



Miteris School

**PROYECTO DESARROLLO DE APLICACIONES
MULTIPLATAFORMA**

Ciclo Formativo de Grado superior de Desarrollo de Aplicaciones
Multiplataforma

Sistema de Venta Online

Autor:

Enrique Pajares Briso-Montiano

Tutor:

Víctor del Pozo Gómez

Valladolid, 12 de junio de 2022

Resumen

Para cualquier empresa de un mundo actual, cada vez más informatizado y competitivo, es indispensable poseer herramientas que ayuden a organizar los recursos y clientes de la misma.

En este caso, se va a desarrollar un sistema de ventas para una tienda online de productos tecnológicos la cual va a ser fácilmente adaptable y aprovechable para otros negocios que requieran de un sistema de venta online.

Concretamente, esta aplicación web se ha diseñado siguiendo el patrón MVC en lenguaje Java con Spring Boot, un almacenamiento en una base de datos SQL y otras tecnologías y librerías que se detallarán en este documento para el desarrollo completo del proyecto.

Para ello, se ha tenido que realizar un previo estudio de requisitos, modelos y diagramas para una correcta organización de la estructura de este trabajo, además de las tecnologías antes mencionadas.

Abstract

For any company in today's increasingly computerized and competitive world, it is essential to have tools that help organize its resources and clients.

In this case, a sales system will be developed for an online store of technological products which will be easily adaptable and usable for other businesses that require an online sales system.

Specifically, this web application has been designed following the MVC pattern in Java language with Spring Boot, storage in an SQL database and other technologies and libraries that will be detailed in this document for the complete development of the project.

For this, it has had to carry out a previous study of requirements, models and diagrams for a correct organization of the structure of this work, in addition to the aforementioned technologies.

Resumen	2
Abstract	2
1. Introducción	8
1.1. Justificación	8
1.2. Objetivos.....	8
1.3. Metodología	9
1.3.1. Java	9
1.3.2. Eclipse	10
1.3.3. MySql	10
1.3.4. Spring Boot	11
1.3.4.1. Facilidad de despliegue con los servidores embebidos.....	11
1.3.4.2. Inversión de control e inyección de dependencias	12
1.3.4.3. Arquitectura REST en Spring Boot y estereotipos	12
1.3.5. Spring Security	12
1.3.5.1. Diseño del espacio de nombres.....	13
1.3.6. Hibernate	13
1.3.7. Thymeleaf	14
1.3.8. JavaScript	14
1.3.9. HTML.....	15
1.3.10. .CSS	16
1.3.11. Bootstrap	17
1.4. Estado del arte.....	18
2. Análisis.....	19
2.1. Requisitos funcionales:.....	19
2.1.1. RF_01 Registro de usuario	19
2.1.2. RF_02 Acceso del usuario a la aplicación	20
2.1.3. RF_03 Cierre de sesión del usuario	20
2.1.4. RF_04 Gestión de productos	21
2.1.5. RF_05 Gestión de usuarios	21
2.1.6. RF_06 Gestión de pedidos	22
2.1.7. RF_07 Gestión de compras.....	22
2.1.8. RF_08 Ver producto.....	23
2.1.9. RF_09 Añadir producto al carrito	23
2.1.10. RF_10 Quitar producto del carrito.....	24

2.1.11.	RF_11 Ver detalle del pedido del carrito	24
2.1.12.	RF_12 Realizar pedido	25
2.1.13.	RF_13 Envío de email.....	25
2.1.14.	RF_14 Gestión de perfil	26
2.1.15.	RF_15 Filtrar productos	26
2.1.16.	RF_16 Ver catálogo productos	27
2.1.17.	RF_17 Ver carrito	27
2.1.18.	RF_18 Validar usuario	28
2.2.	Requisitos no funcionales:.....	28
3.	Diseño	30
3.1.	Modelos y diagramas.....	30
3.1.1.	Diagrama Entidad – Relación.....	30
3.1.2.	Modelo Relacional	31
3.1.3.	Diagrama casos de uso	31
3.1.3.1.	Diagrama casos de uso general	32
3.1.3.2.	Diagrama de casos de uso cliente	33
3.1.3.3.	Diagrama de casos de uso administrador	34
3.1.4.	Modelo de datos.....	35
3.2.	Tablas de casos de uso	36
3.2.1.	CU_01 Registro de usuario	36
3.2.2.	CU_02 Acceso del usuario	37
3.2.3.	CU_03 Ver catálogo de productos.....	38
3.2.4.	CU_04 Ver producto	39
3.2.5.	CU_05 Ver carrito	40
3.2.6.	CU_06 Añadir producto al carrito.....	41
3.2.7.	CU_07 Quitar producto del carrito.....	42
3.2.8.	CU_08 Filtrar productos	43
3.2.9.	CU_09 Gestión de compras	44
3.2.10.	CU_10 Gestión de perfil.....	45
3.2.11.	CU_11 Cierre de sesión.....	46
3.2.12.	CU_12 Gestión de productos.....	47
3.2.13.	CU_13 Gestión de usuarios	48
3.2.14.	CU_14 Gestión de pedidos	49
3.2.15.	CU_15 Validar usuario	50

3.2.16.	CU_16 Envío email	51
3.2.17.	CU_17 Ver detalle del pedido del carrito	52
3.2.18.	CU_18 Realizar pedido.....	53
3.2.19.	CU_19 Ver detalle compra.....	54
3.2.20.	CU_20 Solicitar cancelación.....	55
3.2.21.	CU_21 Exportar PDF	56
3.2.22.	CU_22 Editar perfil	57
3.2.23.	CU_23 Crear producto.....	58
3.2.24.	CU_24 Ver producto	59
3.2.25.	CU_25 Modificar producto	60
3.2.26.	CU_26 Eliminar producto	61
3.2.27.	CU_27 Exportar productos a Excel	62
3.2.28.	CU_28 Crear usuario.....	63
3.2.29.	CU_29 Ver usuario	64
3.2.30.	CU_30 Modificar usuario.....	65
3.2.31.	CU_31 Eliminar usuario	66
3.2.32.	CU_32 Ver pedido.....	67
3.2.33.	CU_33 Editar pedido.....	68
3.2.34.	CU_34 Eliminar pedido	69
3.2.35.	CU_35 Exportar PDF	70
3.3.	Tablas de entidades.....	71
3.3.1.	Rol.....	71
3.3.2.	Usuario.....	71
3.3.3.	Pedido.....	72
3.3.4.	Detalle_Pedido	72
3.3.5.	Producto	73
4.	Implementación.....	74
4.1.	Patrón MVC	74
4.1.1.	¿Qué es el patrón MVC?	74
4.1.2.	¿Por qué utilizar el patrón MVC?	74
4.1.3.	¿Qué es el Modelo?.....	74
4.1.4.	¿Qué es la Vista?.....	75
4.1.5.	¿Qué es el Controlador?	75
4.1.6.	Arquitectura de aplicaciones MVC	75

4.1.7.	Lógica de negocio	76
4.2.	Arquitectura.....	77
4.2.1.	Clases.....	77
4.2.1.1.	Paquete com.tienda.online	78
4.2.1.2.	Paquete com.tienda.online.controller	78
4.2.1.3.	Paquete com.tienda.online.model	79
4.2.1.4.	Paquete com.tienda.online.repository.....	79
4.2.1.5.	Paquete com.tienda.online.service	80
4.2.1.6.	Paquete com.tienda.online.utils	80
4.2.2.	Recursos.....	81
4.2.2.1.	Static	81
4.2.2.2.	Templates	81
4.2.2.3.	Application Properties	82
4.2.3.	Javadoc y Log	83
4.2.4.	POM.....	84
5.	Conclusiones.....	87
5.1.	Conclusiones del proyecto.....	87
5.2.	Trabajo futuro.....	87
6.	Bibliografía.....	88
7.	Anexos	90
7.1.	Manual de usuario.....	90
7.1.1.	Pantalla principal (anónimo)	90
7.1.2.	Login	91
7.1.3.	Registro.....	92
7.1.4.	Vista admin	93
7.1.5.	Vista admin (Productos)	94
7.1.6.	Creación de producto	94
7.1.7.	Vista admin (Usuarios).....	96
7.1.8.	Creación de usuario	97
7.1.9.	Vista admin (Pedidos).....	98
7.1.10.	Vista cliente	99
7.1.11.	Vista cliente (Compras).....	100
7.1.12.	Vista producto	101
7.1.13.	Vista carrito	102

7.1.14. Vista pedido 103

1. Introducción

En los últimos tiempos, el negocio online ha ido creciendo considerablemente en el ámbito comercial de las empresas. Hoy en día, numerosas compañías ofrecen sus productos y servicios a través de internet como vía alternativa o complementaria al sistema de venta tradicional de tienda física que durante tantos años ha sido el único método de comercio que existía.

En este proyecto se va a tratar de desarrollar una aplicación web funcional para la gestión de una tienda online de productos tecnológicos.

1.1. Justificación

Toda empresa necesita evolucionar para mantenerse competitiva en el mercado, adaptarse a los cambios y no estancarse. La digitalización mundial y el Covid ha obligado a muchas empresas a tomar medidas al respecto, una de ellas es la venta online y, por ello, una de las evoluciones necesarias para cualquier negocio es un sistema de venta online.

En particular, he decidido desarrollar esta aplicación web ya que un sistema de venta online es un proyecto básico pero muy completo el cual se puede aplicar a numerosos negocios diferentes.

1.2. Objetivos

Se pretende desarrollar en Java la parte de negocio y visual de una aplicación web, en concreto, una tienda online de informática, implementando la funcionalidad y robustez necesarias para su buen funcionamiento.

Se desea que sea intuitiva, sencilla y fácil de manejar para cualquier tipo de usuario, independientemente de sus conocimientos informáticos.

Se van a desarrollar dos usuarios principales. Uno es el “administrador” que se encargará de gestionar 3 módulos: “Productos”, “Usuarios” y “Pedidos”. El otro usuario es el “cliente”, que realizará compras y podrá gestionarlas. En ambos casos, pueden tratar los datos de su perfil de registro.

Además, otro objetivo principal (a nivel personal), es conocer cómo realizar un proyecto al completo, desde cero, aplicando todos los conocimientos adquiridos, agrandar dichos conocimientos de forma autodidacta y reflejarlos en una sola aplicación.

1.3. Metodología

Realizar completamente un proyecto requiere, a parte del desarrollo de la aplicación en sí, una organización del trabajo. Investigación de diferentes ámbitos, desde la organización de las clases, pasando por el estudio de las diversas tecnologías a aplicar, y acabando en la revisión del proyecto al completo junto con la documentación.

En este proyecto se han utilizado múltiples tecnologías, las cuales se detallarán a continuación:

1.3.1. Java



Java es un lenguaje de programación basado en clases, orientado a objetos. Es un lenguaje de programación de propósito general, es decir, sirve para realizar diversos tipos de aplicaciones (aplicaciones de escritorio, aplicaciones web, apps para dispositivos móviles).

La decisión sobre su utilización radica tanto en el previo conocimiento del lenguaje como en la cantidad de librerías y tecnologías con las que permite la interacción.

Fue comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán, probablemente, a menos que tengan Java instalado, y cada día se crean más. Java es rápido, seguro y fiable. Desde ordenadores portátiles hasta centros de datos, desde consolas para juegos hasta computadoras avanzadas, desde teléfonos móviles hasta Internet, Java está en todas partes. Si es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados.

El lenguaje de programación Java fue desarrollado originalmente por James Gosling, de Sun Microsystems (constituida en 1983 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle),⁴ y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales y librerías de clases en 1991, y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento de las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros han desarrollado también implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.



1.3.2. Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).



1.3.3. MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias

versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.



1.3.4. Spring Boot

Java Spring Boot es una de las herramientas principales del ecosistema de desarrollo web backend con Java, y si tienes interés en convertirte en backend web developer con este lenguaje, necesitarás conocerlo y estar al tanto del impacto que está teniendo este proyecto en las aplicaciones con características enterprise, principalmente en arquitecturas basadas en servicios web (REST y SOAP) y microservicios.

Para ello debes comprender todos estos conceptos que nacen a partir de este proyecto que nos provee Spring.

Primero que todo hay que hacer énfasis en que Spring Boot NO es Spring y que este proyecto surge de la necesidad de hacer aplicaciones Java sin tantas complicaciones de configuración y toda problemática que eso conlleva.

Por ello y por muchas razones más nace Spring Boot, que junto a proyectos como Spring framework, Spring Data, Spring Security, Spring Cloud, entre otros, hacen la combinación perfecta para desarrollar, probar y desplegar nuestras aplicaciones en un entorno rápido, eficaz y bastante simple.

1.3.4.1. Facilidad de despliegue con los servidores embebidos

Con Spring Boot nos olvidamos de tener que desplegar artefactos Jar o War de manera independiente en uno o muchos servidores web diferentes. Porque nos provee una serie de contenedores web servlet para que se despliegue nuestra aplicación automáticamente solo con un “Run”.

Así mismo, de estos contenedores web podemos elegir el que nos convenga: Tomcat, Jetty u Undertow porque vienen embebidos como dependencias y simplemente agregamos el que se adapte a nuestras necesidades. Todo esto con el gestor de dependencias que elijamos, bien sea Maven o Gradle, sin tocar un servidor o configurar otro tipo de cosas.

1.3.4.2. Inversión de control e inyección de dependencias

En el contexto de Spring tenemos dos conceptos muy importantes: la Inyección de dependencias y la inversión de control. Quizás habrás escuchado estos conceptos si has utilizado Spring y es probable que tuvieras la confusión de para qué sirve una cosa y la otra.

1.3.4.3. Arquitectura REST en Spring Boot y estereotipos

Spring Boot al hacer parte de toda la arquitectura de Spring puede interactuar con los demás proyectos, entre ellos tenemos Spring Framework, proyecto que nos provee soporte para construir aplicaciones web, principalmente podemos crear nuestras propias API y desplegar nuestros servicios REST para que podamos interactuar con otros servicios. Incluso para que los desplaguemos de tal forma que cualquier cliente los consuma, bien sea una aplicación móvil, una aplicación web, o cualquier otro tipo de cliente que pueda conectarse bajo el protocolo HTTP.

Toda esta arquitectura la integramos desde Spring Boot como foco principal, además podemos hacer uso de anotaciones tales como Repository, Service, Component, entre muchas otras que nos permiten desarrollar nuestra aplicación bajo la arquitectura que nos provee el framework de Spring, bien sea para interactuar con bases de datos, crear lógica de negocio o simplemente desarrollar un componente general para todas las capas de la aplicación.



1.3.5. Spring Security

Spring Security es un framework de apoyo al marco de trabajo Spring, que dota al mismo de una serie de servicios de seguridad aplicables para sistemas basados en la arquitectura basados en J2EE, enfocado particularmente sobre proyectos contruidos usando SpringFramework. De esta dependencia, se minimiza la curva de aprendizaje si ya es conocido Spring.

Los procesos de seguridad están destinados principalmente, a comprobar la identidad del usuario mediante la autenticación y los permisos asociados al mismo mediante la autorización. La autorización es dependiente de la autenticación ya que se produce posteriormente a su proceso.

Por regla general muchos de estos modelos de autenticación son proporcionados por terceros o son desarrollados por estándares importantes como el IETF adicionalmente, Spring Security proporciona su propio conjunto de características de autenticación.

1.3.5.1. Diseño del espacio de nombres

El espacio de nombres está diseñado para capturar los usos más comunes del framework y para proporcionar una simple y concisa sintaxis para adaptarlos a la aplicación. El diseño se basa en la gran escala de dependencias en el framework, y puede ser dividido en las siguientes áreas:

- **Web/HTTP Security** La parte más compleja. Configura los filtros y los "service beans" usados para aplicar los mecanismos de autenticación.
- **Business Object Security** Opciones para asegurar la capa de servicio
- **AuthenticationManager** Maneja las peticiones de autenticación de otras partes del framework
- **AccessDecisionManager** Proporciona acceso a las decisiones para la web y la seguridad del método. Una por defecto será registrada, pero también se puede optar por utilizar una personalizada, declarada con la sintaxis normal de spring bean
- **AuthenticationProviders** Mecanismos con los que el **AuthenticationManager** autentica los usuarios
- **UserDetailsService** estrechamente relacionada con los **AuthenticationProviders**, pero a menudo también requiere de otros beans.



1.3.6. Hibernate

Hibernate es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

1.3.7. Thymeleaf



Thymeleaf es una biblioteca Java que implementa un motor de plantillas de XML/XHTML/HTML5 (también extensible a otros formatos) que puede ser utilizado tanto en modo web como en otros entornos no web. Se acopla muy bien para trabajar en la capa vista del MVC de aplicaciones web, pero puede procesar cualquier archivo XML, incluso en entornos desconectados.

Proporciona un módulo opcional para la integración con Spring MVC, por lo que se puede utilizar para reemplazar completamente a los archivos JSP en tus aplicaciones construidas con esta tecnología.

El objetivo principal de Thymeleaf es permitir la creación de plantillas de una manera elegante y un código bien formateado. Sus dialectos Standard y SpringStandard permiten crear potentes plantillas naturales que se pueden visualizar correctamente en los navegadores de Internet, por lo que también funcionan como prototipos estáticos. Thymeleaf también puede extenderse desarrollando tus propios dialectos.

1.3.8. JavaScript



JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas y JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo, en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Desde 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de JavaScript. Los navegadores más antiguos soportan por lo menos ECMAScript 3. La sexta edición se liberó en julio de 2015.

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes. Su relación es puramente comercial, tras la compra del creador de Java (Sun Microsystems) de Netscape Navigator (creador de LiveScript) y el cambio de nombre del lenguaje de programación.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). JavaScript es el único lenguaje de programación que entienden de forma nativa los navegadores.

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.



1.3.9. HTML

HTML, siglas en inglés de HyperText Markup Language ('lenguaje de marcado de hipertexto'), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. HTML se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

HTML es un lenguaje de marcado que nos permite indicar la estructura de nuestro documento mediante etiquetas. Este lenguaje nos ofrece una gran adaptabilidad, una estructuración lógica y es fácil de interpretar tanto por humanos como por máquinas.

Sin embargo, a lo largo de sus diferentes versiones, se han incorporado y suprimido diversas características, con el fin de hacerlo más eficiente y facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas (PC de escritorio, portátiles, teléfonos inteligentes, tabletas, etc.) No obstante, para interpretar correctamente una nueva versión de HTML, los desarrolladores de navegadores web deben incorporar estos cambios y el usuario debe ser capaz de usar la nueva versión del navegador con los cambios incorporados. Normalmente los cambios son aplicados mediante parches de actualización automática (Firefox, Chrome) u ofreciendo una nueva versión del navegador con todos los cambios incorporados, en un sitio web de descarga oficial (Internet Explorer). Por lo que un navegador desactualizado no será capaz de interpretar correctamente una página web escrita en una versión de HTML superior a la que pueda interpretar, lo que obliga muchas veces a los desarrolladores a aplicar técnicas y cambios que permitan corregir problemas de visualización e incluso de interpretación de código HTML. Así mismo, las páginas escritas en una versión anterior de HTML deberían ser actualizadas o reescritas, lo que no siempre se cumple. Es por ello que ciertos navegadores todavía mantienen la capacidad de interpretar páginas web de versiones HTML anteriores. Por estas razones, todavía existen diferencias entre distintos navegadores y versiones al interpretar una misma página web.



1.3.10. CSS

CSS (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles (como Firefox OS).

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes. Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales, permitir que varios documentos HTML compartan un

mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento.

La separación del formato y el contenido hace posible presentar el mismo documento marcado en diferentes estilos para diferentes métodos de renderizado, como en pantalla, en impresión, en voz (mediante un navegador de voz o un lector de pantalla), y dispositivos táctiles basados en el sistema Braille. También se puede mostrar una página web de manera diferente dependiendo del tamaño de la pantalla o tipo de dispositivo. Los lectores pueden especificar una hoja de estilos diferente, como una hoja de estilos CSS guardado en su computadora, para sobrescribir la hoja de estilos del diseñador.

La especificación CSS describe un esquema prioritario para determinar qué reglas de estilo se aplican si más de una regla coincide para un elemento en particular. Estas reglas son aplicadas con un sistema llamado de cascada, de modo que las prioridades son calculadas y asignadas a las reglas, así que los resultados son predecibles.

La especificación CSS es mantenida por el World Wide Web Consortium (W3C). El MIME type text/css está registrado para su uso por CSS descrito en el RFC 2318. El W3C proporciona una herramienta de validación de CSS gratuita para los documentos CSS.

1.3.11. Bootstrap



Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación, iconos y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

1.4. Estado del arte

El comercio electrónico está implantado mundialmente en la actualidad. Cada vez son más las empresas que ofrecen sus productos y servicios a través cualquier dispositivo electrónico con acceso a internet.

Este sistema de comercio ofrece grandes ventajas tanto a empresas como a clientes. La comunicación instantánea entre empresa y cliente o empresa y empresa, la globalización y accesibilidad para alcanzar nuevos mercados, la reducción de costes de producción debido a la disminución de intermediarios, automatización y digitalización de los procesos administrativos, amplitud de variedad de clientes y ofrecer un mejor servicio son unas de las ventajas para las empresas. Además, en cuanto a los clientes, existen ventajas para ellos también, como por ejemplo, la comodidad y la rapidez de compra, la mayor oferta de productos o servicios diferentes con lo que se traduce en precios más competitivos... En definitiva, es un sistema en el que ambas partes salen beneficiadas.

Como hemos mencionado, existen dos tipos diferentes de comercio electrónico principalmente, que son:

- B2B (Business to Business): suelen ser intercambios de grandes cantidades de productos o lotes.
- B2C (Business to Customer): es el tipo, en principio, al cual está más enfocada esta aplicación que se ha desarrollado donde la empresa vende uno o varios productos, en cantidades pequeñas, al cliente que lo compre.

Por todo ello, este tipo de negocio hace posible que las empresas adquieran un mayor poder estratégico dentro del mercado actual, siempre y cuando, se mantengan en constante desarrollo y no se queden estancadas sin evolucionar.

2. Análisis

A continuación, se detallarán los requisitos a cumplir de la aplicación:

2.1. Requisitos funcionales:

2.1.1. RF_01 Registro de usuario

RF_01	Registro de usuario
Versión	1.0
Dependencias	-
Descripción	<p>El sistema deberá permitir el registro de un usuario con los siguientes campos:</p> <ul style="list-style-type: none">- Email- Contraseña- Nombre- Apellido 1- Apellido 2- Teléfono- Dirección- Localidad- Provincia- Dni
Prioridad	Alta
Comentarios	<p>Por defecto, tendrá rol de cliente.</p> <p>El campo "email" será único.</p>

2.1.2. RF_02 Acceso del usuario a la aplicación

RF_02	Acceso del usuario a la aplicación
Versión	1.0
Dependencias	-
Descripción	<p>El sistema deberá permitir el acceso a un usuario, independientemente de su rol, a la aplicación en cualquier momento. El acceso será a través del email del usuario y su contraseña.</p> <p>El sistema deberá mantener la sesión del usuario conectado.</p>
Prioridad	Alta
Comentarios	-

2.1.3. RF_03 Cierre de sesión del usuario

RF_03	Cierre de sesión del usuario
Versión	1.0
Dependencias	- RF_02
Descripción	<p>El sistema deberá permitir cerrar sesión al usuario en cualquier momento.</p>
Prioridad	Alta
Comentarios	-

2.1.4. RF_04 Gestión de productos

RF_04	Gestión de productos
Versión	1.0
Dependencias	- RF_02
Descripción	El sistema deberá permitir al usuario “administrador” gestionar productos: <ul style="list-style-type: none">- Creación- Lectura- Modificación- Eliminación- Exportar productos a excel
Prioridad	Alta
Comentarios	-

2.1.5. RF_05 Gestión de usuarios

RF_05	Gestión de usuarios
Versión	1.0
Dependencias	- RF_02
Descripción	El sistema deberá permitir al usuario “administrador” gestionar usuarios: <ul style="list-style-type: none">- Creación- Lectura- Modificación- Eliminación
Prioridad	Alta
Comentarios	-

2.1.6. RF_06 Gestión de pedidos

RF_06	Gestión de pedidos
Versión	1.0
Dependencias	- RF_02
Descripción	El sistema deberá permitir al usuario “administrador” gestionar pedidos: <ul style="list-style-type: none">- Lectura- Modificación- Eliminación- Exportar factura en pdf
Prioridad	Alta
Comentarios	Solamente se podrá exportar la factura si su estado es “Enviado”

2.1.7. RF_07 Gestión de compras

RF_07	Gestión de compras
Versión	1.0
Dependencias	- RF_02
Descripción	El sistema deberá permitir al usuario “cliente” gestionar sus compras: <ul style="list-style-type: none">- Ver detalle de la compra- Solicitar cancelación de pedido- Exportar factura en pdf
Prioridad	Alta
Comentarios	Solamente se podrá exportar la factura si su estado es “Enviado”

2.1.8. RF_08 Ver producto

RF_08	Ver producto
Versión	1.0
Dependencias	-
Descripción	El sistema deberá permitir al usuario ver los detalles de un producto.
Prioridad	Alta
Comentarios	-

2.1.9. RF_09 Añadir producto al carrito

RF_09	Añadir producto al carrito
Versión	1.0
Dependencias	-
Descripción	El sistema deberá permitir al usuario “cliente” añadir un producto al carrito. Podrá seleccionar cuántas unidades del mismo desea añadir.
Prioridad	Alta
Comentarios	Los productos están limitados a un stock.

2.1.10. RF_10 Quitar producto del carrito

RF_10	Quitar producto del carrito
Versión	1.0
Dependencias	-
Descripción	El sistema deberá permitir al usuario “cliente” quitar un producto al carrito.
Prioridad	Alta
Comentarios	Se eliminarán todas las unidades de ese producto.

2.1.11. RF_11 Ver detalle del pedido del carrito

RF_11	Ver detalle del pedido del carrito
Versión	1.0
Dependencias	<ul style="list-style-type: none">- RF_02- RF_09
Descripción	El sistema deberá permitir al usuario “cliente” ver los detalles del pedido actual del carrito. En estos detalles, se mostrará la información del cliente y de los productos seleccionados, previo paso a realizar la compra.
Prioridad	Alta
Comentarios	-

2.1.12. RF_12 Realizar pedido

RF_12	Realizar pedido
Versión	1.0
Dependencias	<ul style="list-style-type: none">- RF_02- RF_11
Descripción	El sistema deberá permitir al usuario “cliente” realizar el pedido del carrito. El cliente deberá seleccionar el método de pago.
Prioridad	Alta
Comentarios	-

2.1.13. RF_13 Envío de email

RF_13	Envío de email
Versión	1.0
Dependencias	<ul style="list-style-type: none">- RF_12- RF_06
Descripción	El sistema deberá enviar un email con los datos de la factura automáticamente después de que el “administrador” haya modificado el estado del pedido a “Enviado”.
Prioridad	Alta
Comentarios	-

2.1.14. RF_14 Gestión de perfil

RF_14	Gestión de perfil
Versión	1.0
Dependencias	- RF_02
Descripción	El sistema deberá permitir al usuario gestionar su propio perfil pudiendo realizar modificaciones de sus campos.
Prioridad	Alta
Comentarios	-

2.1.15. RF_15 Filtrar productos

RF_15	Filtrar productos
Versión	1.0
Dependencias	-
Descripción	El sistema deberá permitir al usuario filtrar productos por su nombre o por su categoría.
Prioridad	Alta
Comentarios	-

2.1.16. RF_16 Ver catálogo productos

RF_16	Ver catálogo productos
Versión	1.0
Dependencias	-
Descripción	El sistema deberá permitir al usuario mostrar el catálogo de productos en la vista principal.
Prioridad	Alta
Comentarios	-

2.1.17. RF_17 Ver carrito

RF_17	Ver carrito
Versión	1.0
Dependencias	-
Descripción	El sistema deberá permitir al usuario mostrar el carrito con los productos que lleva almacenados.
Prioridad	Alta
Comentarios	-

2.1.18. RF_18 Validar usuario

RF_17	Validar usuario
Versión	1.0
Dependencias	- RF_02
Descripción	El sistema deberá validar el usuario que acceda a la aplicación y mantener su sesión.
Prioridad	Alta
Comentarios	-

2.2. Requisitos no funcionales:

- Aplicación web.
- El sistema mantendrá la sesión del usuario conectado.
- Los productos solamente se podrán gestionar por el usuario “administrador” a través de su id y contraseña.
- Los usuarios solamente se podrán gestionar por el usuario “administrador” a través de su id y contraseña.
- Los pedidos solamente se podrán gestionar por el usuario “administrador” a través de su id y contraseña.
- Se conectará con una base de datos MySQL.
- El usuario “cliente” podrá, como máximo, añadir de 10 en 10 unidades al carrito.
- El usuario “anónimo” podrá añadir productos al carrito, pero tendrá que iniciar sesión si quiere realizar la compra.
- Si no se selecciona ninguna imagen para un producto, éste tendrá una imagen por defecto.
- Al acceder a la página inicial se mostrará el catálogo completo de productos.
- Los pedidos realizados podrán encontrarse en alguno de los siguientes estados: ‘pendiente envío’ (PE), ‘pendiente cancelación’ (PC), ‘enviado’ (E) o ‘cancelado’ (C).
- Inicialmente se dispondrá de dos métodos de pago: ‘tarjeta’ o ‘paypal’.
- La aplicación contará con un sistema de log para el registro de las trazas.

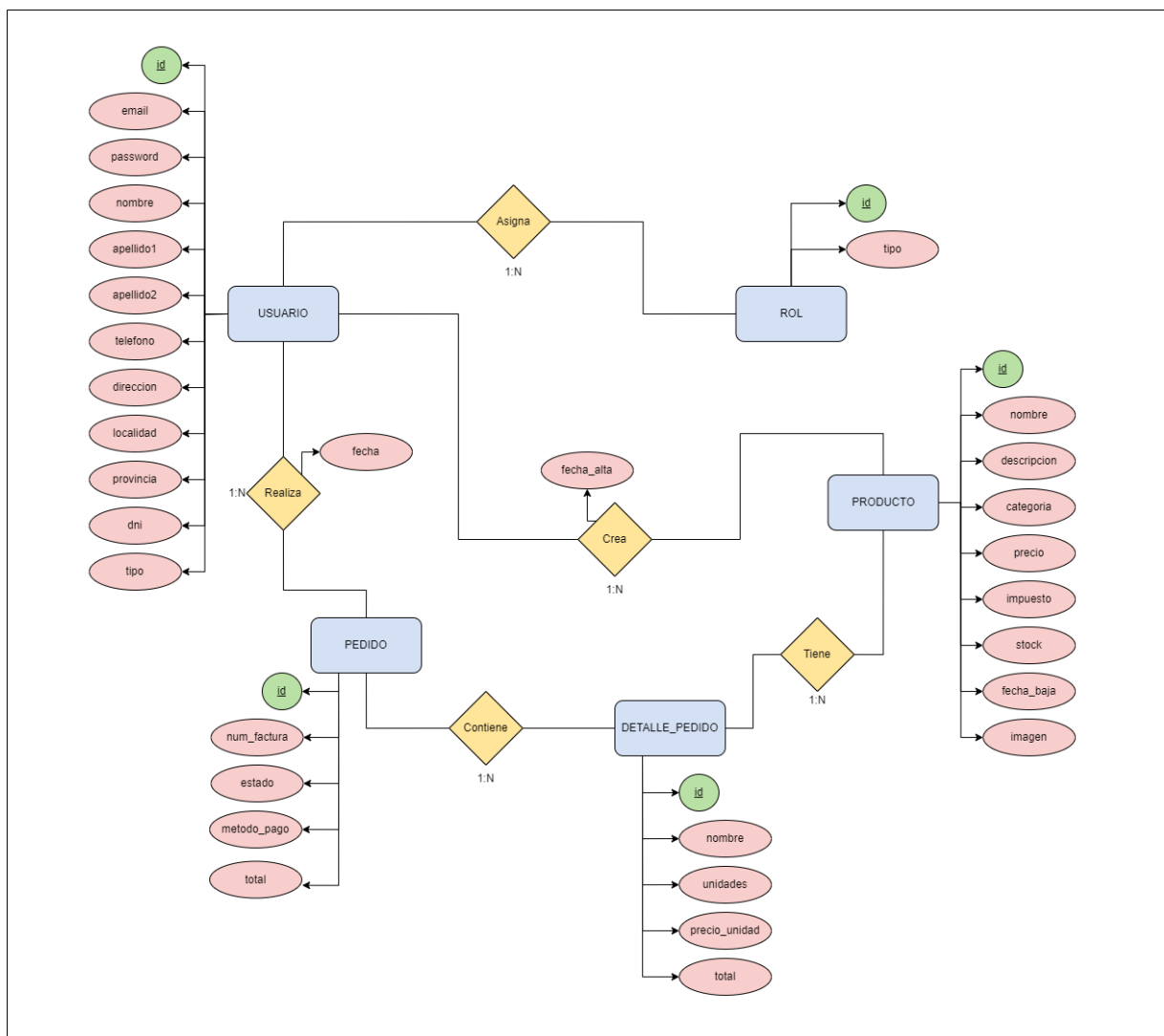
- Se realizará una encriptación de la clave de acceso en el proceso de alta de los usuarios.
- Se realizarán comprobaciones de los datos introducidos por los usuarios, tanto en el lado del cliente como en el lado del servidor (en formulario de registro).

3. Diseño

A continuación, se detallarán los diferentes modelos y tablas sobre el diseño de la aplicación realizados:

3.1. Modelos y diagramas

3.1.1. Diagrama Entidad – Relación



Podemos observar las relaciones de las cinco entidades principales:

- A un Usuario se le “asigna” un Rol.
- Un Usuario (administrador) “crea” varios productos en una fecha de alta concreta.
- Un Usuario (cliente) “realiza” un pedido en una fecha concreta.
- Un Pedido “contiene” varios Detalles del pedido.
- Un Detalle del pedido “tiene” varios Productos.

En este caso, tenemos el mismo diagrama de E-R para la base de datos que para las clases de Java. Esto es así debido a que mediante la anotación `@Entity` que proporciona Spring, se especifica que esa clase representa una entidad de la base de datos, y se crea una tabla de forma automática con cada clase que se encuentre anotada de esta forma.

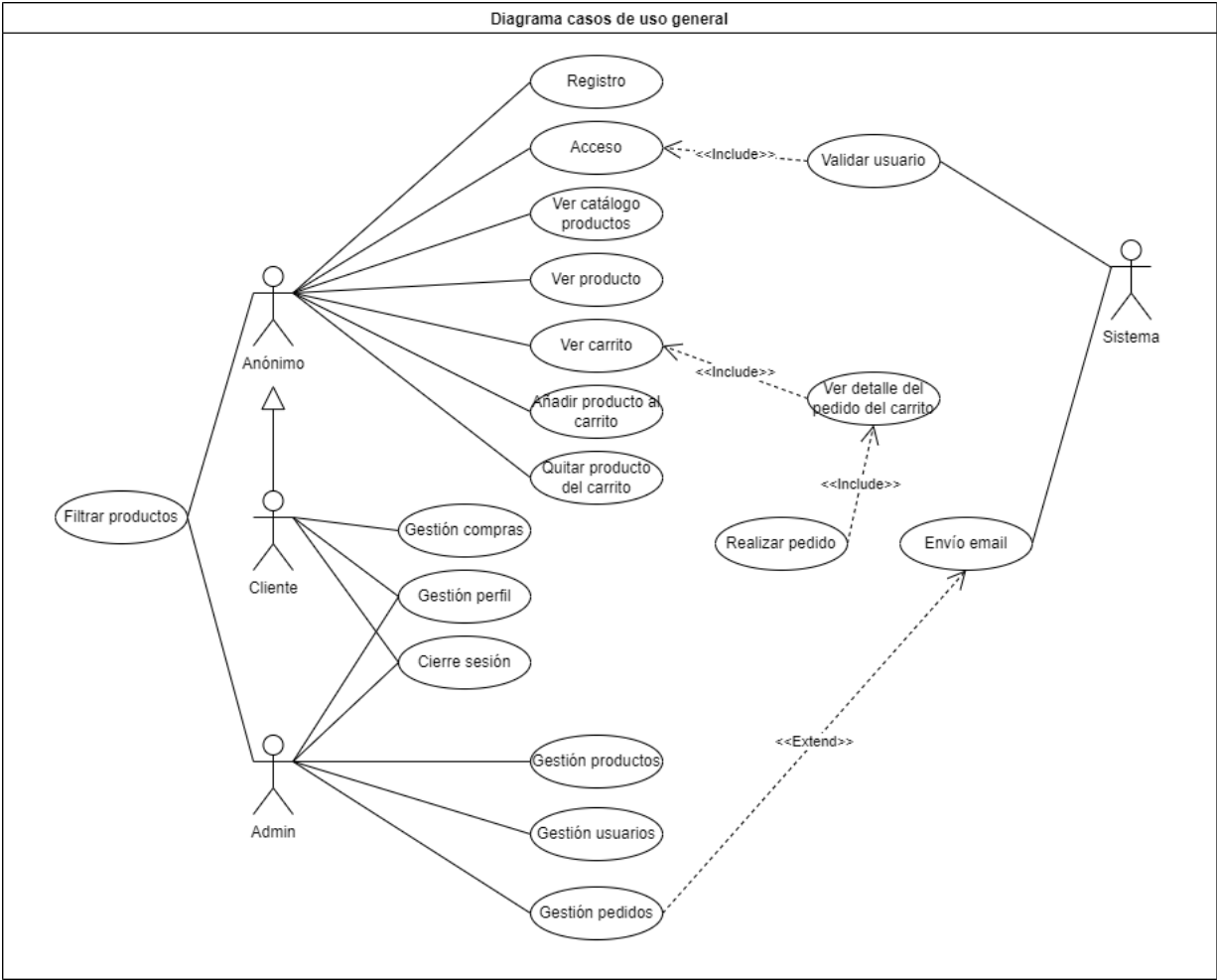
3.1.2. Modelo Relacional

-
- **Rol** (id, tipo)
 - **Usuario** (id, email, password, nombre, apellido1, apellido2, teléfono, dirección, localidad, provincia, dni, tipo, rol_id)
 - **Producto** (id, categoría, descripción, fecha_alta, fecha_baja, imagen, impuesto, nombre, precio, stock, usuario_id)
 - **Pedido** (id, estado, fecha, método_pago, num_factura, total, usuario_id)
 - **Detalle_Pedido** (id, nombre, precio_unidad, total, unidades, pedido_id, producto_id)

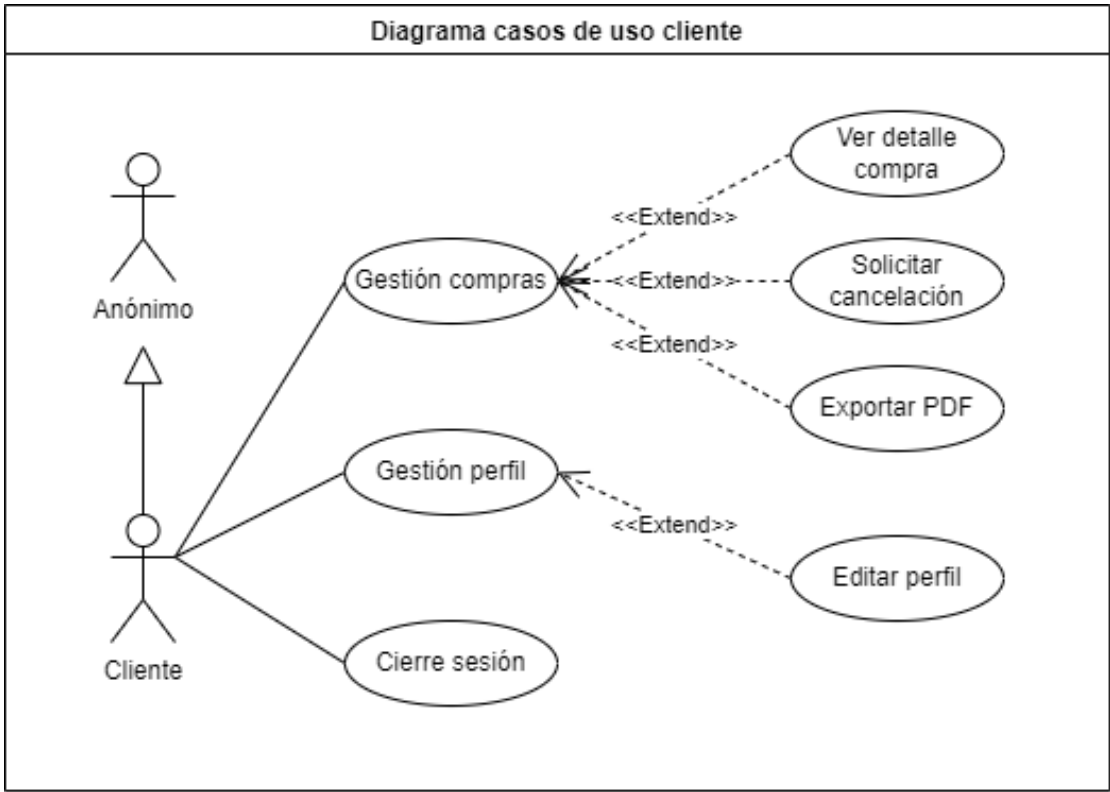
3.1.3. Diagrama casos de uso

A continuación, se mostrará tres diagramas de casos de uso, el primero tendrá una visión global, el segundo sobre el actor “cliente” y el tercero sobre el actor “administrador”. Más adelante en el apartado de tablas de casos de uso detallaremos cada uno.

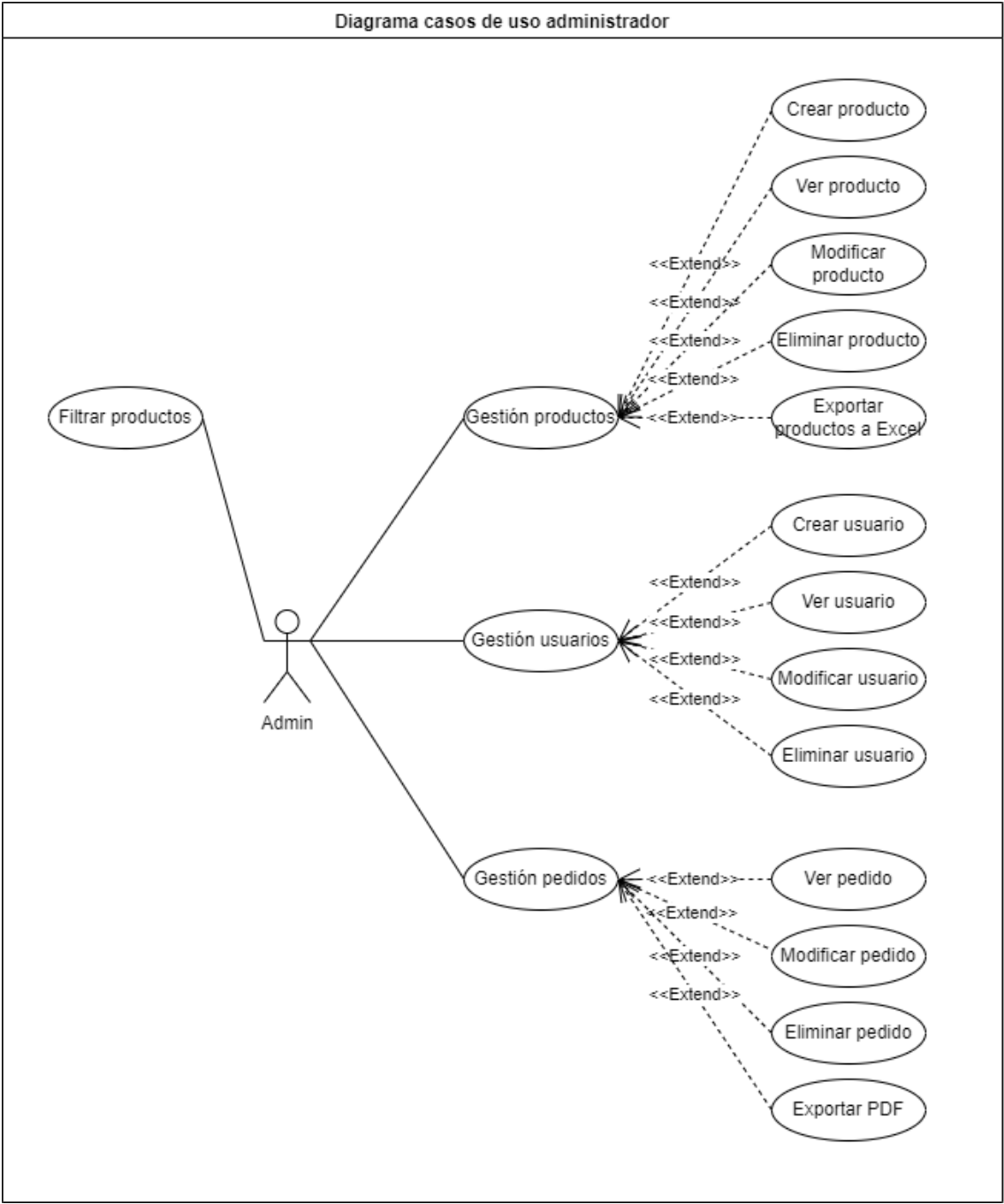
3.1.3.1. Diagrama casos de uso general



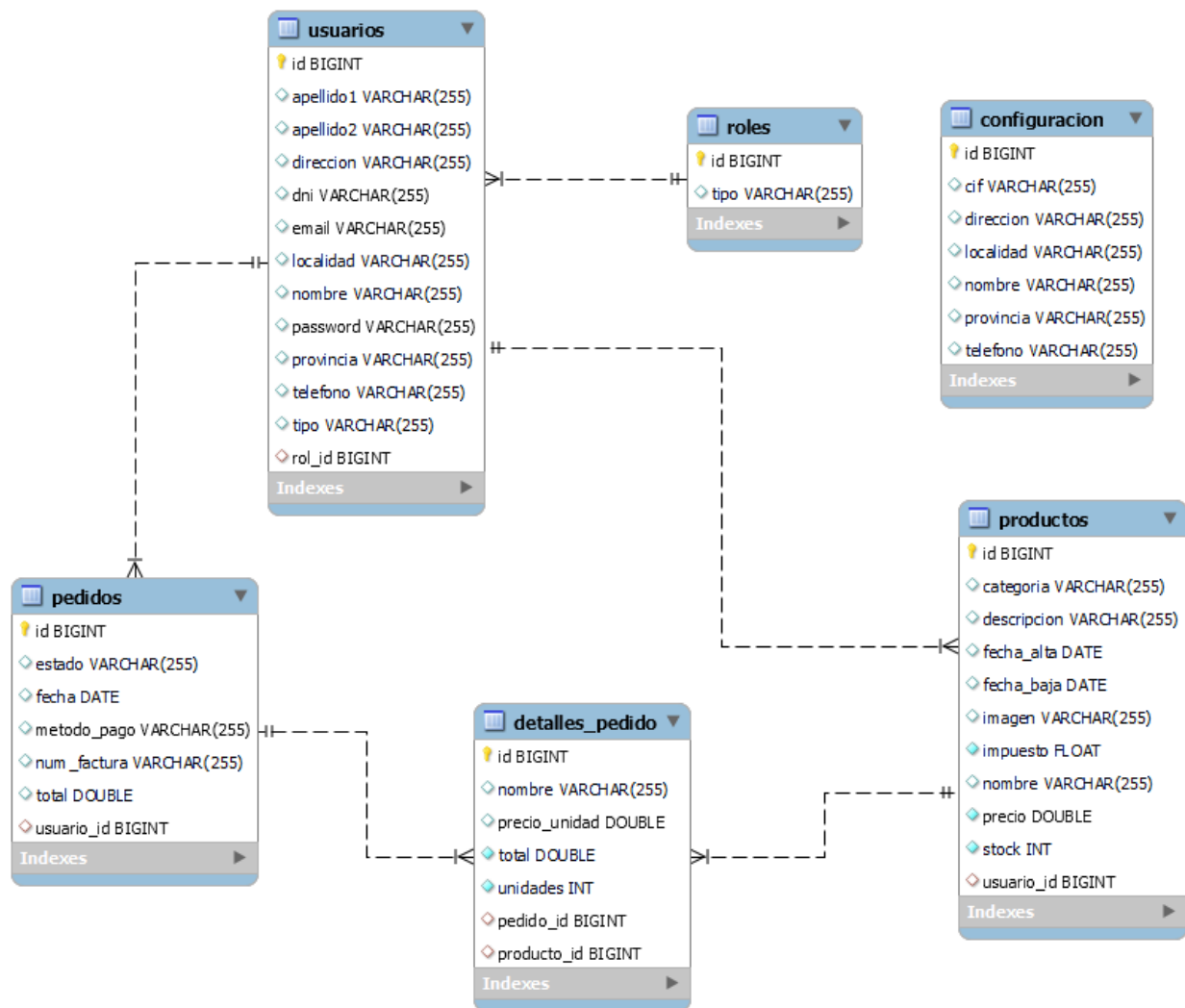
3.1.3.2. Diagrama de casos de uso cliente



3.1.3.3. Diagrama de casos de uso administrador



3.1.4. Modelo de datos



3.2. Tablas de casos de uso

3.2.1. CU_01 Registro de usuario

CASO DE USO	Registro de usuario		CU_01
DESCRIPCIÓN	El usuario se registra en la aplicación.		
ACTORES	Usuario		
PRECONDICIONES	El email no puede estar repetido en la base de datos.		
POSTCONDICIONES	N/A		
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:	
	El usuario pulsa el botón de “Login/Register”.	El sistema accede a la vista “login”.	
	El usuario pulsa al enlace para registrarse.	El sistema accede al formulario.	
	El usuario rellena los campos del formulario.	El sistema recoge los datos introducidos por el usuario.	
	El usuario pulsa el botón “Guardar”		
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:	
	El usuario deja en blanco algún campo requerido.	El sistema avisa al usuario de qué campos le faltan por rellenar.	
	El usuario vuelve a la ventana principal.	El sistema vuelve a la vista principal.	
	El usuario pulsa en “login”.	El sistema vuelve a la vista de “login”.	
COMENTARIOS	N/A		

3.2.2. CU_02 Acceso del usuario

CASO DE USO	Acceso del usuario	CU_02
DESCRIPCIÓN	El usuario accede a la aplicación	
ACTORES	Usuario	
PRECONDICIONES	<p>El usuario debe estar registrado en la base de datos.</p> <p>El email y la contraseña deben ser correctos.</p>	
POSTCONDICIONES	El sistema debe mantener la sesión del usuario.	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	<p>El usuario pulsa el botón "Login/Register".</p> <p>El usuario rellena los campos "email" y "contraseña".</p> <p>El usuario pulsa el botón de "Acceder".</p>	<p>El sistema muestra la vista de "login".</p> <p>El sistema comprueba el "email" y la "contraseña" introducida.</p> <p>El sistema mantiene la sesión de usuario.</p>
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	<p>El usuario introduce un "email" o "contraseña" incorrectos.</p> <p>El usuario deja en blanco algún campo.</p> <p>El usuario accede al enlace de registro.</p> <p>El usuario pulsa el enlace de la "Tienda Online".</p>	<p>El sistema muestra mensaje de error.</p> <p>El sistema informa con un mensaje.</p> <p>El sistema muestra el formulario de registro.</p> <p>El sistema muestra la vista principal.</p>
COMENTARIOS	N/A	

3.2.3. CU_03 Ver catálogo de productos

CASO DE USO	Ver catálogos de productos	CU_03
DESCRIPCIÓN	El usuario accede a la vista principal.	
ACTORES	Usuario	
PRECONDICIONES	Debe de haber registrados productos.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario accede a la vista principal.	El sistema muestra la vista principal.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.4. CU_04 Ver producto

CASO DE USO	Ver producto	CU_04
DESCRIPCIÓN	El usuario ve los detalles de un producto.	
ACTORES	Usuario	
PRECONDICIONES	Debe de estar registrado el producto.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón "Ver producto".	El sistema muestra la vista del producto.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.5. CU_05 Ver carrito

CASO DE USO	Ver carrito	CU_05
DESCRIPCIÓN	El usuario accede a su carrito.	
ACTORES	Anónimo, Cliente	
PRECONDICIONES	N/A	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón "Carrito".	El sistema muestra la vista del "carrito".
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.6. CU_06 Añadir producto al carrito

CASO DE USO	Añadir producto al carrito		CU_06
DESCRIPCIÓN	El usuario añade un producto a su carrito.		
ACTORES	Anónimo, Cliente		
PRECONDICIONES	Debe haber stock de ese producto.		
POSTCONDICIONES	N/A		
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:	
	El usuario pulsa el botón “Ver producto”.	El sistema muestra la vista del producto.	
	El usuario selecciona las unidades que quiera.	El sistema comprueba las unidades en stock de ese producto.	
	El usuario pulsa botón de “Añadir al carrito”.	El sistema añade al carrito el producto.	
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:	
	El usuario pulsa el botón de “Comprar ya!”.	El sistema comprueba las unidades en stock de ese producto.	
		El sistema añade al carrito una unidad del producto.	
COMENTARIOS	N/A		

3.2.7. CU_07 Quitar producto del carrito

CASO DE USO	Quitar producto del carrito	CU_07
DESCRIPCIÓN	El usuario quita un producto de su carrito.	
ACTORES	Anónimo, Cliente	
PRECONDICIONES	El usuario debe haber añadido algún producto a su carrito.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón "Quitar".	El sistema elimina ese producto del carrito. El sistema recalcula el total del carrito.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.8. CU_08 Filtrar productos

CASO DE USO	Filtrar productos	CU_08
DESCRIPCIÓN	El usuario filtra productos del catálogo.	
ACTORES	Usuario	
PRECONDICIONES	Debe haber productos en el catálogo.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario introduce una cadena de caracteres en el campo de "Buscar". El usuario pulsa el botón de "Buscar".	El sistema muestra los productos que coincidan con la cadena de caracteres introducidos en el campo de "Buscar".
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario selecciona una categoría del desplegable "Categorías". El usuario pulsa el botón de "Buscar" asociado.	El sistema muestra los productos que estén en esa categoría.
COMENTARIOS	N/A	

3.2.9. CU_09 Gestión de compras

CASO DE USO	Gestión de compras	CU_09
DESCRIPCIÓN	El cliente accede al apartado de “Compras”	
ACTORES	Cliente	
PRECONDICIONES	El usuario debe haber accedido a la aplicación.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón “Compras”.	El sistema muestra la vista de “compras”.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.10. CU_10 Gestión de perfil

CASO DE USO	Gestión de perfil	CU_10
DESCRIPCIÓN	El usuario accede a su perfil.	
ACTORES	Cliente, Administrador	
PRECONDICIONES	El usuario debe haber accedido a la aplicación.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón donde aparece su nombre.	El sistema muestra la vista de su perfil.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.11. CU_11 Cierre de sesión

CASO DE USO	Cierre de sesión	CU_11
DESCRIPCIÓN	El usuario cierra su sesión.	
ACTORES	Cliente, Administrador	
PRECONDICIONES	El usuario debe haber accedido a la aplicación.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón "Salir".	El sistema cierra la sesión del usuario. El sistema muestra la vista principal.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.12. CU_12 Gestión de productos

CASO DE USO	Gestión de productos	CU_12
DESCRIPCIÓN	El administrador accede al apartado de “Productos”	
ACTORES	Administrador	
PRECONDICIONES	El usuario debe haber accedido a la aplicación.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón “Productos”.	El sistema muestra la vista de “productos”.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.13. CU_13 Gestión de usuarios

CASO DE USO	Gestión de productos	CU_13
DESCRIPCIÓN	El administrador accede al apartado de “Usuarios”	
ACTORES	Administrador	
PRECONDICIONES	El usuario debe haber accedido a la aplicación.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón “Usuarios”.	El sistema muestra la vista de “usuarios”.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.14. CU_14 Gestión de pedidos

CASO DE USO	Gestión de pedidos	CU_14
DESCRIPCIÓN	El administrador accede al apartado de “Pedidos”	
ACTORES	Administrador	
PRECONDICIONES	El usuario debe haber accedido a la aplicación.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón “Pedidos”.	El sistema muestra la vista de “pedidos”.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.15. CU_15 Validar usuario

CASO DE USO	Validar usuario	CU_15
DESCRIPCIÓN	El sistema comprueba las credenciales del usuario.	
ACTORES	Sistema	
PRECONDICIONES	Acción que incluye del CU_02 Acceso del usuario. El usuario debe haber introducido datos en los campos "email" y "contraseña".	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El sistema comprueba las credenciales del usuario.	El sistema comprueba las credenciales del usuario.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.16. CU_16 Envío email

CASO DE USO	Envío email	CU_16
DESCRIPCIÓN	El sistema envía un email al cliente con los datos de la factura.	
ACTORES	Sistema	
PRECONDICIONES	<p>Acción que extiende del CU_14 Gestión de pedidos.</p> <p>El usuario debe haber realizado una compra.</p> <p>El administrador debe haber modificado el estado de la compra a "Enviado".</p>	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El sistema comprueba los datos del cliente.	El sistema comprueba los datos del cliente.
	El sistema comprueba los datos del pedido.	El sistema comprueba los datos del pedido.
	El sistema envía un email con los datos de la factura al cliente.	El sistema envía un email con los datos de la factura al cliente.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.17. CU_17 Ver detalle del pedido del carrito

CASO DE USO	Ver detalle del pedido del carrito	CU_17
DESCRIPCIÓN	El usuario ve los detalles del pedido de su carrito	
ACTORES	Anónimo, Cliente	
PRECONDICIONES	Acción que incluye del CU_05 Ver carrito. El usuario debe haber añadido productos a su carrito.	
POSTCONDICIONES	El usuario se debe haber registrado.	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Ver pedido".	El sistema comprueba que el usuario está registrado. El sistema muestra la vista de "orden". El sistema carga los detalles del cliente. El sistema carga los detalles del carrito.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Ver pedido".	El sistema comprueba que el usuario está registrado, si no lo está, muestra la vista de "login" con el mensaje correspondiente.
COMENTARIOS	N/A	

3.2.18. CU_18 Realizar pedido

CASO DE USO	Realizar pedido	CU_18
DESCRIPCIÓN	El cliente realiza el pedido.	
ACTORES	Cliente	
PRECONDICIONES	Acción que incluye del CU_17 Ver detalle del pedido del carrito.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	<p>El cliente selecciona un método de pago.</p> <p>El cliente pulsa el botón de “Realizar pedido”.</p>	<p>El sistema recoge la información de método de pago.</p> <p>El sistema efectúa el registro del pedido.</p> <p>El sistema muestra la vista principal y el mensaje.</p>
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.19. CU_19 Ver detalle compra

CASO DE USO	Ver detalle compra	CU_19
DESCRIPCIÓN	El cliente ve el detalle de una compra.	
ACTORES	Cliente	
PRECONDICIONES	Acción que extiende de CU_09 Gestión de compras.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El cliente pulsa el botón de "Detalle".	El sistema muestra el detalle de la compra seleccionada.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.20. CU_20 Solicitar cancelación

CASO DE USO	Solicitar cancelación	CU_20
DESCRIPCIÓN	El cliente solicita la cancelación de un pedido.	
ACTORES	Cliente	
PRECONDICIONES	Acción que extiende del CU_09 Gestión de compras. El pedido tiene que realizarse previamente. El estado del pedido debe estar Pendiente de Envío	
POSTCONDICIONES	El estado del pedido debe pasar a Pendiente de Cancelación.	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El cliente pulsa el botón de "Cancelar".	El sistema cambia el estado del pedido a Pendiente de Cancelación. El sistema muestra un mensaje al cliente.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.21. CU_21 Exportar PDF

CASO DE USO	Exportar PDF	CU_21
DESCRIPCIÓN	El usuario descarga la factura en PDF	
ACTORES	Cliente	
PRECONDICIONES	Acción que extiende del CU_09 Gestión de compras. El estado del pedido debe ser Enviado	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Exportar PDF".	El sistema genera un PDF con los datos de la factura y lo descarga en la carpeta de "Descargas" en el dispositivo del cliente.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.22. CU_22 Editar perfil

CASO DE USO	Editar perfil	CU_22
DESCRIPCIÓN	El usuario modifica algún campo de su perfil.	
ACTORES	Cliente, Administrador	
PRECONDICIONES	Acción que extiende del CU_10 Gestión perfil.	
POSTCONDICIONES	En caso de modificar la contraseña, el sistema encriptará la contraseña introducida nueva.	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	<p>El usuario pulsa el botón de “Editar”.</p> <p>El usuario modifica el campo que desee.</p> <p>El usuario pulsa el botón de “Guardar”.</p>	<p>El sistema muestra los campos del perfil.</p> <p>El sistema comprueba los requerimientos de los campos.</p> <p>El sistema guarda las modificaciones realizadas.</p>
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de “Cancelar”.	El sistema vuelve a la vista del perfil del cliente.
COMENTARIOS	N/A	

3.2.23. CU_23 Crear producto

CASO DE USO	Crear producto	CU_23
DESCRIPCIÓN	El administrador crea un producto nuevo.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_12 Gestión de productos.	
POSTCONDICIONES	En caso de no seleccionar ninguna imagen, el sistema cargará la imagen por defecto "producto.png"	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	<p>El administrador pulsa el botón "Crear producto".</p> <p>El administrador completa los campos del producto.</p> <p>El administrador pulsa el botón de "Guardar".</p>	<p>El sistema muestra el formulario para el nuevo producto.</p> <p>El sistema comprueba los requerimientos de los campos.</p> <p>El sistema guarda el producto.</p>
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista de productos.
COMENTARIOS	N/A	

3.2.24. CU_24 Ver producto

CASO DE USO	Ver producto	CU_24
DESCRIPCIÓN	El administrador accede a los detalles de un producto.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_12 Gestión de productos. El producto debe estar creado.	
POSTCONDICIONES		
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Ver producto".	El sistema muestra los datos del producto.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.25. CU_25 Modificar producto

CASO DE USO	Modificar producto	CU_25
DESCRIPCIÓN	El administrador modifica un producto.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_12 Gestión de productos. El producto debe estar creado.	
POSTCONDICIONES	En caso de no seleccionar ninguna imagen nueva, el sistema cargará la imagen que ya tenía asociada.	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Editar". El administrador cambia los datos del producto. El administrador pulsa el botón "Guardar".	El sistema muestra los datos del producto. El sistema comprueba los requerimientos de los campos. El sistema guarda las modificaciones del producto.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista de productos.
COMENTARIOS	N/A	

3.2.26. CU_26 Eliminar producto

CASO DE USO	Eliminar producto	CU_26
DESCRIPCIÓN	El administrador elimina un producto.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_12 Gestión de productos. El producto debe estar creado.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Eliminar".	El sistema solicita confirmación del usuario.
	El administrador pulsa el botón de "Aceptar".	El sistema elimina el producto.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista de productos.
COMENTARIOS	N/A	

3.2.27. CU_27 Exportar productos a Excel

CASO DE USO	Exportar productos a Excel	CU_27
DESCRIPCIÓN	El administrador modifica un producto.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_12 Gestión de productos. Los productos deben estar creados.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Exportar Excel".	El sistema genera un archivo Excel con los datos de los productos en la carpeta de "Descargas" del dispositivo del usuario.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.28. CU_28 Crear usuario

CASO DE USO	Crear usuario	CU_28
DESCRIPCIÓN	El administrador crea un usuario.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_13 Gestión de usuarios. El campo "email" debe ser único en la base de datos.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Crear usuario". El administrador completa los campos del usuario. El administrador pulsa el botón de "Guardar"	El sistema muestra formulario para crear usuario. El sistema comprueba los campos requeridos. El sistema guarda los datos del nuevo usuario.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista de usuarios.
COMENTARIOS	N/A	

3.2.29. CU_29 Ver usuario

CASO DE USO	Ver usuario	CU_29
DESCRIPCIÓN	El administrador accede a los datos de un usuario.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_13 Gestión de usuarios. El usuario debe estar creado.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Detalle".	El sistema muestra los datos del perfil del usuario.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.30. CU_30 Modificar usuario

CASO DE USO	Modificar usuario	CU_30
DESCRIPCIÓN	El administrador modifica un usuario.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_13 Gestión de usuarios. El usuario debe estar creado. El campo "email" debe ser único en la base de datos.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Editar".	El sistema muestra los datos del usuario.
	El administrador modifica los campos del usuario.	El sistema comprueba los campos requeridos.
	El administrador pulsa el botón de "Guardar"	El sistema guarda los datos modificados.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista del perfil del usuario.
COMENTARIOS	N/A	

3.2.31. CU_31 Eliminar usuario

CASO DE USO	Eliminar usuario	CU_31
DESCRIPCIÓN	El administrador elimina un usuario.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_13 Gestión de usuarios. El usuario debe estar creado.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Eliminar".	El sistema solicita confirmación del usuario.
	El administrador pulsa el botón "Aceptar".	El sistema elimina el usuario.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista del perfil del usuario.
COMENTARIOS	N/A	

3.2.32. CU_32 Ver pedido

CASO DE USO	Ver pedido	CU_32
DESCRIPCIÓN	El administrador accede a los datos de un pedido.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_14 Gestión de pedidos. El pedido debe estar creado.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Detalle".	El sistema muestra los datos del pedido.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.2.33. CU_33 Editar pedido

CASO DE USO	Editar pedido	CU_33
DESCRIPCIÓN	El administrador modifica los datos de un pedido.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_14 Gestión de pedidos. El pedido debe estar creado.	
POSTCONDICIONES	Si el administrador modifica el estado del pedido a Enviado se mostrará el número de factura del pedido. Si el administrador modifica el estado del pedido a Enviado se dará al CU_16 Envío email.	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Editar".	El sistema muestra los datos del pedido.
	El administrador modifica los campos del pedido.	El sistema guarda las modificaciones del pedido.
	El administrador pulsa el botón de "Guardar".	El sistema vuelve a la vista de Pedidos.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista de pedidos.
COMENTARIOS	N/A	

3.2.34. CU_34 Eliminar pedido

CASO DE USO	Eliminar pedido	CU_34
DESCRIPCIÓN	El administrador elimina un pedido.	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_14 Gestión de pedidos. El pedido debe estar creado.	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El administrador pulsa el botón "Eliminar". El administrador pulsa el botón "Aceptar".	El sistema solicita confirmación al usuario. El sistema elimina el pedido.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Cancelar".	El sistema vuelve a la vista de pedidos.
COMENTARIOS	N/A	

3.2.35. CU_35 Exportar PDF

CASO DE USO	Exportar PDF	CU_35
DESCRIPCIÓN	El usuario descarga la factura en PDF	
ACTORES	Administrador	
PRECONDICIONES	Acción que extiende del CU_14 Gestión de pedidos. El estado del pedido debe ser Enviado	
POSTCONDICIONES	N/A	
FLUJO PRINCIPAL	Acción del Actor:	Respuesta del Sistema:
	El usuario pulsa el botón de "Exportar PDF".	El sistema genera un PDF con los datos de la factura y lo descarga en la carpeta de "Descargas" en el dispositivo del administrador.
FLUJO ALTERNATIVO	Acción del Actor:	Respuesta del Sistema:
	N/A	N/A
COMENTARIOS	N/A	

3.3. Tablas de entidades

Con la anotación @Entity, tanto en base de datos como en la propia aplicación, encontramos las siguientes entidades o clases:

3.3.1. Rol

ROL	TIPO	DESCRIPCIÓN
id	Long	Identificador del rol
tipo	String	Nombre del rol

3.3.2. Usuario

USUARIO	TIPO	DESCRIPCIÓN
id	Long	Identificador del usuario
apellido1	String	Primer apellido del usuario
apellido2	String	Segundo apellido del usuario
dirección	String	Dirección del usuario
dni	String	Dni del usuario
email	String	Email del usuario
localidad	String	Localidad del usuario
nombre	String	Nombre del usuario
password	String	Contraseña del usuario

provincia	String	Provincia del usuario
telefono	String	Teléfono del usuario
tipo	String	Tipo del usuario
rol_id	Long	Número de rol asociado al usuario (clave foránea)

3.3.3. Pedido

PEDIDO	TIPO	DESCRIPCIÓN
id	Long	Identificador del pedido
estado	String	Estado del pedido
fecha	Date	Fecha del pedido
método_pago	String	Método de pago del pedido
num_factura	String	Número de factura del pedido
total	Double	Suma total del pedido
usuario_id	Long	Número de usuario asociado (clave foránea)

3.3.4. Detalle_Pedido

DETALLE_PEDIDO	TIPO	DESCRIPCIÓN
id	Long	Identificador del detalle del pedido
nombre	String	Nombre del producto

precio_unidad	Double	Precio por unidad del producto
total	Double	Suma total del detalle del pedido
unidades	Int	Número de unidades del producto
pedido_id	Long	Número de pedido asociado (clave foránea)
producto_id	Long	Número de producto asociado (clave foránea)

3.3.5. Producto

PRODUCTO	TIPO	DESCRIPCIÓN
id	Long	Identificador del producto
categoria	String	Categoría del producto
descripcion	String	Descripción del producto
fecha_alta	Date	Fecha de alta del producto
fecha_baja	Date	Fecha de baja del producto
imagen	String	Imagen del producto
impuesto	Float	Impuesto del producto
nombre	String	Nombre del producto
precio	Double	Precio del producto
stock	Int	Stock del producto en almacén
usuario_id	Long	Número de usuario asociado al producto (clave foránea)

4. Implementación

A continuación, se va a explicar la implementación de la aplicación, su arquitectura y distribución de clases referente al código fuente.

4.1. Patrón MVC

En esta aplicación nos hemos basado en el patrón Modelo Vista Controlador. A continuación, vamos a explicar en qué consiste este modelo y por qué lo hemos utilizado:

4.1.1. ¿Qué es el patrón MVC?

En líneas generales, MVC es una propuesta de arquitectura del software utilizada para separar el código por sus distintas responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea muy concreta, lo que ofrece beneficios diversos.

MVC se usa inicialmente en **sistemas donde se requiere el uso de interfaces de usuario**, aunque en la práctica el mismo patrón de arquitectura se puede utilizar para distintos tipos de aplicaciones. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la **separación del código en tres capas diferentes**, acotadas por su responsabilidad, en lo que se llaman **Modelos, Vistas y Controladores**, o lo que es lo mismo, Model, Views & Controllers, si lo prefieres en inglés. En este artículo estudiaremos con detalle estos conceptos, así como las ventajas de ponerlos en marcha cuando desarrollamos.

4.1.2. ¿Por qué utilizar el patrón MVC?

La rama de la ingeniería del software se preocupa por crear procesos que aseguren calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para todo desarrollo, como la estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo y el mantenimiento.

Los ingenieros del software se dedican a estudiar de qué manera se pueden mejorar los procesos de creación de software y una de las soluciones a las que han llegado es la arquitectura basada en capas que separan el código en función de sus responsabilidades o conceptos. Por tanto, cuando estudiamos MVC lo primero que tenemos que saber es que está ahí para **ayudarnos a crear aplicaciones con mayor calidad**.

4.1.3. ¿Qué es el Modelo?

Es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar librerías que nos permiten trabajar con abstracción de bases de datos y persistencia en objetos, en nuestro caso, Hibernate. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.

4.1.4. ¿Qué es la Vista?

Las vistas, como su nombre nos hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más tenemos los códigos HTML, CSS, JS y Bootstrap que nos permite **mostrar la salida**.

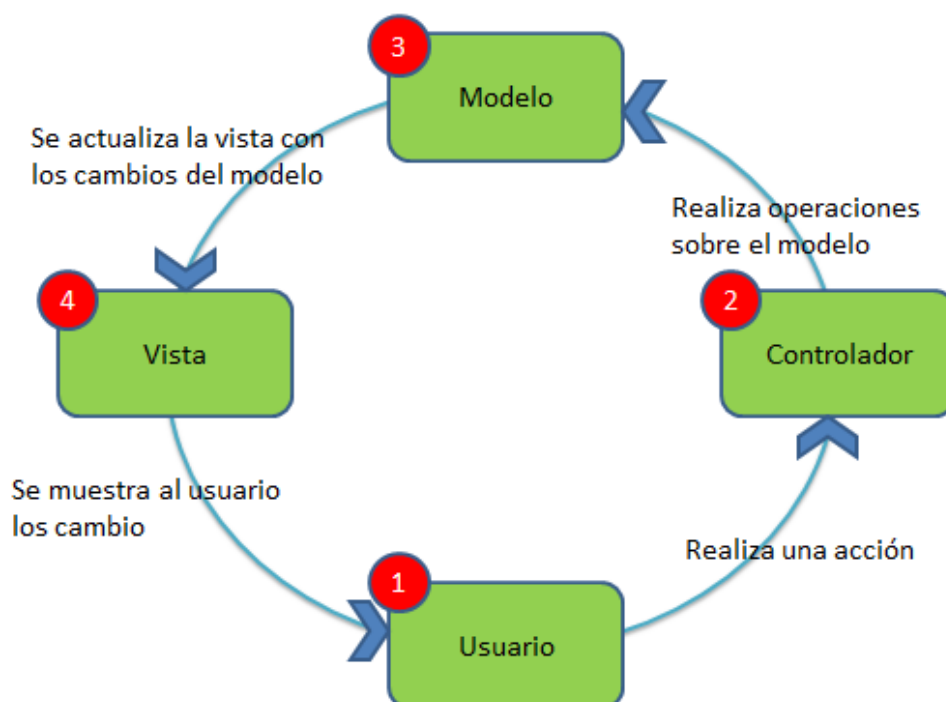
En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generará la salida, tal como nuestra aplicación requiera. En nuestro caso utilizamos las etiquetas Thymeleaf para esa unión.

4.1.5. ¿Qué es el Controlador?

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad, es una capa que sirve de **enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación**. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

4.1.6. Arquitectura de aplicaciones MVC



4.1.7. Lógica de negocio

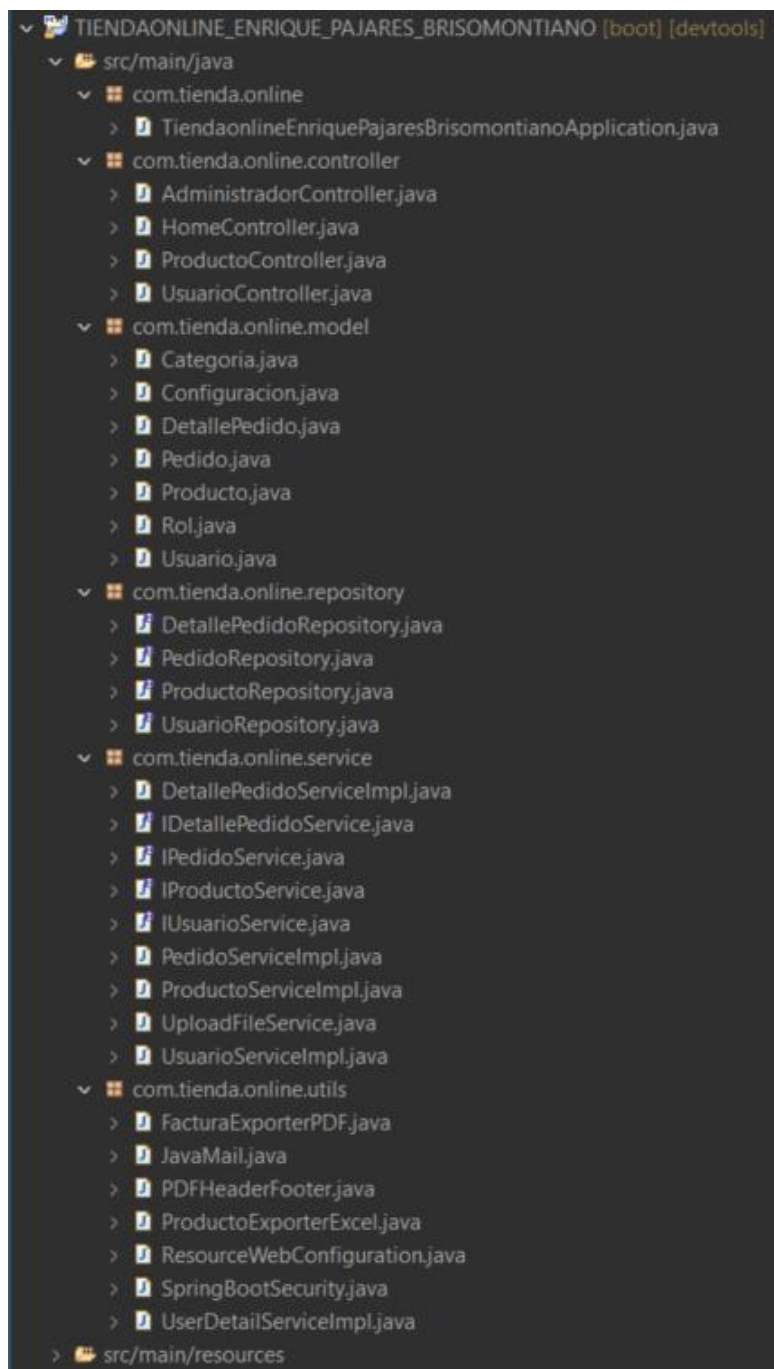
Hay un concepto que se usa mucho cuando se explica el MVC que es la "**lógica de negocio**". **Es un conjunto de reglas que se siguen en el software para reaccionar ante distintas situaciones.** En una aplicación el usuario se comunica con el sistema por medio de una interfaz, pero cuando acciona esa interfaz para realizar acciones con el programa, se ejecutan una serie de procesos que se conocen como la lógica del negocio. Este es un concepto de desarrollo de software en general.

La lógica del negocio, aparte de marcar un comportamiento cuando ocurren cosas dentro de un software, también tiene normas sobre lo que se puede hacer y lo que no se puede hacer. Eso también se conoce como reglas del negocio. Bien, pues en el MVC la lógica del negocio queda del lado de los modelos. Ellos son los que deben saber cómo operar en diversas situaciones y las cosas que pueden permitir que ocurran en el proceso de ejecución de una aplicación.

4.2. Arquitectura

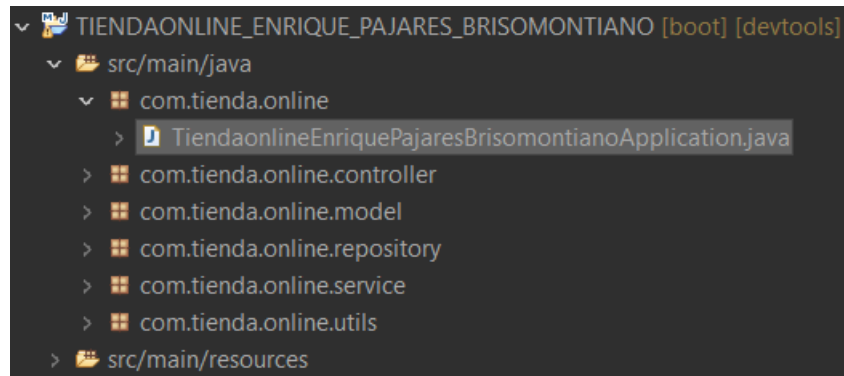
4.2.1. Clases

A continuación, podemos ver una imagen general de todas las clases distribuidas en un paquete principal “com.tienda.online” con cinco subpaquetes (“controller”, “model”, “repository”, “service” y “.utils”), que detallaremos más adelante.



4.2.1.1. Paquete com.tienda.online

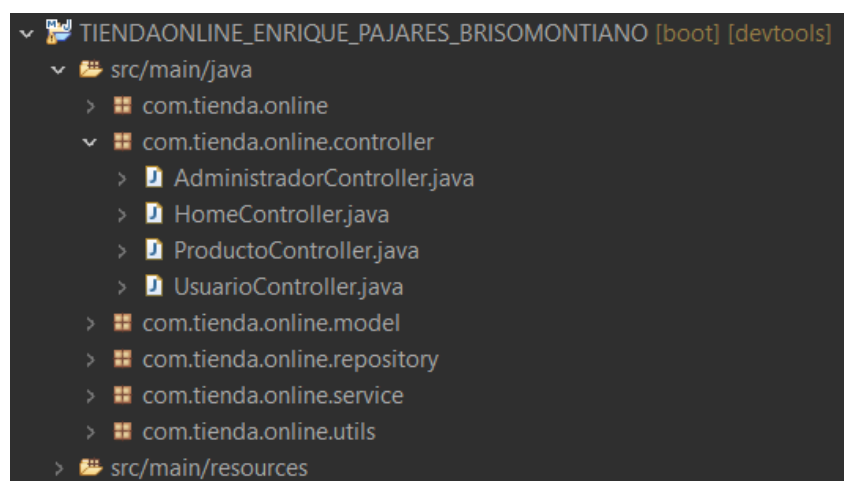
Paquete principal donde se inicia la aplicación.



4.2.1.2. Paquete com.tienda.online.controller

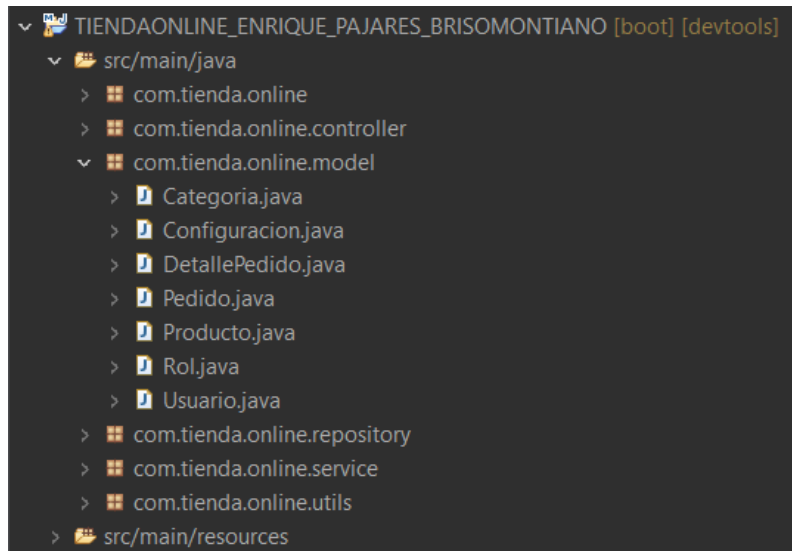
Paquete controlador encargado de gestionar las peticiones del usuario y manipular los datos para llevarlos a la vista. Se designa con la etiqueta @Controller de Spring Framework. Tenemos cuatro clases:

- AdministradorController que gestiona lo relacionado con el administrador.
- HomeController que gestiona la parte general.
- ProductoController que gestiona lo relacionado con los productos.
- UsuarioController que gestiona lo relacionado con el usuario general.



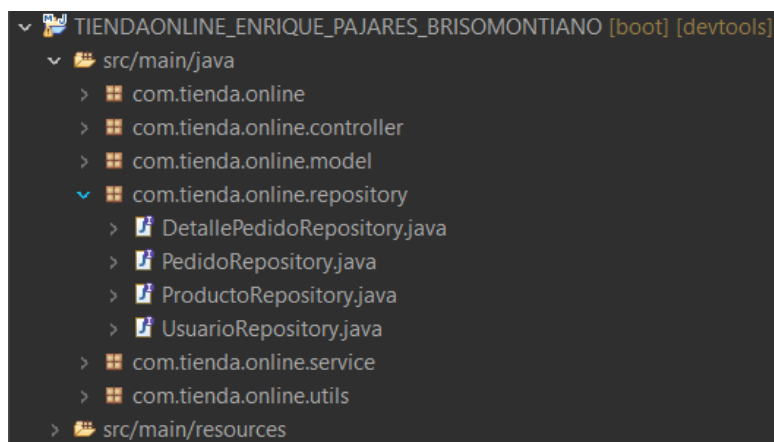
4.2.1.3. Paquete com.tienda.online.model

Paquete modelo donde se tienen todas las clases “POJO” donde se establecen los atributos de cada una y las anotaciones de `javax.persistence.*` y `javax.validation.constraints.*` que determinan la persistencia de los atributos y sus relaciones en la base de datos y la validación de los mismos. . Se designa con la etiqueta `@Entity` de Java Persistence para indicar que es una entidad.



4.2.1.4. Paquete com.tienda.online.repository

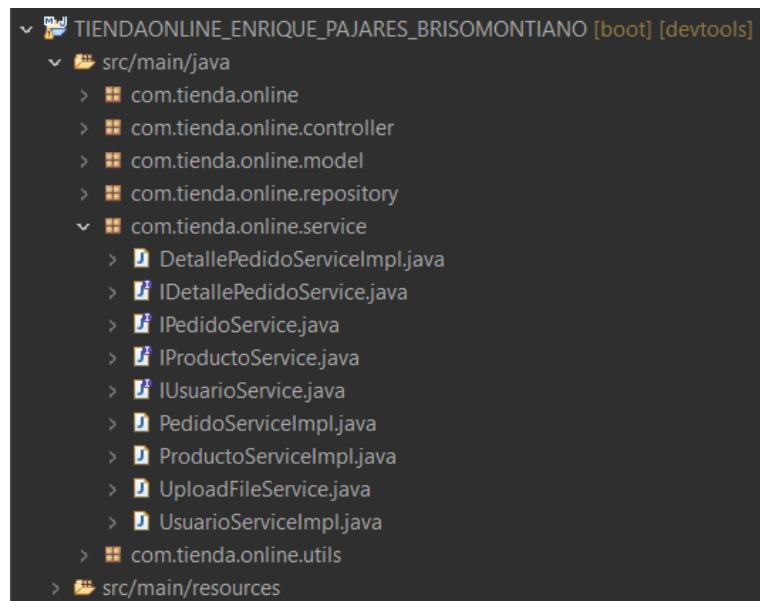
Paquete de interfaces repositorios donde se extiende de `JpaRepository` para utilizar sus métodos, y en algún caso concreto, se determina uno propio. Se designa con la etiqueta `@Repository` de Spring Framework.



4.2.1.5. Paquete com.tienda.online.service

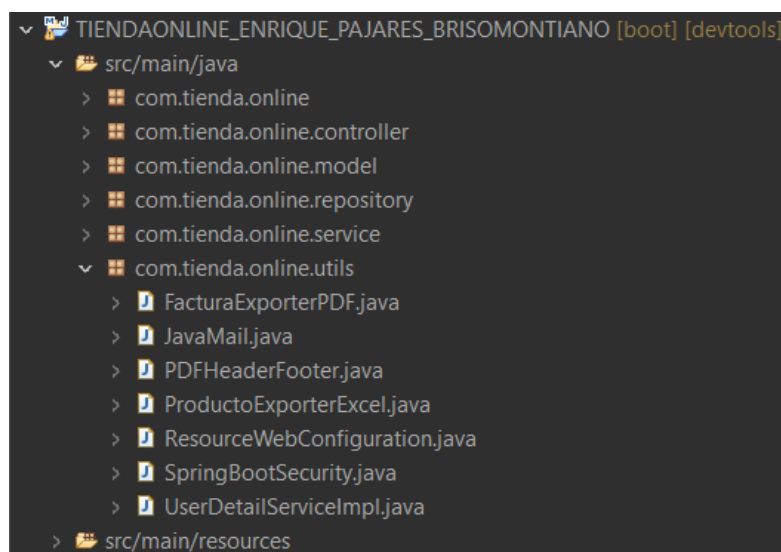
Paquete de servicio donde se establecen e implementan los servicios que se utilizan para realizar determinadas acciones a través de la etiqueta `@Autowired` de Spring para inyectar el objeto a la clase de servicio. Se designa con la etiqueta `@Service` de Spring Framework.

Especialmente, se tiene la clase “UploadFileService” que gestiona la asociación de una imagen a un producto.



4.2.1.6. Paquete com.tienda.online.utils

Paquete de utilidades varias generales que gestionan diferentes acciones como la seguridad de la aplicación con “Spring Security”, la configuración de la carpeta donde dispondrá de las imágenes, exportación de datos a PDF y Excel y el envío de emails.



4.2.2. Recursos

A continuación, podemos ver una imagen general de todos los recursos distribuidas en dos carpetas principales:

4.2.2.1. Static

Recursos estáticos:

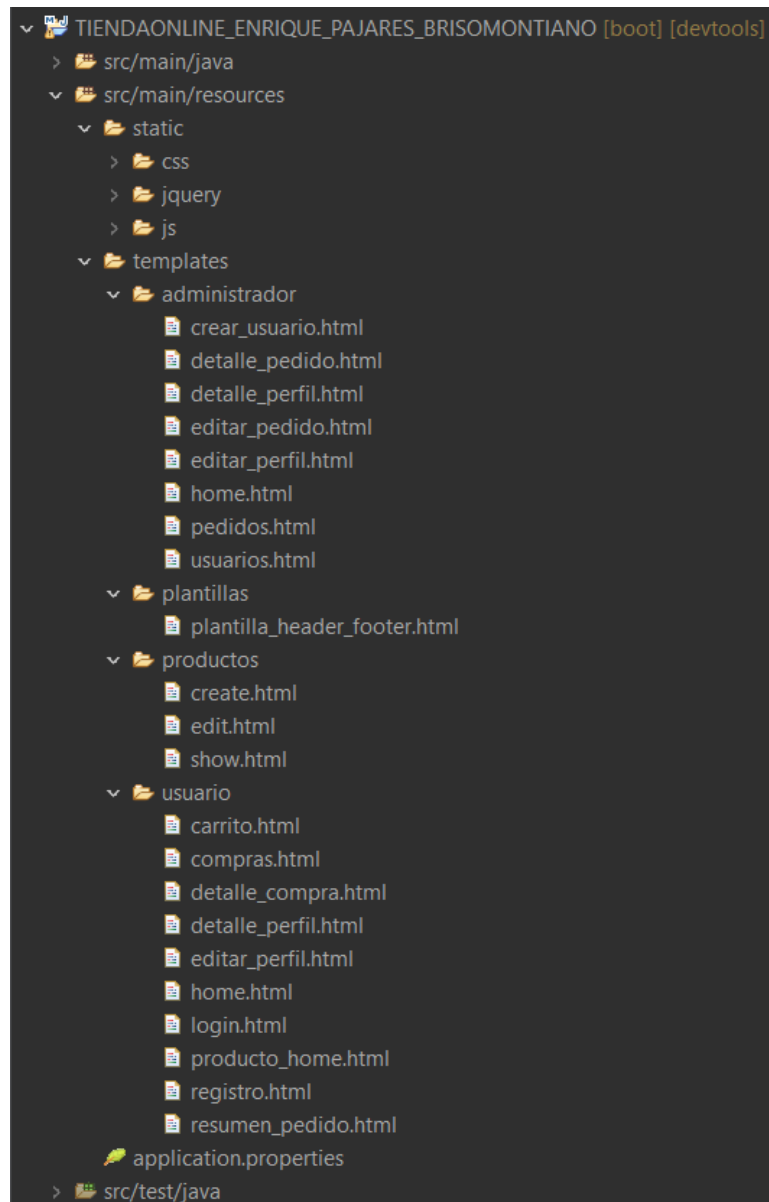
- CSS / bootstrap.min.css – Esta librería contiene todos los estilos de Bootstrap. También se encuentra /heroic-features.css y /custom-background-style.css que se ha personalizado para unos estilos concretos.
- JQuery / jquery.min.js – Esta librería, de código abierto, simplifica la tarea de programar en JavaScript y permite agregar interactividad a un sitio web sin tener conocimientos del lenguaje.
- JS / bootstrap.bundle.min.js – Esta librería contiene el javascript necesario para que los elementos funcionen.

Nota: En la aplicación se importa el cdn “[cdn.jsdelivr.net/npm/bootstrap-icons@1.8.1](https://cdnjs.cloudflare.com/ajax/libs/bootstrap/4.5.0/css/bootstrap.min.css)” utilizado para los iconos.

4.2.2.2. Templates

Plantillas HTML:

- Administrador – Se encuentran todas las vistas relacionadas con el administrador
- Plantillas – Plantilla utilizada para todos los headers y footers de las vistas
- Productos – Se encuentran todas las vistas relacionadas con los productos
- Usuario – Se encuentran todas las vistas relacionadas con los usuarios



4.2.2.3. Application Properties

En este archivo especial se definen las diferentes propiedades para la aplicación como, por ejemplo, en relación con la base de datos, la url, el usuario, la contraseña, la forma de actualización, el dialecto...

En este archivo también se ha definido el tamaño máximo de imágenes y el fichero log.

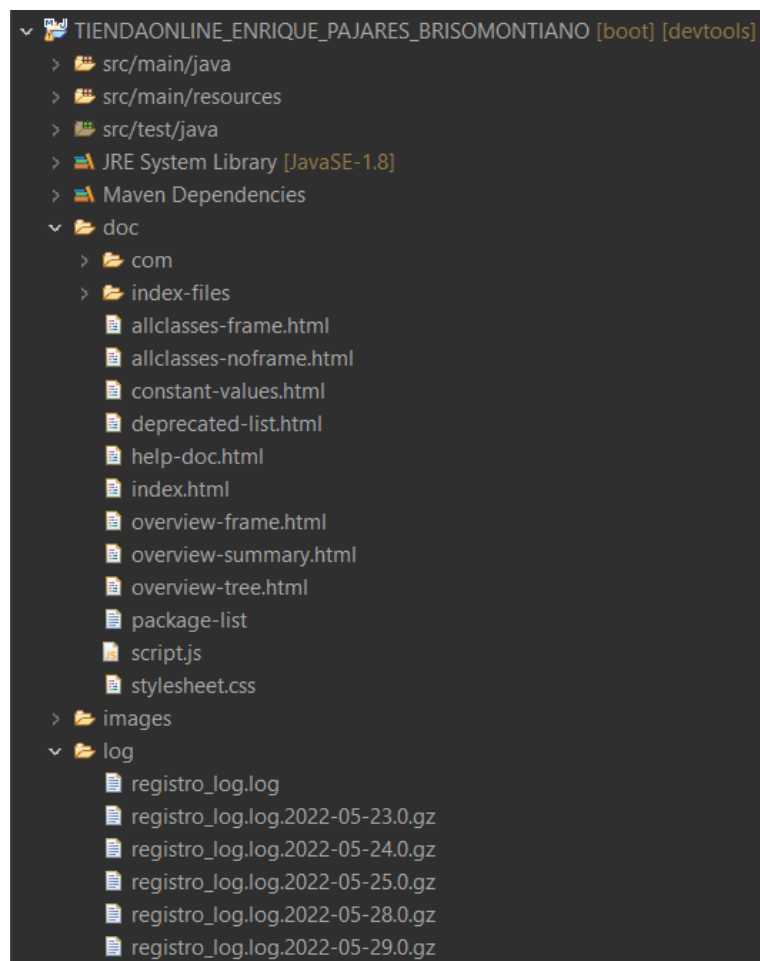
```
spring.datasource.url=jdbc:mysql://localhost:3396/tienda_enrique_pajares_brisomontiano
spring.datasource.username=root
spring.datasource.password=serbatic
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.servlet.multipart.max-file-size=2MB
logging.file.name=log/registro_log.log
```

4.2.3. Javadoc y Log

A continuación, se muestran las carpetas “doc” y “log” donde se ha generado, en el primer caso, el Javadoc que es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java, y en el segundo caso, el sistema log para el registro de las trazas.

Nota: El Log es un mensaje que le indica al desarrollador la ocurrencia de un evento que desea monitorear dentro de la aplicación.



4.2.4. POM

Project Object Model o POM es la parte básica de la funcionalidad de Maven. Este es un archivo XML que tiene información sobre las dependencias, configuraciones y otra información importante sobre el proyecto. Maven revisa esta información y luego realiza la tarea designada.

En la siguiente imagen se puede apreciar la configuración de la aplicación.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.7</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.tienda.online</groupId>
  <artifactId>TIENDAONLINE_ENRIQUE_PAJARES_BRISOMONTIANO</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>TIENDAONLINE_ENRIQUE_PAJARES_BRISOMONTIANO</name>
  <description>Proyecto formación Serbatic de Enrique Pajares</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
```

En la siguiente imagen se establecen las dependencias que se utilizan que son:

❖ **Grupo org.springframework.boot:**

- spring-boot-starter-data-jpa
 - Starter para usar Spring Data JPA con Hibernate
- spring-boot-starter-thymeleaf
 - Starter para usar las etiquetas “th” de thymeleaf en las vistas
- spring-boot-starter-web
 - Starter para crear aplicaciones web
- spring-boot-devtools
 - Para reiniciar de forma automática la aplicación cada vez que se produce un cambio en el código
- spring-boot-starter-test
 - Starter para testear las aplicaciones
- spring-boot-starter-validation
 - Starter para validaciones de datos

- spring-boot-starter-security
 - Starter para usar Spring Security
- ❖ **Grupo mysql:**
 - mysql-connector-java
 - Para la conexión con la base de datos
- ❖ **Grupo org.thymeleaf.extras:**
 - thymeleaf-extras-springsecurity5
 - Para etiquetas de seguridad especiales como sec:authorize
- ❖ **Grupo com.itextpdf:**
 - Itextpdf
 - Para exportar a pdf
- ❖ **Grupo org.apache.poi**
 - poi-ooxml
 - Para exportar a excel
- ❖ **Grupo javax.mail:**
 - Mail
 - Para el envío de emails

```

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>[]
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!-- https://mvnrepository.com/artifact/org.thymeleaf.extras/thymeleaf-extras-springsecurity5 -->
<dependency>
  <groupId>org.thymeleaf.extras</groupId>
  <artifactId>thymeleaf-extras-springsecurity5</artifactId>
  <version>3.0.4.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.itextpdf/itextpdf -->
<dependency>
  <groupId>com.itextpdf</groupId>
  <artifactId>itextpdf</artifactId>
  <version>5.5.3</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi-ooxml</artifactId>
  <version>4.1.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/javax.mail/mail -->
<dependency>
  <groupId>javax.mail</groupId>
  <artifactId>mail</artifactId>
  <version>1.4</version>
</dependency>

```

5. Conclusiones

5.1. Conclusiones del proyecto

Tras el desarrollo de la aplicación he podido comprobar que, hasta la tarea más sencilla, se puede complicar y requiere un esfuerzo extra. Un sistema de venta para una tienda online al completo es complejo de desarrollar, a pesar de ser un sistema muy común que utilizamos habitualmente, hasta que no lo desarrolla uno mismo no eres consciente de las problemáticas que el usuario consumidor, normalmente, no ve. Al estudiar esta formación, me he dado cuenta y soy más consciente de ello cada vez que utilizo una aplicación o navego por una web. Al igual que el realizar la documentación y el trabajo de investigación que conlleva puede ser un trabajo pesado y poco valorado, me doy cuenta que es muy necesario para cuando otra persona quiera usar la aplicación o continuar su desarrollo. Y ya no solo para una persona ajena, sino para una misma en un futuro.

Respecto al desarrollo funcional y tecnológico, el tiempo invertido ha valido para conocer librerías, bibliotecas, frameworks y tecnologías en general, como, por ejemplo, Spring Boot y Bootstrap, que eran casi desconocidas.

La experiencia de realizar una aplicación web desde cero, como ésta, personalmente, ha sido muy positiva y satisfactoria una vez dada por finalizada. Aunque otra percepción que he podido comprobar es que toda aplicación se puede seguir desarrollando para continuar evolucionando y realizar mejoras en ella. En el siguiente punto se van a exponer diferentes progresos en los que se podría seguir trabajando.

5.2. Trabajo futuro

Como he mencionado anteriormente, absolutamente toda aplicación se puede seguir desarrollando. En este caso, se podrían dividir las acciones que desempeña el administrador actualmente y que las pasara a realizar el empleado. Esta idea la dejé a un lado ya que he preferido seguir aumentando la funcionalidad de la aplicación.

Se podría desarrollar una pasarela de pago real e integra, por ejemplo, “Easyredsys”, que es una librería basada en Java que facilita la integración de cualquier TPV Virtual de entidades bancarias colaboradoras con Redsys, como BSCH, BBVA, ING Direct, La Caixa, etc, realizando todo el trabajo técnico y centrándose en la funcionalidad final para los integradores de medios de pago.

En las vistas, donde aparecen los gifs, son espacios reservados donde se mostraría la publicidad comercial de los patrocinadores privados destinados a la financiación de la aplicación.

6. Bibliografía

- *Añade SEGURIDAD a Java con SPRING SECURITY - //Arteco.* (s. f.). Recuperado 4 de junio de 2022, de <https://www.arteco-consulting.com/post/securizando-una-aplicacion-con-spring-boot>
- *Baeldung.* (2020, octubre 30). <https://www.baeldung.com/>
- *Casos a incluir y casos a extender.* (2022, mayo 29). <https://www.abiztar.com.mx/articulos/casos-a-incluir-casos-a-extender.html>
- *Configuración de Spring Security | Marco de Desarrollo de la Junta de Andalucía.* (2022, mayo 28). <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/213>
- contributors, M. O., Jacob Thornton, and Bootstrap. (s. f.). *Introduction.* Recuperado 4 de junio de 2022, de <https://getbootstrap.com/docs/4.5/getting-started/introduction/>
- *Diferencia entre Spring MVC y Spring Boot – Acervo Lima.* (2022, mayo 29). <https://es.acervolima.com/diferencia-entre-spring-mvc-y-spring-boot/>
- *Documentar proyectos Java con Javadoc. Comentarios, símbolos, tags (deprecated, param, etc.) (CU00680B).* (s. f.). Recuperado 4 de junio de 2022, de https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=646:documentar-proyectos-java-con-javadoc-comentarios-simbolos-tags-deprecated-param-etc-cu00680b&catid=68&Itemid=188
- Eclipse (software). (2021). En *Wikipedia, la enciclopedia libre.* [https://es.wikipedia.org/w/index.php?title=Eclipse_\(software\)&oldid=136445513](https://es.wikipedia.org/w/index.php?title=Eclipse_(software)&oldid=136445513)
- García, D. L. (2019, noviembre 2). *Aprende la librería para CSS de Bootstrap desde cero.* Coding Potions - Blog de programación y desarrollo web. <https://codingpotions.com/bootstrap>
- Hibernate. (2022). En *Wikipedia, la enciclopedia libre.* <https://es.wikipedia.org/w/index.php?title=Hibernate&oldid=141565248>
- HTML. (2022). En *Wikipedia, la enciclopedia libre.* <https://es.wikipedia.org/w/index.php?title=HTML&oldid=143753808>
- Java (lenguaje de programación). (2022). En *Wikipedia, la enciclopedia libre.* [https://es.wikipedia.org/w/index.php?title=Java_\(lenguaje_de_programaci%C3%B3n\)&oldid=143137133](https://es.wikipedia.org/w/index.php?title=Java_(lenguaje_de_programaci%C3%B3n)&oldid=143137133)
- JavaScript. (2022). En *Wikipedia, la enciclopedia libre.* <https://es.wikipedia.org/w/index.php?title=JavaScript&oldid=143460295>

- jquery.org, jQuery F.-. (s. f.). *jQuery 3.0 Final Released! | Official jQuery Blog*. Recuperado 4 de junio de 2022, de <https://blog.jquery.com/2016/06/09/jquery-3-0-final-released/>
- *Maven Repository: Search/Browse/Explore*. (s. f.). Recuperado 4 de junio de 2022, de <https://mvnrepository.com/>
- *¿Qué es Java y por qué lo necesito?* (2022, mayo 27). https://www.java.com/es/download/help/whatis_java.html
- *Qué es MVC*. (2022, mayo 29). <https://desarrolloweb.com/articulos/que-es-mvc.html>
- *Qué es y para qué sirve Java Spring Boot*. (2022, mayo 28). Platzi. <https://platzi.com/blog/que-es-spring-boot/>
- *¿Qué son POM (Project Object Model) y pom.xml en Maven? - Otro*. (s. f.). Recuperado 4 de junio de 2022, de <https://spa.myservername.com/what-are-pom>
- *Spring Boot*. (s. f.). Recuperado 4 de junio de 2022, de <https://spring.io/projects/spring-boot>
- *Spring Boot—Starters*. (2021, julio 10). *GeeksforGeeks*. <https://www.geeksforgeeks.org/spring-boot-starters/>
- *Thymeleaf*. (2021). En *Wikipedia, la enciclopedia libre*. <https://es.wikipedia.org/w/index.php?title=Thymeleaf&oldid=138353313>
- *W3Schools Free Online Web Tutorials*. (s. f.). Recuperado 4 de junio de 2022, de <https://www.w3schools.com/>

7. Anexos

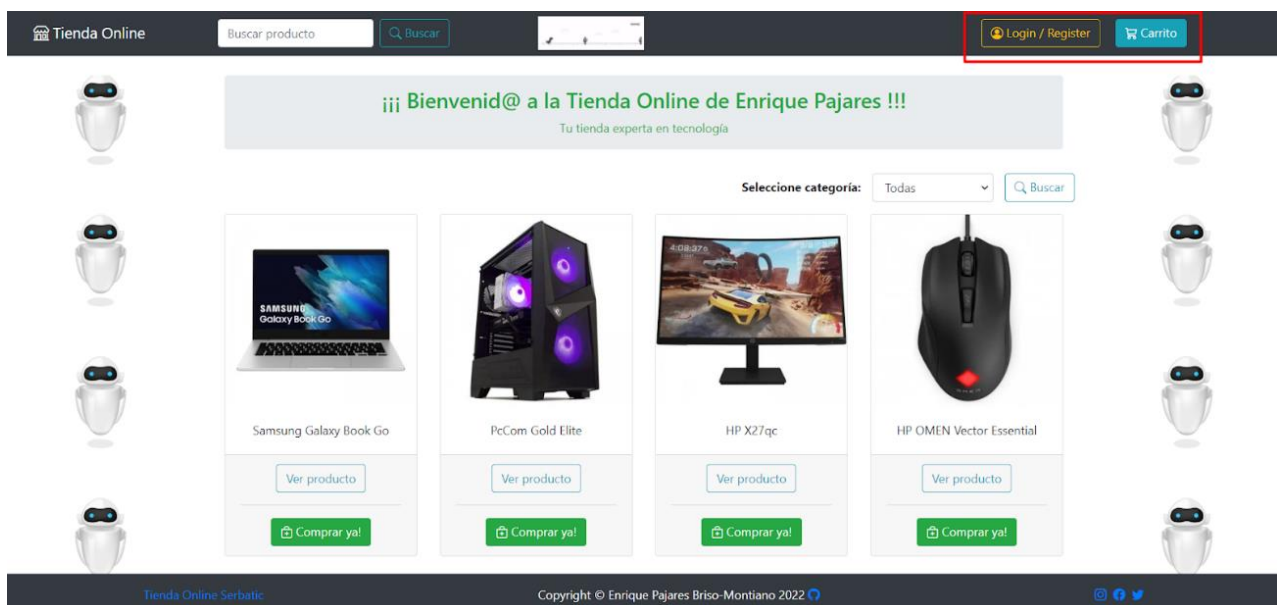
7.1. Manual de usuario

7.1.1. Pantalla principal (anónimo)

En la pantalla inicial se muestra el catálogo con todos los productos. Existen dos tipos de filtros, uno en el encabezado de la página donde se puede filtrar por nombre, y otro encima de los productos donde se puede filtrar por categoría.

Se permite visualizar un producto a través del botón “Ver producto” y añadir al carrito directamente a través del botón “Comprar ya!”

En el encabezado también se encuentran dos botones, el de “Login/Register” para acceder a la vista de “Login” y el botón de “Carrito” para acceder al carrito.



7.1.2. Login

En la vista de “Login” se accede a la aplicación a través del email del usuario y su contraseña. En caso de no estar registrado, se puede ir al formulario de “Registro” desde el enlace inferior para registrarse en la aplicación.

Acceso a la tienda:

Email:

Contraseña:

[Acceder](#)

[Si aún no tiene cuenta puede registrarse aquí](#)

7.1.3. Registro

En el formulario de “Registro” se debe introducir los campos requeridos del usuario. En caso de dejar algún campo obligatorio en blanco o erróneo el sistema lo indicará. En este formulario, los mensajes son personalizados.

Registro

Nombre:

Debe introducir su nombre

Primer apellido:

Debe introducir su apellido

Segundo apellido:

Email:

Debe introducir su email

Teléfono:

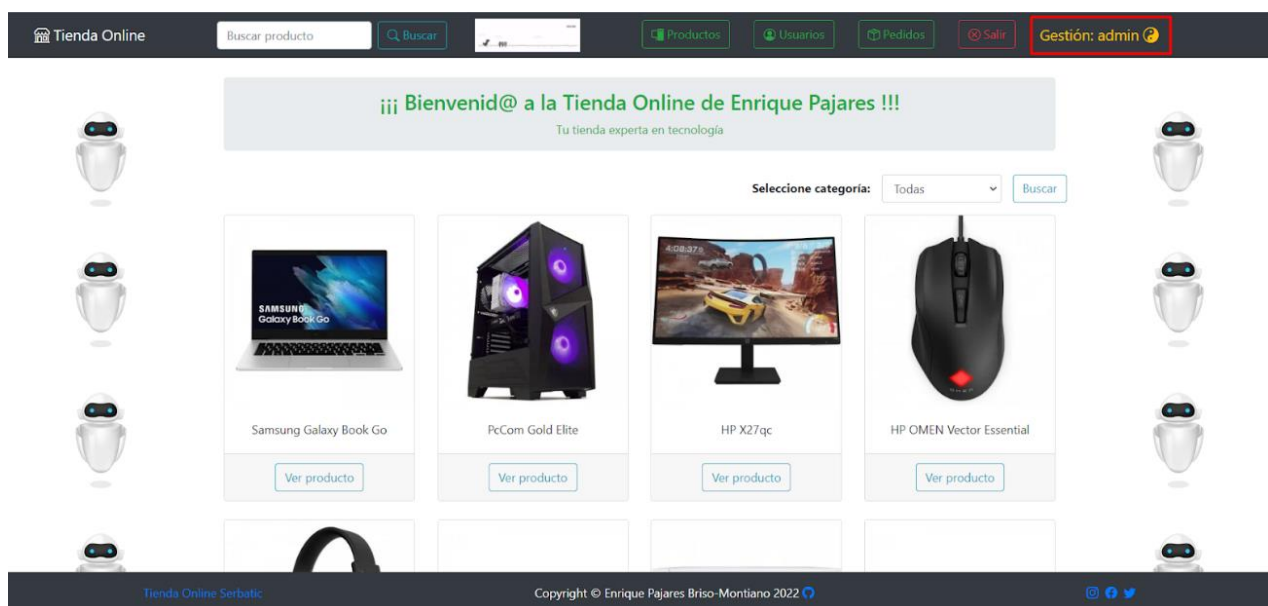
Debe componerse de 9 cifras
Debe introducir su teléfono

Dirección:

7.1.4. Vista admin

En la vista principal de administrador se muestra el catálogo de productos, pero solo se pueden visualizar a través del botón “Ver producto”, es decir, no se tiene el botón anterior de “Comprar ya!” ya que no tiene sentido para el administrador.

En el encabezado se muestran las opciones del administrador que se verán más adelante: Productos, Usuarios, Pedidos, Salir y el nombre del administrador desde donde se gestiona su perfil, además de los filtros antes indicados.



7.1.5. Vista admin (Productos)

En el apartado del administrador de “Productos” se gestiona todo lo relacionado con los mismos. A través de sus botones se puede crear un nuevo producto, se puede editar, eliminar y exportar a un archivo Excel todos los productos.

Productos

Exportar excel | Crear producto

Home / Ver productos

Nombre	Categoría	Descripción	Stock	Precio	Fecha	Editar	Eliminar
Samsung Galaxy Book Go	portátiles	Qualcomm Snapdragon 7c Gen2 4GB 128GB 14" Plata	447	401.64	2022-05-23		
PcCom Gold Elite	torres	Intel Core i5 11400F 16GB 500GBSSD RTX3060	257	1114.47	2022-05-23		
HP X27qc	monitores	LED QHD 165Hz FreeSync Premium Curva	544	329.0	2022-05-23		
HP OMEN Vector Essential	ratones	Ratón Gaming 7200 DPI Negro	682	32.81	2022-05-23		
Forgeon Sergeant	auriculares	Auriculares Gaming 7.1 PC PS4 PS5 Xbox	426	69.99	2022-		

Tienda Online Serbatic | Copyright © Enrique Pajares Briso-Montiano 2022 |

7.1.6. Creación de producto

En el formulario de “Crear producto” el administrador debe rellenar los campos que definen un producto.

En la parte inferior, desde el botón de “Seleccionar archivo”, se puede elegir una imagen local para asociarla al producto. Si no se selecciona ninguno, el sistema asociará una imagen por defecto que se llama “producto.png”.

Crear producto

Nombre:

Categoría:

Seleccione categoría. ▾

Descripción:

Stock:

Precio:

Impuesto:

Fecha:


dd/mm/aaaa 

Imagen:

Seleccionar archivo Ninguno archivo selec.

Guardar

Cancelar

7.1.7. Vista admin (Usuarios)

En el apartado de “Usuarios” se gestiona todo lo relacionado con los usuarios registrados. A través de sus botones se puede crear un nuevo usuario, ver su detalle, editar y eliminar.

Usuarios [Crear Usuario](#)

[Home](#) / [Ver Usuarios](#)

Nombre	Apellido	Email	Teléfono	Tipo	Ver	Editar	Eliminar
admin	admin	admin@admin	666666666	ADMIN	Detalle	Editar	Eliminar
Enrique	Pajares	kike_montiano@hotmail.com	696680508	USER	Detalle	Editar	Eliminar
Anabel	Escudero	anaes@gmail.com	654897125	USER	Detalle	Editar	Eliminar
Gonzalo	García	gonga@gmail.com	654792014	USER	Detalle	Editar	Eliminar
Sara	Sanz	sarsa@gmail.com	654780514	USER	Detalle	Editar	Eliminar

Tienda Online Serbatic Copyright © Enrique Pajares Briso-Montiano 2022

7.1.8. Creación de usuario

El formulario de “Crear usuario” es el mismo para cuando un usuario se registra. Se debe rellenar todos los campos requeridos y al estar en la sesión del administrador, se tiene que seleccionar el tipo de usuario que se va a registrar.

Crear usuario

Nombre:

Introduzca el nombre del usuario

Primer apellido:

Introduzca su primer apellido

Segundo apellido:

Introduzca su segundo apellido

Email:

Introduzca su email

Teléfono:

Introduzca su teléfono

Dirección:

Introduzca su dirección

Localidad:

Introduzca su localidad

Provincia:

Introduzca su provincia

DNI:

Introduzca su dni

Contraseña:

Introduzca su contraseña

Tipo:

Usuario

Guardar

Cancelar

7.1.9. Vista admin (Pedidos)

En el apartado de “Pedidos” se gestiona todo lo relacionado con los mismos. El administrador puede ver el detalle de un pedido, editarlo, eliminarlo o exportarlo a un archivo pdf en caso de que el pedido esté enviado, al igual que la sesión fuera del cliente como veremos más adelante.

Igualmente, el número de factura se mostrará en el caso de que el estado del pedido sea “E” de enviado.

Pedidos

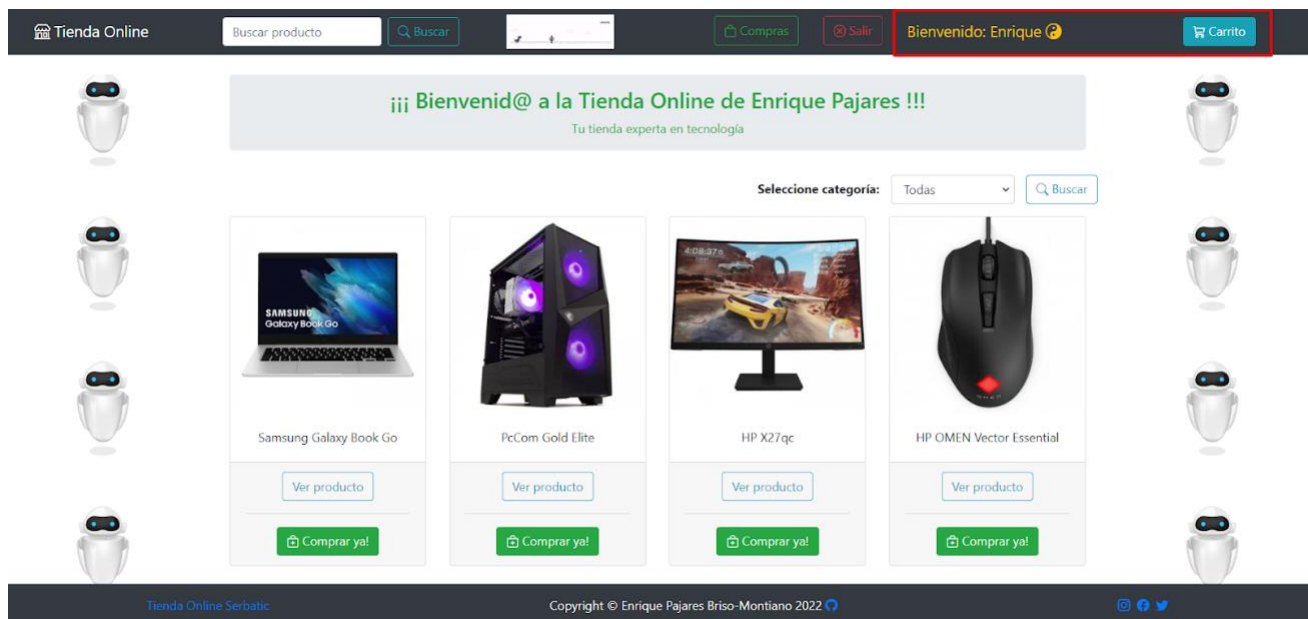
Home / Pedidos

N° de Factura	Valor Total	Estado	Método de pago	Fecha compra	Ver	Editar	Eliminar	Exportar
FRA.2022-05-23T21:11:38.466	401.64 €	E	tarjeta	2022-05-23	Detalle	Editar	Eliminar	PDF
S/N	299.0 €	C	tarjeta	2022-05-23	Detalle	Editar	Eliminar	
S/N	1551.3 €	PE	paypal	2022-05-23	Detalle	Editar	Eliminar	
S/N	401.64 €	PC	tarjeta	2022-05-23	Detalle	Editar	Eliminar	
S/N	1605.89 €	PE	paypal	2022-05-23	Detalle	Editar	Eliminar	
FRA.2022-05-23T21:12:47.854	2299.5 €	E	paypal	2022-05-23	Detalle	Editar	Eliminar	PDF

Tienda Online Serbatic Copyright © Enrique Pajares Briso-Montiano 2022

7.1.10. Vista cliente

En el encabezado de la vista principal del cliente se muestran sus opciones que se detallan a continuación: Compras, Salir, el nombre del cliente desde donde se puede ver/editar el perfil y su Carrito.



7.1.11. Vista cliente (Compras)

En el apartado de “Compras” se muestran todos los pedidos del cliente donde se puede ver el detalle de uno, cancelar en caso de que el estado del pedido sea “PE” de pendiente de envío y exportar la factura a un archivo pdf en caso de que el estado del pedido sea “E” de Enviado.

El número de la factura se muestra si el pedido se ha enviado.

Top navigation bar: Tienda Online, Buscar producto, Buscar, Compras (highlighted), Salir, Bienvenido: Enrique, Carrito.

Compras

Home / Compras

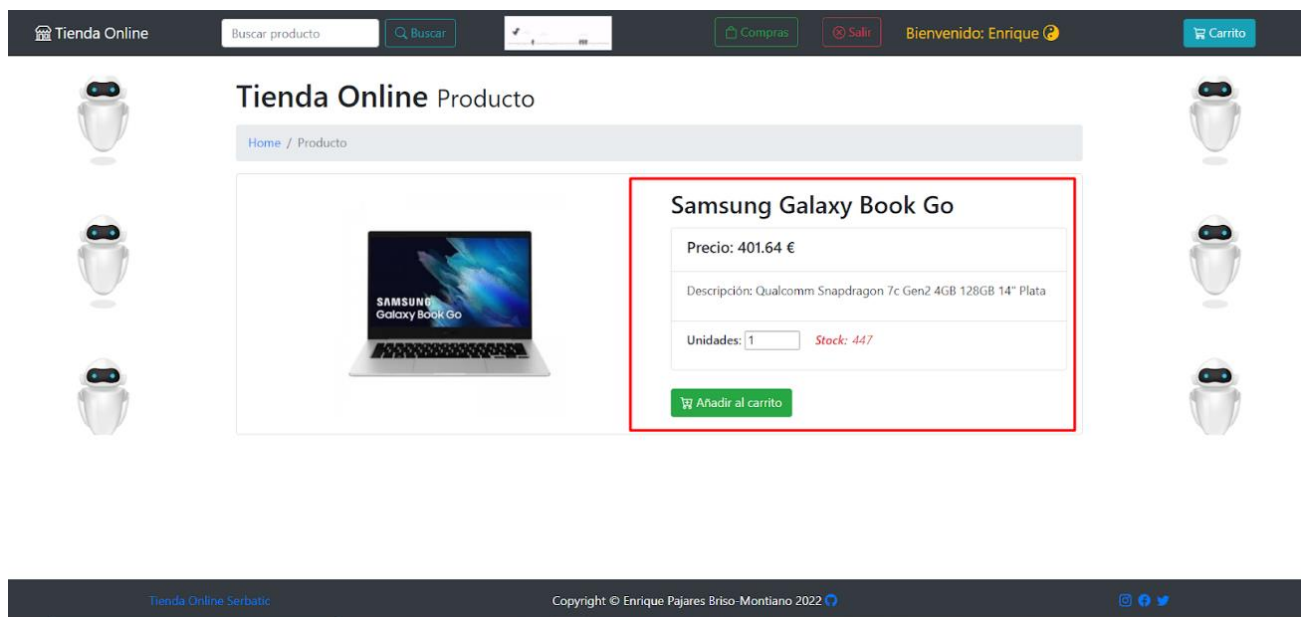
N° de Factura	Valor Total	Estado	Método de pago	Fecha	Ver	Cancelar	Exportar
S/N	401.64	PC	tarjeta	2022-05-23	Detalle		
S/N	1605.89	PE	paypal	2022-05-23	Detalle	Cancelar	
FRA-2022-05-23T21:12:47.854	2299.5	E	paypal	2022-05-23	Detalle		PDF
S/N	1801.37	PE	tarjeta	2022-05-24	Detalle	Cancelar	

Footer: Tienda Online Serbatic, Copyright © Enrique Pajares Briso-Montiano 2022, Social media icons.

7.1.12. Vista producto

En la vista de detalle de un producto se puede elegir las unidades de ese producto, limitado desde 1 unidad hasta 10 para que el cliente no pueda colapsar un producto, y se puede añadir al carrito desde su propio botón.

Se muestra y controla el stock de ese producto.



7.1.13. Vista carrito

En la vista del carrito se muestran los productos añadidos al mismo. El cliente puede eliminar el producto que ya no le interese y puede continuar con el proceso de compra a través del botón “Ver pedido”.

Se muestra la suma total de los productos del carrito.

Tienda Online Carrito

Home / Carrito

Producto	Precio	Unidades	Total	Acción
PcCom Gold Elite	1114.47	1	1114.47	Quitar
HP OMEN Sequencer	189.95	1	189.95	Quitar
Razer Viper Ultimate	124.79	1	124.79	Quitar
Razer Blackshark V2 Special Edition	102.17	1	102.17	Quitar
Newskill Icarus RGB IC27QRC	269.99	1	269.99	Quitar

TOTAL
Total: 1801.37 €
[Ver Pedido](#)

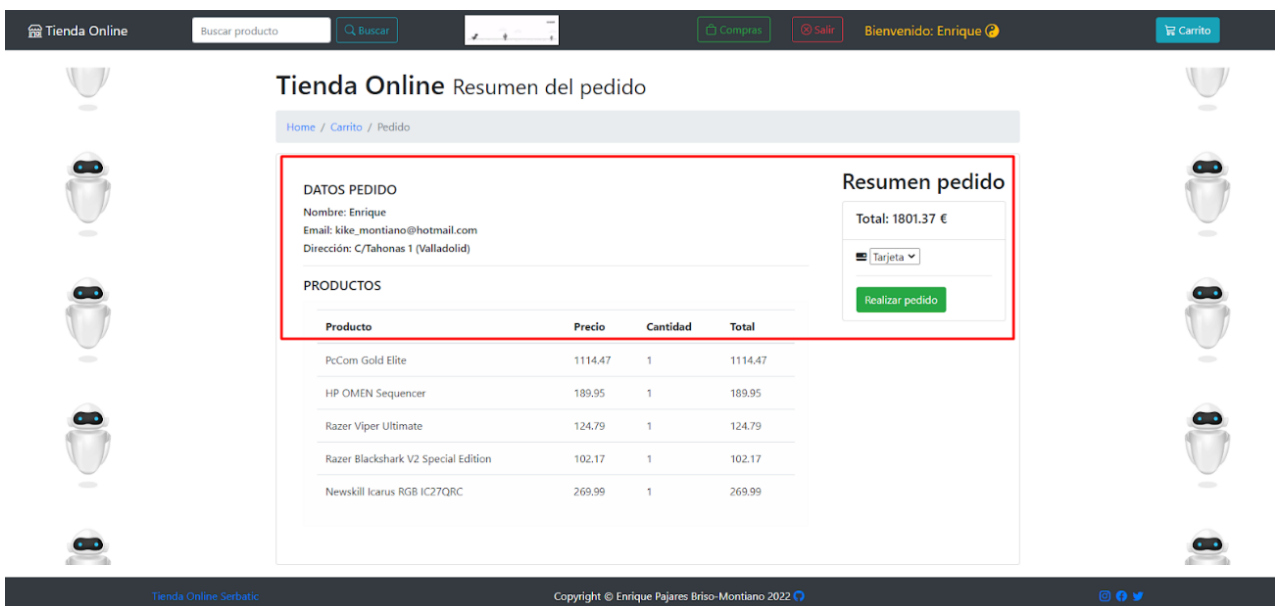
Tienda Online Serbiatic Copyright © Enrique Pajares Briso-Montiano 2022

7.1.14. Vista pedido

En la vista final del proceso de compra de un pedido el cliente debe elegir el método de pago que desee.

Se muestran los datos del pedido que son: los datos del cliente para su envío, los productos seleccionados anteriormente, el total y el método de pago.

Para finalizar la compra el cliente debe pulsar el botón de “Realizar pedido” una vez comprobado todos los datos.



Tienda Online Resumen del pedido

Home / Carrito / Pedido

DATOS PEDIDO

Nombre: Enrique
Email: kike_montiano@hotmail.com
Dirección: C/Tahonas 1 (Valladolid)

Resumen pedido

Total: 1801.37 €

Tarjeta ▼

Realizar pedido

Producto	Precio	Cantidad	Total
PcCom Gold Elite	1114.47	1	1114.47
HP OMEN Sequencer	189.95	1	189.95
Razer Viper Ultimate	124.79	1	124.79
Razer Blackshark V2 Special Edition	102.17	1	102.17
Newskill Icarus RGB IC27QRC	269.99	1	269.99

Tienda Online Sorbatic Copyright © Enrique Pajares Briso-Montiano 2022

Nota: Una vez acabado el procedimiento de compra, el administrador deberá comprobar todos los pedidos registrados y editar su “estado” a “E” de enviado para finalizar el proceso.

El sistema enviará automáticamente los datos al correo del cliente y éste ya podrá ver el número de factura de la compra y descargarla desde la aplicación.