- A **nivel de datos**, se utilizan componentes tales como JDBC y JPA. JDBC (Java Database Connectivity) es la interfaz estándar para bases de datos Java, el equivalente Java para ODBC. JPA (Java Persistence API) es el API de persistencia de la plataforma Java EE basado en POJO's (Plain Old Java Object) para mapear bases de datos relacionales.

3.4.2 Spring MVC

Spring MVC es un framework que proporciona una arquitectura basada en el patrón MVC (Model View Controller) para el desarrollo de aplicaciones WEB flexibles. El patrón MVC separa los diferentes aspectos de la aplicación (lógica de entrada de información, lógica de negocio e interfaz de usuario) mientras que proporciona simultáneamente un acoplamiento entre ellos:

- El **Modelo** encapsula los datos de la aplicación, por lo general son los POJOs.
- La **Vista** es responsable de la presentación de los datos del modelo, generando una salida en HTML que el navegador sepa interpretar.
- El **Controlador** se encarga de procesar las peticiones del usuario y la construcción de modelo apropiado y lo pasa a la vista para su representación.

El modelo-vista-controlador Web de Spring está diseñado en torno a un *DispatcherServlet* que maneja todas las peticiones y respuestas HTTP:

- Después de recibir una petición HTTP, *DispatcherServlet* consulta el *HandlerMapping* para llamar al controlador adecuado.
- El controlador recoge la solicitud y llama a los métodos de servicios apropiados basados en métodos GET o POST. El método de servicio establecerá los datos del modelo basado en la lógica de negocio definido y retornará el nombre de la vista al *DispatcherServlet* .
- El *DispatcherServlet* contará con la ayuda del *ViewResolver* para recoger la vista definida para la solicitud.
- Tras esto, el *DispatcherServlet* pasa los datos del modelo a la vista que finalmente se muestra en el navegador.

Todos los componentes antes mencionados, *HandlerMapping* , *Controller* y *ViewResolver* son partes de *WebApplicationContext* que es una extensión del *ApplicationContext* plano con algunas características adicionales necesarias para las aplicaciones web .

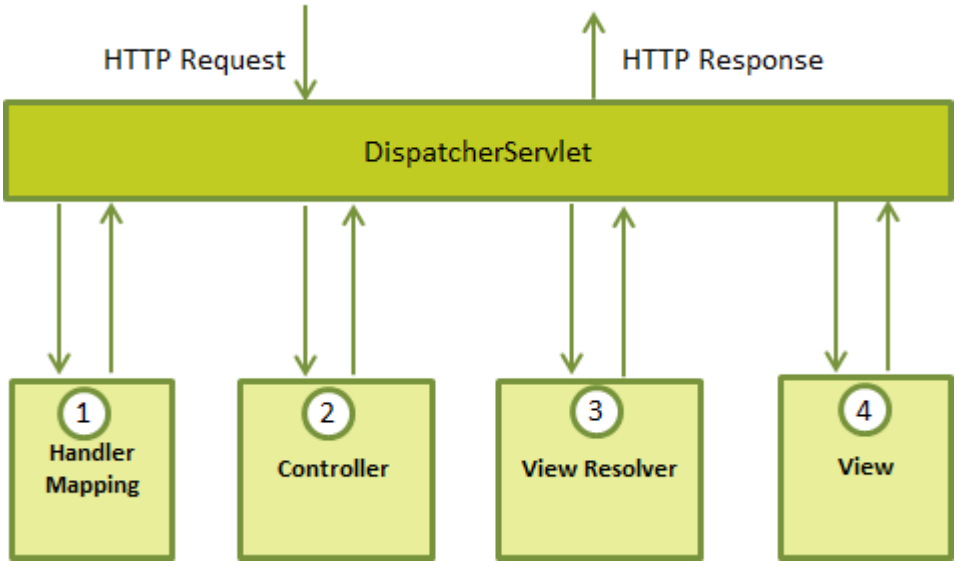


Figura 19. Esquema arquitectura Spring MVC

3.4.3 Hibernate

Hibernate es una solución ORM (Object-Relational Mapping) para Java que aparece como un framework de código abierto proporcionando servicio de alto rendimiento en consultas y persistencia facilitando el mapeo de atributos entres las BBDD relacionales y el modelo de datos de las aplicación.

Hibernate mapea clases Java a tablas de bases de datos y tipos de datos Java a tipos de datos SQL proporcionando a los desarrolladores la eliminación de un alto porcentaje de las tareas comúnmente relacionadas con la persistencia de datos.

Hibernate se ubica entre los objetos Java y el servidor de base de datos para gestionar todo el trabajo de persistencia en función de los mecanismos y patrones de relación en que estos objetos están basados.

Figura 20. Esquema arquitectura Hibernate

Entre las principales ventajas del uso de Hibernate encontramos que :

- Hibernate se encarga de mapear las clases de Java a tablas de base de datos mediante archivos XML o anotaciones y sin necesidad de código.
- Proporciona APIs sencillas para almacenar y recuperar objetos Java directamente hacia y desde la base de datos.
- Ante cambios en base de datos o en cualquier tabla solamente se necesita cambiar las propiedades del archivo XML.
- No requiere un servidor de aplicaciones para funcionar.
- Manipula asociaciones complejas de objetos de la base de datos.
- Reducir al mínimo el acceso a base de datos con las estrategias de búsqueda inteligentes.
- Proporciona consultas simples de datos.
- Soporta la gran mayoría de las bases de datos más importantes como Oracle, Informix, MySql, PostgreSQL o Microsoft SQL Server.

3.4.4 JSP (Java ServerPages)

Java Server Pages (**JSP**) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML.

Las JSP's permiten la utilización de código Java mediante scripts. Además es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Librerías de Etiquetas (Tag Libs o Tag Libraries) externas e incluso personalizadas. En el caso de la aplicación Uniformes escolares se han utilizado en todos los casos etiquetas y Tag Libs. En ningún caso se han incluido en las páginas scripts embebidos en el código HTML ya que actualmente esta es una práctica desaconsejada principalmente por motivos de seguridad.

El funcionamiento general de las páginas JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se presentará en la pantalla del navegador del usuario.

Entre las ventajas a destacar caben las siguientes:

- El rendimiento de una página JSP es el mismo que tendría el servidor equivalente, ya que el código es compilado como cualquier otra clase Java.
- La JVM (Java Virtual Machine o Maquina Virtual Java) compila dinámicamente a código de máquina las partes de la aplicación que lo requieran.
- Es más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.
- JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios.

3.4.5 Otras tecnologías

Además de las ya comentadas anteriormente se han usado para la implementación del proyecto las siguientes herramientas y tecnologías:

- **Eclipse IDE**: Entorno integrado de desarrollo de aplicaciones Java. En mi caso he utilizado para el desarrollo la versión 4.5.1 (Mars).
- **Maven**: Herramienta de software para la gestión y construcción de proyectos Java. En el caso de mi aplicación web he utilizado esta tecnología como apoyo para configurar y obtener las librerías y dependencias usadas en la aplicación.

- **MagicDraw**: Herramienta de diseño de diagramas UML. Se ha utilizado para la realización de diagramas la versión 18.0 bajo licencia de uso proporcionada por la UOC:
- **MySQL**: Sistema de gestión de bases de datos relacional. Se ha utilizado para la creación de tablas y scripts SQL.

- **MySql Workbench**: Software complementario de MySQL usado como herramienta visual de diseño y gestión de de bases de datos.

Capitulo 4. Implementación.

Tras finalizar la fase de implementación de la aplicación web descrita y diseñada en los capítulos anteriores, en los siguientes apartados se indicará el software utilizado para poder ejecutar la aplicación así como las configuraciones que sean necesarias para su correcto funcionamiento.

Además se intentará explicar los motivos y razones que se han seguido y las decisiones que se han tomado en cuanto al diseño y la implementación de la aplicación y finalmente se hará un resumen del porcentaje de implementación alcanzado para cubrir las funcionalidades de los casos de pruebas que se especificaron en el capítulo 2

4.1 Requerimientos, Software y Configuración

A continuación se detalla los requerimientos de software para arrancar con éxito la aplicación. Únicamente se describirán los software o tecnologías necesarias así como los posibles detalles de configuración en los casos que sean necesarios.

4.1.1 Requerimientos de máquina

Aquí se expondrá la información acerca de la máquina PC en que se ha desarrollado y ejecutado la aplicación. No necesariamente se debe consideran esta configuración como mínima pero si es recomendable que sea similar o superior.

- **Sistema operativo:** Windows 7 Professional Edition (64 bits)
- **Procesador:** Intel-Core i5 1.70Ghz
- **RAM:** 8GB

Comentar que aparte de la configuración anterior también se ha trabajado sin problema en una máquina Windows 10 Home Edition 64bits i7 2.20GHz 8GB RAM.

4.1.2 Configuración JAVA

Para el desarrollo y la ejecución en local de cualquier aplicación basada en Java es necesario tener instalada en la máquina la plataforma JAVA.

En nuestro caso se ha instalado la versión JDK 7 (Java SE Development Kit 7u80) de Java. Se puede obtener de forma gratuita para distintos sistemas operativos en el siguiente enlace:

http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html

4.1.3 Base de datos

El servidor de base de datos utilizado es **MySQL versión 5.6 Comunnity**. Se puede descargar de forma gratuita desde la página web de MySQL en el siguiente enlace:

http://dev.mysql.com/downloads/mysql/

Configuración: Para la correcta ejecución de la aplicación web, es necesario tener arrancado el servidor de bases de dato MySQL. Los parámetros de conexión que se han configurado son los siguientes:

- **Nombre de máquina o IP:** En caso de que ejecute en la máquina local será localhost o 127.0.0.1. en caso de que el servidor esté en un máquina remota será el nombre de host de dicha máquina o su dirección IP.
- **Puerto:** por defecto usaremos el 3306
- **Usuario**: tfcuoc
- **contraseña**: tfcuoc
- **Nombre de esquema:** tiendavirtual

La instalación del servidor de base de datos se puede realizar con la configuración por defecto ya que la creación del usuario/contraseña, el esquema datos, las tablas y relaciones así como un juego de datos de prueba, se proporcionaran en un único archivo ".sql" que se adjuntará con el paquete de software entregado dentro de la carpeta "database".

Los datos de configuración indicados anteriormente son los usados para conectar la aplicación a la base de datos. Si dicha configuración cambiara habría que modificar los siguientes ficheros de la aplicación:

/META-INF/context.xml :

```
<Context>
    <Resource name="jdbc/tiendavirtual_DB" auth="Container"
            type="javax.sql.DataSource"
driverClassName="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/tiendavirtual"
            username="tfcuoc" password="tfcuoc" maxActive="30" maxIdle="10"/>
</Context>
```

/WEB-INF/mvc-dispatcher-servlet.xml :

```
<bean id="dataSource"

      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
```

```
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/tiendavirtual"
/>             <property name="username" value="tfcuoc" />
        <property name="password" value="tfcuoc" />
</bean>
```

Finalmente y aunque no son imprescindibles para el funcionamiento de la aplicación, como dato adicional indicar que para la gestión de la base de datos se ha utilizado estos softwares:

- **MySQL Workbench 6.3:** http://dev.mysql.com/downloads/workbench/
- **HeidiSQL 9.3:** http://www.heidisql.com/download.php

4.1.4 Servidor Web Apache Tomcat

Primeramente comentar que Apache Tomcat no es un servidor de aplicaciones sino que es un contenedor web con soporte de servlets y JSPs que también puede actuar como servidor web.

La versión utilizada del servidor es Apache Tomcat 7.0.65 Server. Esta versión se puede descargar de modo gratuito del siguiente enlace:

http://tomcat.apache.org/download-70.cgi

Para ejecutar correctamente Tomcat es necesario tener previamente instalada en el PC la máquina virtual Java.

Configuración: Para el despliegue y ejecución de la aplicación no es necesario realizar ninguna configuración especial en Tomcat más allá de las especificadas durante la instalación. Sin embargo si es recomendable indicar algunas modificaciones en la configuración que pueden realizarse. Para ello tendremos que actualizar dos archivos ubicados en el directorio 'conf' de Apache Tomcat:

server.xml: En este fichero se pueden definir un puerto distinto al puerto que usa Tomcat por defecto, el 8080. en la ejecución local de la aplicación se ha cambiado el puerto al 8089. La modificación en el ficheros sería así:

...

```
<Connector port="8089" protocol="HTTP/1.1" connectionTimeout="20000"
redirectPort="8443" />
```

tomcat-users.xml: En este fichero es posible configurar los roles y los usuarios de Tomcat. En nuestra aplicación se han configurado del siguiente modo:

...

```
<tomcat-users>
```

```
      <role rolename="manager-gui"/>
      <role rolename="admin-gui"/>
      <user username="tomcat" password="tomcat" roles="manager-gui"/>
      <user username="admin" password="admin" roles="admin-gui"/>
</tomcat-users>
```

Despliegue y arranque de aplicación. Para desplegar la aplicación accedemos a la consola de Tomcat a través de la siguiente URL:

http://localhost:8080/

Si se ha modificado el puerto o el nombre del host es distinto hay que modificarlo. Una vez dentro de la consola accedemos a la gestión de aplicaciones a través del botón 'Manager app'. Nos pedirá usuario y contraseña, hay que introducir la contraseña de usuario definida anteriormente con rol "manager-gui".

Dentro de la consola podemos seleccionar el archivo WAR del aplicación que queremos desplegar. Una vez desplegado podemos acceder a la aplicación mediante el enlace o escribiendo en el navegador:

http://localhost:8080/UniformesOnlineUOC_TFC/

IMPORTANTE: En las pruebas realizadas al desplegar la aplicación en Tomcat se ha detectado un problema tras subir y desplegar por primera vez el WAR de la aplicación. En este escenario la aplicación arranca y aparece la pantalla de login pero inmediatamente después la aplicación falla. La solución es, después de subir el WAR y desplegarlo, parar el servidor y volver a arrancarlo.

4.2 Decisiones de diseño e implementación

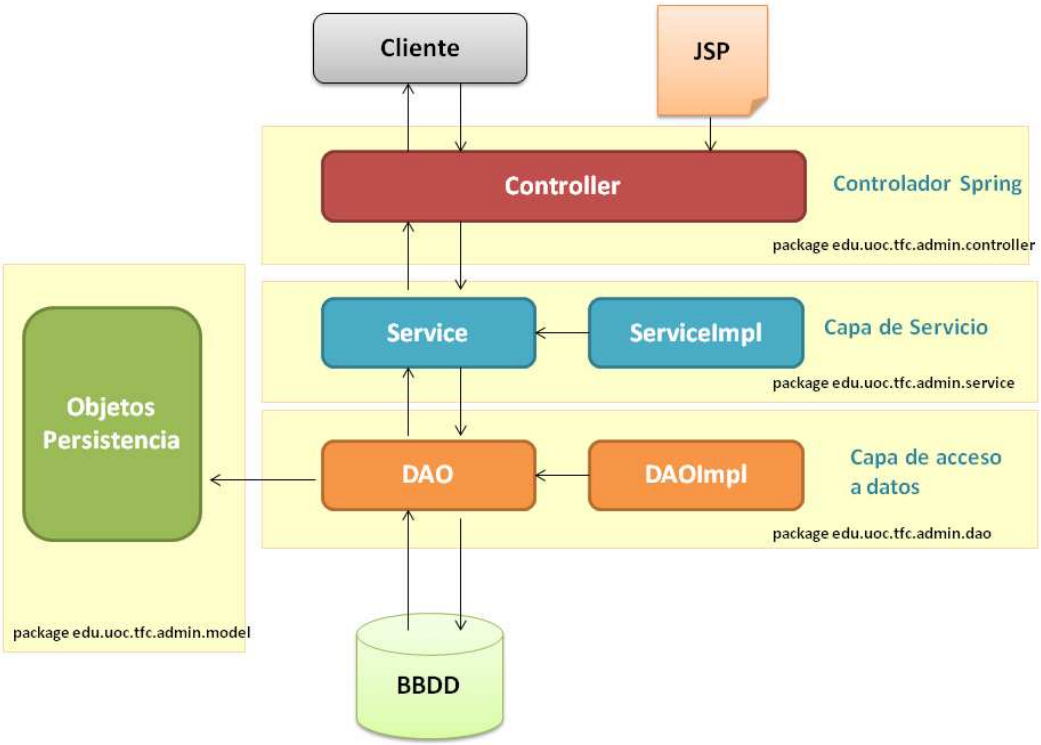


Figura 21. Arquitectura MVC en Uniformes Escolares

La entidades de persistencia han sido creadas desde las tablas de la base de datos. Las relaciones entre entidades se ha definidos mediante anotaciones de **Hibernate y JPA**. Los accesos a los datos de las tablas se han realizado mediante comandos "Criteria" de Hibernate (en un caso puntual, por motivos de eficiencia de código, se ha realizado el acceso a datos mediante una query HQL).

En la pate de **Spring MVC** se ha utilizado la inyección de dependencias declarando los correspondientes 'beans' en los ficheros xml de la aplicación o mediante el uso de anotaciones de Spring, como por ejemplo *@Autowired.*

El control de acceso y la seguridad de la aplicación se ha creado con **Spring Security** que permite interceptar y autenticar al usuario que accede contra la base de daros y gestiona, según el perfil de acceso, la redirección dentro de la aplicación o la comunicación de error en caso de no ser autenticado.

Finalmente para la presentación de páginas .jsp, se utilizado la estructura de **Apache Tiles** que integrado con Spring, permite crear páginas web con elementos estáticos como la cabecera, la barra de navegación y el pié de página junto con un contenedor principal donde se muestras las páginas.

4.3 Objetivos alcanzados de implementación

En la siguiente tabla se ha detallado el grado porcentual de implementación de cada uno de los casos de uso definidos para la aplicación en la fase de diseño.

Caso de uso	Descripción	% Completado	Comentarios
Acceso al sistema	El usuario introduce identificador de usuario y contraseña. El sistema navegara a la pantalla que corresponda a la funcionalidad del usuario.	**100%**	Re direcciona al cliente según su perfil registrado en BBDD. Si el usuario/contraseña son incorrectos muestra mensaje de error.
Consulta de productos	Al acceder a la primera pantalla de la tienda se muestra un listado de productos Al seleccionar una categoría se mostrará el listado de productos	**100%**	Se muestran en un menú lateral todas las categorías. La pantalla principal muestra todos los productos, si se selecciona una categoría se filtran los productos mostrados por esta.
Añade producto a carrito	El cliente pulsa el botón 'Añadir' de un producto. El producto se añade al carrito de compra.	**100%**	El carrito se muestra con un resumen de los artículos seleccionados por el usuario manteniendo la sesión y la información durante todas las pantallas y el proceso de compra hasta su validación.
Formaliza pedido	El cliente pulsa el botón de confirmar el carrito. Se formaliza el carrito como pedido iniciándose el proceso de compra.	**100%**	Se muestra una nueva pantalla con el resumen de los artículos del carrito, el importe total del mismo y las opciones de seguir comprando o finalizar la compra.
Consulta de pedidos	El usuario accede a la pantalla de consulta de pedidos Al usuario Cliente solo se le mostrarán sus propios pedidos. El usuario Administrador podrá visualizar a los pedidos de todos los clientes.	**100%**	El usuario puede acceder a todos sus pedidos realizados cuando finaliza su proceso de compra o acceder en todo momento desde el menú lateral de la tienda.
Pago de pedido	El cliente accede a la pantalla de pago Se introducen los datos de pago en el TPV virtual. Se devuelve la confirmación de compra Se muestra un resumen del	**100%**	En la PEC2 se indicó que se incluía en este caso prueba el pago a través de TPV para simular un proceso de compra online real. Dado que este tipo de componentes suelen

	pedido realizado.		ser componentes proporcionados por empresas o por las entidades bancarias directamente y suelen ser ajenos a la propia tienda online no se consideró oportuno incluirlo para el TFC. En la aplicación se ha implementado la finalización del carrito mostrando el detalle del pedido final realizado y un mensaje de fin de compra.
Consulta de tablas maestras	El administrador accede a la pantalla de consulta de alguna de las tablas maestras Se muestra la lista de las tablas El Administrador tendrá la opción de crear, modificar o eliminar registros de las tablas.	**100%**	El administrador puede consultar los listados de Usuarios, Artículos, Categorías, Atributos y Pedidos. Se le muestran en todas las pantallas opciones para crear un nuevo elemento y modificar o eliminar alguno de los ya existentes.
Alta en tablas maestras	El administrador accede a la pantalla de alta de alguna de las tablas maestras El administrador rellena el formulario con los datos requeridos. El Administrador valida el nuevo registro.	**100%**	Se accede al formulario para rellenar los datos del nuevo elemento. Al pulsar el botón, el elemento se graba validando los datos del formulario.
Baja en tablas maestras	El administrador selecciona un registro de la lista. El administrador pulsa el botón de eliminar registro. El Administrador valida la baja del registro.	**100%**	Cada registro de la tabla tiene un botón para eliminar dicho registro.
Modificación en tablas maestras	El administrador accede a la pantalla de modificación de alguna de las tablas maestras. El administrador modifica los datos del registro. El Administrador valida el cambio realizado.	**100%**	Se accede al formulario para modificar los datos del elemento seleccionado. Al pulsar el botón, el elemento se graba validando los datos del formulario
Actualiza estado pedidos	El administrador accede a la pantalla de modificación de pedidos.	**100%**	El administrador puede actualizar desde el detalle de un pedido el

	El administrador actualiza el estado del pedido. El Administrador valida el cambio realizado.		estado de este.
% Total implementado =		**100%**	

Capitulo 5. Coste económico.

En las circunstancias en que se ha desarrollado la aplicación es difícil determinar un coste económico real para la misma. Hay que tener en cuenta las siguientes premisas que se han seguido en el proyecto:

- Software gratuito. Todas las herramientas usadas son gratuitas, por lo que no se ha empleado ningún coste en licencias para el uso de las mismas.
- Desarrollo personal. Evidentemente y dado que es un trabajo académico personal, no se ha contratado ningún otro desarrollador o empresa de software en la realización del proyecto.
- Publicación local. En el caso de la publicación de la aplicación para su visualización publica en Internet, no se ha contratado ni usado ningún servicio de hosting ni contratado ningún dominio sino que se ha utilizado mi propia máquina personal como servidor web.

En cualquier caso siempre es interesante y necesario desde una visión comercial el evaluar este aspecto por lo que voy a intentar dar unos datos de costes aproximados en el caso de que una empresa, en este caso el colegio que vende los uniformes de *Uniformes Escolares*, quisiera hacer realidad el proyecto.

Concepto	Coste (apróx.)
Dominios y Hosting (Servidor dedicado)	35-125 €/mes
Desarrollo de la aplicación	1000-6000 €
Mantenimiento y soporte	100-600 €/año
TPV Virtual	1-3 % comisión por venta
Pasarela de pago	50-200 €
Módulos	10-500 €
Certificados seguridad SSL	7-85 €/año
Protección de datos (LOPD)	50-200 €

En definitiva y a grandes rasgos según los datos de la anterior tabla para poner en marcha nuestra aplicación necesitaríamos realizar una inversión inicial, haciendo una media de los precios mostrados, de unos **5.000€** aproximadamente. A esto habría que sumar un coste medio de otros **150€** por mes.

Capitulo 6. Conclusiones.

Al principio del proyecto comentaba en la primera PEC las motivaciones y expectativas que tenía a la hora de el Trabajo Fin de Carrera. Ahora que estoy finalizando la asignatura creo que en general si se han cumplido tanto a nivel personal como en consecución de los objetivos del proyecto.

6.1 Conclusiones personales

Particularmente para mí ha supuesto una experiencia interesantísima ya que probablemente desde mi puesto de vista el TFC sea la mejor asignatura de la carrera. Verdaderamente esta asignatura logra que pongamos en práctica los conocimientos y técnicas adquiridas al cursar las otras asignaturas de la carrera, en especial las relacionadas con la programación Java y la gestión de proyectos.

También ha sido muy satisfactorio a nivel personal el reto de sacar adelante un proyecto de tipo empresarial de cierta importancia, aplicando los conceptos de la ingeniería del software desde el principio hasta el final.

Otro punto importante a destacar es el uso y aprendizaje de nuevas herramientas y tecnologías. Ha costado en algunos casos ya que el aprendizaje ha llevado su tiempo y en muchos casos este no se completaba sino durante la propia implementación a base de muchos fallos e intentos. Una vez superadas estas barreras y con el conocimiento y la experiencia adquirida, el tramo final del proyecto ha sido mucho más sencillo.

6.2 Conclusiones de la aplicación

Cuando tuve que elegir área para la realización del TFC, prácticamente no dude en elegir J2EE. Tanto profesionalmente como durante la trayectoria de la carrera ya conocía y tenía experiencia en el desarrollo de aplicaciones con esta plataforma así como con algunos frameworks de trabajo relacionados.

Creo que la aplicación como resultado presentado y descrito en este documento cumple con lo esperado en los requerimientos del TFC como aplicación web. Las principales funcionalidades de cada módulo han sido satisfechas y la aplicación es completamente operativa.

En cuanto a las tecnologías utilizadas comentar que en general me he sentido cómodo trabajando con ellas aunque en el caso de Hibernate y SpringMVC ha sido más complicado dado la complejidad y amplitud de ambos frameworks aunque también es cierto que existe infinidad de documentación y ejemplos de ayuda para afrontarlos.

En el caso de Hibernate, ya había trabajado con este framework. Me parece muy interesante la propiedad que le hace especial de separar la persistencia de los propios datos con lo que se hace trasparente para el programador la base de datos que se utilice. Eso sí, tengo que decir que a la hora de realizar consultas sobre la base de

datos de cierta complejidad, me ha parecido extremadamente complicado. Aunque no he trabajado profesionalmente con Hibernate, en mi experiencia, en la mayoría de aplicaciones de este tipo las 'queries' se han hecho directamente con lenguaje SQL. Es cierto que se condiciona a la aplicación la rigidez de un lenguaje y un gestor de bases de datos específico pero se gana en potencia a la hora de resolver consultas complejas de datos.

Con Spring MVC el trabajo no ha resultado especialmente complicado, ya que su arquitectura hace que las clases controladoras se encarguen de casi todo. Me ha parecido muy interesante y útil la inyección de dependencias por la funcionalidad que facilita de pasar objetos entre distintas clases y capas de la aplicación.

Sobre la parte de interfaz de usuario, pienso que se podía haber utilizado alguna tecnología como Primefaces o Velocity para la construcción de las páginas. Sin embargo debido a que ya tenía que realizar el aprendizaje de Hibernate y Spring MVC, más necesarios y complejos, consideré que no podía arriesgar en este punto. Las páginas se han implementado con JSP usando taglibs y etiquetas junto con HTML. Añadiendo además a las página hojas de estilo CSS, se han logrado páginas bien construidas y estructuradas y visualmente atractivas.

Glosario

Actor. Especifica el rol representado por un usuario o cualquier otro sistema que interactúa con el sujeto.

Administrador. Usuario del sistema con privilegios especiales sobre el módulo de mantenimiento que está encargado de realizar la gestión de los datos maestros de la aplicación.

Bean. Es un componente de software que tiene la particularidad de encapsular varios objetos en uno

Cliente. Usuario del sistema que interactúa con el módulo de ventas de la aplicación.

Framework En español, Marco de Trabajo, son un conjunto de librerías que proporcionan soporte al desarrollo rápido de aplicaciones.

JPA. Es la API de persistencia desarrollada para la plataforma Java EE

Login. Acción de registrase en una aplicación.

MVC. Modelo Vista Controlador (Model View Controller en inglés) Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.

Persistencia. Acción de preservar la información de un objeto de forma permanente.

Prototipado: Diseño del modelo del producto final que permite visualizar el aspecto de un aplicación y efectuar pruebas sobre determinados atributos de la misma sin necesidad de que esta esté disponible.

Servlet. Clase del lenguaje de programación Java, usada para ampliar las capacidades de un servidor. Son utilizados comúnmente para extender las aplicaciones alojadas por servidores web.

Sistema. Es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, que se generan para cubrir una necesidad. En nuestro proyecto se corresponde con la aplicación *Uniformes escolares.*

Subsistema. Cada uno de los componentes funcionales del sistema.

TFC. Acrónimo de Trabajo Final de Carrera

UML: Lenguaje Unificado de Modelado (Unified Model Language en inglés). Especificación de notaciones orientada a objetos.

Usuario. Persona que se conecta al sistema y hace uso de él. En la aplicación *Uniformes Escolares*, los usuarios se dividen en Clientes y Administradores.

XML Acrónimo de Extensible Markup Language, metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

Bibliografía

Libros:

Hibernate. Persistencia de objetos en JEE. Eugenia Pérez Martínez. Editorial RA-MA.

Spring MVC: Beginner's Guide. Amuthan G. Editorial PCKT publishing.

Webs:

http://4cuatros.blogspot.com.es/2009/01/cargar-combos-con-spring-mvc-form.html
http://docs.jboss.org/hibernate/orm/5.0/quickstart/html/
http://docs.spring.io/spring/docs/current/spring-framework-reference/html/index.html
http://howtodoinjava.com/2015/02/13/spring-mvc-populate-and-validate-dropdown-example/
http://javarevolutions.com/site/tutoriales/tutoriales.jr
http://jesuslc.com/2013/03/19/poco-a-poco-con-maven-spring-hibernate/
http://simplecart.sourceforge.net/downloads.php
http://springinpractice.com/2012/01/07/making-formselect-work-nicely-using-spring-3-formatters/
http://stackoverflow.com/questions/10499641/how-to-populate-dropdown-box-in-spring-mvc
http://stackoverflow.com/questions/15526425/springmvc-formoptions-items-attribute-what-exactly-is-it-expecting
http://stackoverflow.com/questions/23895715/spring-mvc-register-custom-propertyeditor
http://websystique.com/springmvc/springmvc-hibernate-many-to-many-example-annotation-using-join-table/
http://websystique.com/spring-security/spring-security-4-custom-login-form-annotation-example/
http://www.arquitecturajava.com/spring-security-configuracion/
http://www.codejava.net/frameworks/spring/spring-4-and-hibernate-4-integration-tutorial-part-2-java-based-configuration
http://www.concretepage.com/spring/spring-mvc/spring-mvc-validator-with-initbinder-webdatabinder-registercustomeditor-example
http://www.javacodegeeks.com/2013/04/spring-mvc-hibernate-maven-crud-operations-example.html
http://www.javacodegeeks.com/2013/04/spring-mvc-session-tutorial.html

http://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial

https://gerrydevstory.com/2014/03/04/using-sessionattribute-in-spring-mvc-to-implement-a-shopping-cart/

https://github.com/armdev/JSF2.2-Spring4-Hibernate4-MySQL

https://javajeecastellano.wordpress.com/2013/12/15/spring3-mvc-hibernate4-maven/

Anexo I

En la siguiente figura se muestra el script SQL para la creación del esquema de datos de la aplicación, las tablas que lo componen, los datos maestros necesarios para su funcionamiento y un juego de datos de prueba.

Esta misma información se adjunta en un fichero .sql junto con el código fuente del programa.

```
-- --------------------------------------------------------
-- TFC - PEC2 ANALISIS Y DISENYO
-- --------------------------------------------------------

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- Volcando estructura de base de datos para tiendavirtual
DROP DATABASE IF EXISTS `tiendavirtual`;
CREATE DATABASE IF NOT EXISTS `tiendavirtual` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `tiendavirtual`;


-- Volcando estructura para tabla tiendavirtual.articulo
DROP TABLE IF EXISTS `articulo`;
CREATE TABLE IF NOT EXISTS `articulo` (
  `ID_ARTICULO` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. articulo',
  `ARTICULO` varchar(100) NOT NULL DEFAULT '0' COMMENT 'Articulo',
  `PRECIO` float NOT NULL DEFAULT '0' COMMENT 'Precio',
  `STOCK` int(11) NOT NULL DEFAULT '0' COMMENT 'Cantidad en stock',
  `FOTOGRAFIA` varchar(240) NOT NULL DEFAULT '0' COMMENT 'URL de foto',
  PRIMARY KEY (`ID_ARTICULO`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='Tabla de articulos';

-- Volcando datos para la tabla tiendavirtual.articulo: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `articulo` DISABLE KEYS */;
INSERT INTO `articulo` (`ID_ARTICULO`, `ARTICULO`, `PRECIO`, `STOCK`, `FOTOGRAFIA`) VALUES
        (1, 'ZAPATO', 39.9, 30, '/images/zapato.png');
/*!40000 ALTER TABLE `articulo` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.articulo_atributo
DROP TABLE IF EXISTS `articulo_atributo`;
CREATE TABLE IF NOT EXISTS `articulo_atributo` (
  `ID_ART_ATRIBUTO` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID. de articulo_atributo',
  `ID_ARTICULO` int(11) NOT NULL COMMENT 'Id. de articulo',
  `ID_ATRIBUTO` int(11) NOT NULL COMMENT 'Id. de atributo',
  PRIMARY KEY (`ID_ART_ATRIBUTO`),
  KEY `FK_ART_ATRIBUTO_ARTICULO_ID_ARTICULO` (`ID_ARTICULO`),
  KEY `FK_ART_ATRIBUTO_ATRIBUTO_ID_ATRIBUTO` (`ID_ATRIBUTO`),
  CONSTRAINT `FK_ART_ATRIBUTO_ARTICULO_ID_ARTICULO` FOREIGN KEY (`ID_ARTICULO`) REFERENCES `articulo`
(`ID_ARTICULO`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_ART_ATRIBUTO_ATRIBUTO_ID_ATRIBUTO` FOREIGN KEY (`ID_ATRIBUTO`) REFERENCES `atributo`
(`ID_ATRIBUTO`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 ROW_FORMAT=COMPACT COMMENT='Tabla de relacion
de articulos y categorias';

-- Volcando datos para la tabla tiendavirtual.articulo_atributo: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `articulo_atributo` DISABLE KEYS */;
INSERT INTO `articulo_atributo` (`ID_ART_ATRIBUTO`, `ID_ARTICULO`, `ID_ATRIBUTO`) VALUES
        (1, 1, 1);
/*!40000 ALTER TABLE `articulo_atributo` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.articulo_categoria
DROP TABLE IF EXISTS `articulo_categoria`;
CREATE TABLE IF NOT EXISTS `articulo_categoria` (
  `ID_ART_CAT` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID. de articulo_categoria',
  `ID_ARTICULO` int(11) NOT NULL COMMENT 'Id. de articulo',
  `ID_CATEGORIA` int(11) NOT NULL COMMENT 'Id. de categoria',
  PRIMARY KEY (`ID_ART_CAT`),
  KEY `FK_ART_CAT_ARTICULO_ID_ARTICULO` (`ID_ARTICULO`),
  KEY `FK_ART_CAT_CATEGORIA_ID_CATEGORIA` (`ID_CATEGORIA`),
  CONSTRAINT `FK_ART_CAT_ARTICULO_ID_ARTICULO` FOREIGN KEY (`ID_ARTICULO`) REFERENCES `articulo`
```

```
(`ID_ARTICULO`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FK_ART_CAT_CATEGORIA_ID_CATEGORIA` FOREIGN KEY (`ID_CATEGORIA`) REFERENCES `categoria`
(`ID_CATEGORIA`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='Tabla de relacion de articulos y
categorias';

-- Volcando datos para la tabla tiendavirtual.articulo_categoria: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `articulo_categoria` DISABLE KEYS */;
INSERT INTO `articulo_categoria` (`ID_ART_CAT`, `ID_ARTICULO`, `ID_CATEGORIA`) VALUES
        (1, 1, 2);
/*!40000 ALTER TABLE `articulo_categoria` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.atributo
DROP TABLE IF EXISTS `atributo`;
CREATE TABLE IF NOT EXISTS `atributo` (
  `ID_ATRIBUTO` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. de atributo',
  `ATRIBUTO` varchar(50) NOT NULL DEFAULT '0' COMMENT 'Atributo',
  `VALOR` varchar(50) NOT NULL DEFAULT '0' COMMENT 'Valor de atributo',
  PRIMARY KEY (`ID_ATRIBUTO`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='Tala de atributos';

-- Volcando datos para la tabla tiendavirtual.atributo: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `atributo` DISABLE KEYS */;
INSERT INTO `atributo` (`ID_ATRIBUTO`, `ATRIBUTO`, `VALOR`) VALUES
        (1, 'COLOR', 'NEGRO');
/*!40000 ALTER TABLE `atributo` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.categoria
DROP TABLE IF EXISTS `categoria`;
CREATE TABLE IF NOT EXISTS `categoria` (
  `ID_CATEGORIA` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. categoria',
  `CATEGORIA` varchar(50) NOT NULL COMMENT 'Categoria',
  PRIMARY KEY (`ID_CATEGORIA`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COMMENT='Tabla de categorias';

-- Volcando datos para la tabla tiendavirtual.categoria: ~2 rows (aproximadamente)
/*!40000 ALTER TABLE `categoria` DISABLE KEYS */;
INSERT INTO `categoria` (`ID_CATEGORIA`, `CATEGORIA`) VALUES
        (1, 'NIÑO'),
        (2, 'NIÑA');
/*!40000 ALTER TABLE `categoria` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.det_pedido
DROP TABLE IF EXISTS `det_pedido`;
CREATE TABLE IF NOT EXISTS `det_pedido` (
  `ID_DET_PEDIDO` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. detalle de pedido',
  `ID_PEDIDO` int(11) NOT NULL COMMENT 'Id. del pedido',
  `ID_ARTICULO` int(11) NOT NULL COMMENT 'Id. del articulo',
  `CANTIDAD` int(11) NOT NULL COMMENT 'Cantidad',
  PRIMARY KEY (`ID_DET_PEDIDO`),
  KEY `FK_DET_PEDIDO_ARTICULO_ID_ARTICULO` (`ID_ARTICULO`),
  KEY `FKDET_PEDIDO_PEDIOD_ID_PEDIDO` (`ID_PEDIDO`),
  CONSTRAINT `FKDET_PEDIDO_PEDIOD_ID_PEDIDO` FOREIGN KEY (`ID_PEDIDO`) REFERENCES `pedido`
(`ID_PEDIDO`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_DET_PEDIDO_ARTICULO_ID_ARTICULO` FOREIGN KEY (`ID_ARTICULO`) REFERENCES `articulo`
(`ID_ARTICULO`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='Tabla de detalle del pedido';

-- Volcando datos para la tabla tiendavirtual.det_pedido: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `det_pedido` DISABLE KEYS */;
INSERT INTO `det_pedido` (`ID_DET_PEDIDO`, `ID_PEDIDO`, `ID_ARTICULO`, `CANTIDAD`) VALUES
        (1, 1, 1, 2);
/*!40000 ALTER TABLE `det_pedido` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.pedido
DROP TABLE IF EXISTS `pedido`;
CREATE TABLE IF NOT EXISTS `pedido` (
  `ID_PEDIDO` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. de pedido',
  `ID_USUARIO` int(11) NOT NULL COMMENT 'Id. de usuario',
  `FECHA_PEDIDO` datetime NOT NULL COMMENT 'Fecha de pedido',
  `ESTADO` int(1) NOT NULL COMMENT 'Estado del pedido',
  PRIMARY KEY (`ID_PEDIDO`),
  KEY `FK_PEDIDO_USUARIO_ID_USUARIO` (`ID_USUARIO`),
  CONSTRAINT `FK_PEDIDO_USUARIO_ID_USUARIO` FOREIGN KEY (`ID_USUARIO`) REFERENCES `usuario`
(`ID_USUARIO`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='Tabla de pedidos';

-- Volcando datos para la tabla tiendavirtual.pedido: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `pedido` DISABLE KEYS */;
INSERT INTO `pedido` (`ID_PEDIDO`, `ID_USUARIO`, `FECHA_PEDIDO`, `ESTADO`) VALUES
        (1, 1, '2015-11-01 15:36:08', 0);
/*!40000 ALTER TABLE `pedido` ENABLE KEYS */;
```

```
-- Volcando estructura para tabla tiendavirtual.perfil
DROP TABLE IF EXISTS `perfil`;
CREATE TABLE IF NOT EXISTS `perfil` (
  `ID_PERFIL` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. de perfil',
  `PERFIL` varchar(50) DEFAULT NULL COMMENT 'Perfil',
  PRIMARY KEY (`ID_PERFIL`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COMMENT='Tala de perfiles';

-- Volcando datos para la tabla tiendavirtual.perfil: ~2 rows (aproximadamente)
/*!40000 ALTER TABLE `perfil` DISABLE KEYS */;
INSERT INTO `perfil` (`ID_PERFIL`, `PERFIL`) VALUES
        (1, 'Cliente'),
        (2, 'Administrador');
/*!40000 ALTER TABLE `perfil` ENABLE KEYS */;


-- Volcando estructura para tabla tiendavirtual.usuario
DROP TABLE IF EXISTS `usuario`;
CREATE TABLE IF NOT EXISTS `usuario` (
  `ID_USUARIO` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Id. de usuario',
  `ID_PERFIL` int(11) NOT NULL DEFAULT '0' COMMENT 'Id. de perfil',
  `USUARIO` varchar(10) NOT NULL DEFAULT '0' COMMENT 'Usuario de acceso',
  `CLAVE` varchar(10) NOT NULL DEFAULT '0' COMMENT 'Clave de acceso',
  `NOMBRE` varchar(240) DEFAULT '0' COMMENT 'Nombre y apellidos del usuario',
  `DIRECCION` varchar(240) DEFAULT '0' COMMENT 'Direccion del usuario',
  `CODPOSTAL` int(5) DEFAULT '0' COMMENT 'Codigo Postal',
  `TELEFONO` int(9) DEFAULT '0' COMMENT 'Telefono',
  `EMAIL` varchar(50) DEFAULT '0' COMMENT 'Correo electronico',
  `NIF` varchar(10) DEFAULT '0' COMMENT 'NIF',
  `CCC` varchar(100) DEFAULT '0' COMMENT 'Codigo Cuenta Corriente',
  PRIMARY KEY (`ID_USUARIO`),
  KEY `ID_PERFIL` (`ID_PERFIL`),
  CONSTRAINT `FK_USUARIO_PERFIL_ID_PERFIL` FOREIGN KEY (`ID_PERFIL`) REFERENCES `perfil`
(`ID_PERFIL`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='Tabla de usuarios';

-- Volcando datos para la tabla tiendavirtual.usuario: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `usuario` DISABLE KEYS */;
INSERT INTO `usuario` (`ID_USUARIO`, `ID_PERFIL`, `USUARIO`, `CLAVE`, `NOMBRE`, `DIRECCION`,
`CODPOSTAL`, `TELEFONO`, `EMAIL`, `NIF`, `CCC`) VALUES
        (1, 1, 'rubenteja', 'ruben75', 'Rubén Teja Rubio', 'Calle del Rio, 925 ', 28080, 699999999,
'rteja@uoc.edu', '54037856C', '2077 0024 00 3102575766');
/*!40000 ALTER TABLE `usuario` ENABLE KEYS */;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

Figura 22. Script de base de datos