



GRADO EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2019/2020

Trabajo Fin de Grado

DISEÑO Y DESARROLLO DE APLICACIONES
WEB DE ÚLTIMA GENERACIÓN

Aplicación web de comercio electrónico para una tienda de electrodomésticos

Autor : Sandra Álvarez Gancedo

Tutor : Dr. Gregorio Robles

Trabajo Fin de Grado

Aplicación web de comercio electrónico para una tienda de electrodomésticos

Autor : Sandra Álvarez Gancedo

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 2020, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2020

*Un esfuerzo total
es una victoria completa
Mahatma Gandhi*

Agradecimientos

En primer lugar quiero agradecer a mis padres, pues sin ellos y sin su esfuerzo no hubiese sido capaz de cerrar este ciclo. Muchas han sido las noches que habéis estado a mi lado, aguantando mis desvelos, mis nervios y mis miedos. Habéis sido el consuelo y el apoyo que necesitaba en mis momentos más duros, en los que únicamente un simple gesto vuestro hacía levantarme. Sois y seréis mi mejor ejemplo a seguir.

La vida me ha regalado unos segundos padres, mis tíos, los cuales me quieren y me protegen como a una hija. Me habéis acompañado en los momentos más felices, pero también en los más difíciles, incluso cuando a mis padres les faltaban fuerzas para seguir adelante. Simplemente deciros, gracias.

Mi primo, más que un hermano. Cuantos momentos vividos, siempre liándola, en las buenas y las no tan buenas, pero siempre juntos. Nos separamos en esta etapa universitaria, pero nuestras carreras eran tan iguales que nos ha permitido seguir ayudándonos.

Quiero agradecer a mis abuelos, a mi abuelo por enseñarme lo que es la vida. Por enseñarme que de todo lo malo también hay que saber sacar el lado positivo. A mi abuela, porque sé que desde algún rincón del cielo ha visto como lograba alcanzar todos mis sueños y mis metas. Pero sobre todo sé, que desde ahí arriba, me ha ayudado a luchar y a salir hacia delante ante el mayor reto de mi vida. Sé que estarás orgullosa de mí.

La universidad me ha aportado muchas cosas buenas, pero gracias a ella, conocí a quien es hoy uno de mis pilares, Alejandro. Se ha convertido en mi mejor amigo, mi confidente, mi compañero de viaje, mi otra mitad. Gracias por todo lo que has hecho y haces por mí. Me has ayudado en la universidad, pues sin ti, no sé si hubiese sido capaz de aprobar aquella asignatura que me traía por el camino de la amargura. Pero, sobre todo, me has ayudado en la vida, te quedaste a mi lado en el peor momento, cuando a todos nos invadía la incertidumbre, cuando ninguno sabía que iba a pasar, ahí estuviste conmigo. Me has dado fuerzas cuando más

flaqueaba, me has sacado una sonrisa cuando más lo necesitaba, y me has dado entereza para asumir y afrontar lo que me estaba pasando. Ha habido muchos momentos malos durante estos 5 años, pero juntos los hemos superado y así seguiremos haciéndolo.

Agradecer a mis compañeros y amigos de universidad por todos los apuntes y conocimientos intercambiados.

A mis profesores, en especial a Antonio García Marqués y Carlos Figueras, pues me ayudaron, me apoyaron y me dieron facilidades para poder sacar adelante la asignatura PDAC, a pesar de mis condiciones. Recuerdo, con cariño, el *email* de Antonio para comunicarme que había conseguido aprobar la asignatura y felicitarme por ello, pues sabía del esfuerzo realizado.

Quiero hacer mención especial a mi tutor de proyecto, Gregorio Robles, por darme ánimos para terminar el Trabajo Fin de Grado. Pero, sobre todo, gracias por ser paciente, por entender y comprender mi situación.

Y, por último, no me puedo olvidar de todos los médicos y enfermeras que me han atendido durante estos cinco años, puesto que, sin su esfuerzo, su investigación y su predisposición, no hubiese podido terminar esta etapa.

Agradecer a la vida, porque a pesar de todo he tenido suerte. Suerte por tener a la gente que tengo a mi lado, por haber podido estudiar, pero, sobre todo, suerte porque no todos pueden decir que han superado una enfermedad, como es un linfoma. Y, por qué no, gracias a ti, linfoma, porque me has enseñado a disfrutar y vivir la vida al máximo y a dar importancia a las cosas que realmente las tienen.

Resumen

El Trabajo Fin de Grado tiene como objetivo principal desarrollar una aplicación web de comercio electrónico para la venta de electrodomésticos.

Gracias a aplicaciones como la que se desarrolla en este trabajo, pequeños comercios pueden ampliar su mercado y llegar a más gente, pues simplemente accediendo a la web se puede ver la variedad de productos que hay en el catálogo, conocer su precio y realizar la compra sin tener la necesidad de ir a la tienda física.

Destacar que además de la venta de electrodomésticos, la aplicación permite al administrador, dueño del comercio, gestionar los clientes, los productos, las ofertas, los pedidos y las facturas. Pensando en esta gestión, se implementa un sistema de importación y exportación de ficheros Excel que permite al administrador cargar productos de manera masiva sin tener que ir uno a uno, agilizando el trabajo del administrador.

La aplicación está diseñada para que cualquier usuario pueda navegar por ella, sin ningún tipo de dificultad. Se han adaptado las distintas vistas a las resoluciones de pantalla más comunes, de tal forma que se pueda manejar tanto en un móvil como en un ordenador, sin que se produzca ningún menoscabo de la usabilidad. Esto ha propiciado que para el desarrollo de la aplicación se hayan elegido las siguientes tecnologías: HTML5, CSS3, Bootstrap, Angular 7 y TypeScript (para la parte del cliente) y Node.js, MongoDB y JavaScript (para la parte del servidor).

Summary

The main objective of the project is to develop an e-commerce web application for the sale of household appliances.

Thanks to applications such as the one developed in this work, small shops can expand their market and reach more people, because by simply accessing the web you can see the variety of products in the catalogue, know their price and make the purchase without having to go to the physical shop.

It should be noted that in addition to selling household appliances, the application allows the administrator, owner of the shop, to manage customers, products, offers, orders and invoices. With this management in mind, an Excel file import and export system is implemented that allows the administrator to load products en masse without having to go one by one, speeding up the administrator's work.

The application is designed so that any user can browse it, without any difficulty. The different views have been adapted to the most common screen resolutions, so that it can be used both on a mobile phone and on a computer, without any deterioration in usability. This has meant that the following technologies have been chosen for the development of the application: HTML5, CSS3, Bootstrap, Angular 7 and TypeScript (for the client side) and Node.js, MongoDB and JavaScript (for the server side).

Índice general

1. Introducción	1
1.1. Estructura de la memoria	2
2. Objetivos	4
2.1. Objetivo general	4
2.2. Objetivos específicos	4
2.3. Motivación	5
3. Estado del arte	6
3.1. HTML5	6
3.2. CSS3	7
3.3. JavaScript	8
3.4. TypeScript	9
3.5. Bootstrap 4	10
3.6. Angular	11
3.6.1. Angular CLI	13
3.6.2. Angular Material	14
3.6.3. Angular Editor	14
3.7. Node.js	14
3.8. Npm	15
3.9. Express.js	15
3.10. Nodemon	16
3.11. MongoDB	17
3.12. Mongoose	18

3.13. Bcryptjs	19
3.14. Passport.js	19
3.15. Git	20
3.16. Visual Studio Code	20
3.17. Mongo Compass	21
4. Diseño e implementación	22
4.1. Arquitectura general	22
4.2. Modelo de la base de datos	24
4.3. Diseño e implementación del servidor	30
4.3.1. Estructura del servidor	30
4.3.2. API REST de la aplicación	32
4.3.3. Registro y autenticación de usuarios	39
4.3.4. Sistema de subida de archivos en el servidor	41
4.3.5. Generación de PDFs	42
4.3.6. Sistema de importación y exportación	43
4.3.7. Envío de correos	44
4.4. Diseño e implementación del cliente	44
4.4.1. Estructura del cliente	44
4.4.2. Vistas	47
4.4.3. Uso de Angular Material	54
4.4.4. Descarga de archivos	55
4.4.5. Integración con Paypal	55
4.4.6. Sistema de avisos	57
4.4.7. Diseño <i>responsive</i>	58
5. Resultados	60
5.1. Consecución de objetivos	60
6. Conclusiones	62
6.1. Aplicación de lo aprendido	62
6.2. Lecciones aprendidas	63

6.3. Trabajos futuros	64
A. Manual de Instalación	66
B. Manual de usuario	68
B.1. Registro de un nuevo usuario	68
B.2. Inicio de sesión	69
B.3. Restablecer contraseña	71
B.4. Cerrar sesión	73
B.5. Perfil de usuario	74
B.6. Realizar compra	76
B.7. Tus pedidos	86
B.8. Añadir reseña	86
B.9. Preguntas frecuentes	88
B.10. Quiénes somos	89
B.11. Condiciones de uso	90
B.12. Política de privacidad	90
B.13. Trastienda	92
B.13.1. Pedidos	92
B.13.2. Facturas	94
B.13.3. Clientes	96
B.13.4. Productos	98
B.13.5. Marcas y Categorías	101
B.13.6. Ofertas	102
B.13.7. Servicios	104
B.13.8. Mensajes	105
B.13.9. Importación	107
Bibliografía	109

Índice de figuras

4.1. Ejemplo arquitectura Three-Tier.	22
4.2. Ejemplo arquitectura de tipo MEAN.	23
4.3. Diseño de la base de datos utilizando una base de datos relacional.	29
4.4. Estructura de los directorios del servidor.	31
4.5. Ejemplo PDF factura.	42
4.6. Estructura de los directorios del cliente.	45
4.7. Ejemplo de una tarjeta resumen del producto (Componente client-product). . .	48
4.8. Ejemplo fila de tarjetas resumen del producto (Componente product-row). . . .	48
4.9. Ejemplo detalle del producto (Componente Product-detail-client).	49
4.10. Ejemplo tarjeta subcategoría (Componente Product-detail-client).	49
4.11. Proceso de pago con Paypal.	56
4.12. Ejemplo cargo en la cuenta del cliente y recibí en la cuenta de la tienda.	57
4.13. Ejemplos mensajes de aviso.	57
4.14. Diseño <i>responsive</i> en móvil	59
4.15. Diseño <i>responsive</i> en ordenador	59
B.1. Enlace entrar para registrar usuarios.	68
B.2. Formulario registro usuarios.	68
B.3. Error al registrar un usuario.	69
B.4. Mensaje de confirmación de cuenta.	69
B.5. Enlace entrar para iniciar sesión.	70
B.6. Formulario para iniciar sesión.	70
B.7. Error al iniciar sesión.	70
B.8. Página principal de la tienda.	71

B.9. Enlace para restablecer contraseña.	71
B.10. Formulario para restablecer la contraseña.	72
B.11. Correo no válido al restablecer contraseña.	72
B.12. Formulario cambio de contraseña.	73
B.13. Error cambio de contraseña.	73
B.14. Enlace Salir.	73
B.15. Barra de navegación acceso al perfil de usuario.	74
B.16. Perfil de usuario.	74
B.17. Formulario para los datos de la dirección de envío.	75
B.18. Error al actualizar los datos personales.	75
B.19. Página principal de la tienda.	76
B.20. Menú categorías generales.	76
B.21. Subcategorías.	77
B.22. Productos de la subcategoría.	77
B.23. Ordenación de productos.	78
B.24. Filtros.	78
B.25. Detalle del producto.	79
B.26. Reseñas del producto.	80
B.27. Carrito.	81
B.28. Perfil de usuario.	81
B.29. Métodos de pago.	82
B.30. Formulario tarjeta de crédito.	82
B.31. Error al finalizar pedido con tarjeta de crédito.	83
B.32. Método de pago. Contra reembolso.	83
B.33. Método de pago. Paypal.	84
B.34. Inicio sesión en Paypal.	84
B.35. Pago con Paypal.	85
B.36. Enlace pedidos.	86
B.37. Tabla de pedidos de un usuario.	86
B.38. Enlace para escribir una reseña.	87
B.39. Ventana para publicar la reseña.	87

B.40. Enlace preguntas frecuentes.	88
B.41. Preguntas frecuentes.	88
B.42. Envía su pregunta.	88
B.43. Enlace quiénes somos.	89
B.44. Quiénes somos.	89
B.45. Enlace condiciones de uso.	90
B.46. Condiciones de uso.	90
B.47. Enlace política de privacidad.	91
B.48. Política de privacidad.	91
B.49. Barra de navegación del administrador.	92
B.50. Menú.	92
B.51. Sección Pedidos.	93
B.52. Barra de herramientas.	93
B.53. Iconos del registro.	93
B.54. Ficha del pedido.	94
B.55. Sección Facturas.	95
B.56. Barra de herramientas de Facturas.	95
B.57. Iconos de una factura.	95
B.58. Ficha de la factura.	96
B.59. Sección Clientes.	97
B.60. Barra de herramientas de Clientes.	97
B.61. Iconos de un cliente.	97
B.62. Ventana de confirmación para eliminar un cliente.	97
B.63. Ficha del cliente.	98
B.64. Sección Productos.	99
B.65. Barra de herramientas de productos.	99
B.66. Iconos de un Producto.	99
B.67. Ficha del producto.	100
B.68. Sección Marcas y Categorías.	101
B.69. Barra de herramientas de marcas y categorías.	101
B.70. Añadir una marca y una categoría.	102

B.71. Icono de una marca y una categoría.	102
B.72. Sección Ofertas.	102
B.73. Barra de herramientas de ofertas.	103
B.74. Iconos de una oferta.	103
B.75. Ficha de la oferta.	103
B.76. Sección Métodos de pago y empresas de transportes.	104
B.77. Barra de herramientas de métodos de pago y Empresas de transportes.	104
B.78. Añadir un método de pago y una empresa de transporte.	105
B.79. Icono de un método de pago y una empresa de transporte.	105
B.80. Sección Mensajes.	106
B.81. Barra de herramientas de mensajes.	106
B.82. Iconos de un mensaje.	106
B.83. Detalle del mensaje.	107
B.84. Sección Importación.	107
B.85. Log de importación.	108

Capítulo 1

Introducción

El comercio es el intercambio de mercancías y géneros en el mercado de la compra-venta.

Si nos remontamos a la antigüedad, se puede ver cómo los primeros hombres ya usaban un sistema de trueque para intercambiar pieles, alimentos o animales. Este sistema de trueque ha ido evolucionando conforme iba avanzando la humanidad, surgiendo el comercio tal y como se conoce actualmente.

Gran parte de esta evolución ha venido dada por el avance de la tecnología, ya que aunque no ha supuesto un gran cambio en el concepto de comercio sí lo ha hecho en la manera de comercializar. Hace unos años no se entendía el comercio sin tener que desplazarse a una tienda física.

Con el nacimiento de Internet, aparecieron las primeras transacciones comerciales *online*. Fue en el año 1981 cuando se realizó la primera de todas ellas. A pesar de ello, no empezó a adquirir popularidad entre las grandes compañías de comercio hasta el año 1994, cuando Netscape implantó SSL, que proporcionaba mayor seguridad en las transacciones, y cuando empezaron a surgir los primeros servicios de terceros para el procesamiento en línea de tarjetas de crédito. Muchas de estas compañías invirtieron grandes fortunas, pensando en el boom que tendría este tipo de comercio. Pero no fue hasta el año 2004 cuando empezó a adquirir popularidad, ya que la aparición del *Consejo de Normas de Seguridad* para las tarjetas de pago daba mayor seguridad a la hora de realizar las compras.

A pesar de todo, las personas seguían siendo muy reticentes a comprar por Internet por el miedo a que les clonasen sus datos bancarios. La creación de pasarelas de pago totalmente seguras, como la implantación, por parte de las entidades financieras, de tarjetas de crédito

recargables hizo que la gente fuese perdiendo el miedo y que el comercio electrónico despegara como esperaban las grandes compañías. Además, el uso de los dispositivos móviles, como *smartphones* o *tablets*, así como la aparición de las redes sociales, donde te permiten publicar anuncios para promocionar productos y servicios, ha provocado que el comercio no se entienda sin el comercio electrónico. Teniendo en cuenta la importancia de los dispositivos móviles en este tipo de ventas *online*, se entiende que todas las tiendas quieran adaptar sus aplicaciones web a estos dispositivos y plataformas.

El comercio electrónico, además de suponer un canal adicional de ventas, tiene la ventaja de permitir nuevos mercados y ampliar posibilidades de negocio, lo que hace que aquellas tiendas pequeñas en las que todavía no se ha implantado este tipo de comercio quieran hacerlo. De esta necesidad, surgió la idea de realizar este proyecto, ya que me permitía ampliar mis conocimientos tecnológicos.

1.1. Estructura de la memoria

La memoria comienza con un resumen, en el cual se explica de manera breve en que consiste el proyecto, así como las tecnologías que se han usado para desarrollarlo. A continuación, se sitúan los agradecimientos y los dos índices, el índice general y el índice de figuras. Por último, van los 6 capítulos en los que se divide la memoria y los apéndices.

- **Capítulo 1: Introducción.** Se describe de manera general en que consiste el proyecto. Especificándose cuál es la estructura de la memoria.
- **Capítulo 2: Objetivos y motivaciones.** Se describen los objetivos que se pretenden alcanzar con el desarrollo y finalización del proyecto. Se detallan las motivaciones por las cuales se ha decidido realizar una aplicación web de comercio electrónico.
- **Capítulo 3: Estado del arte.** Se explican las tecnologías usadas en el desarrollo de la aplicación.
- **Capítulo 4: Diseño e implementación.** Se describe de manera detallada la arquitectura, el diseño e implementación del servidor, el diseño de la base de datos y el diseño e implementación del cliente.

- **Capítulo 5: Resultados.** Se analiza si se ha conseguido alcanzar el resultado esperado.
- **Capítulo 6: Conclusiones.** Se detallan las conclusiones obtenidas con el desarrollo de la aplicación, los conocimientos que se han afianzado y adquirido. Y, por último, las posibles líneas futuras que se podrían desarrollar para mejorar la aplicación.
- **Apéndice A: Manual de instalación.** Se explica todo lo que se debe instalar para que la aplicación funcione correctamente en cualquier equipo, siempre en un entorno local.
- **Apéndice B: Manual de usuario.** Se describen los pasos que el cliente debe seguir para la finalización de la compra *online*.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo principal de este Trabajo Fin de Grado es el desarrollo de una aplicación web de comercio electrónico.

A través de la aplicación se podrán realizar compras de productos electrónicos sin necesidad de ir a una tienda física y se llevará a cabo la administración de dicha tienda *online* por parte del dueño o administrador del comercio.

2.2. Objetivos específicos

Para cumplir con el objetivo principal es necesario alcanzar los siguientes objetivos específicos:

- La aplicación web debe ser *responsive*, esto quiere decir que se debe adaptar a cualquier tipo de resolución de pantalla sin perder funcionalidad, ya que los clientes realizarán sus compras a través de diferentes navegadores y dispositivos móviles.
- La interfaz de la aplicación web deber ser sencilla, de tal forma que realizar una compra sea rápido y sencillo para cualquier usuario, incluso para aquellos que no tienen ningún tipo de conocimiento en informática y comercio electrónico.
- Puesto que se van a tratar datos personales susceptibles, la aplicación debe cumplir con los requisitos básicos de protección de datos. Además, debe ser robusta y segura a la hora

de realizar los pagos.

- La competitividad con otras aplicaciones web de comercio electrónico de electrodomésticos vendrá determinada por una interfaz atractiva y por funcionalidades como la calificación y valoración de los productos por parte de los clientes.
- La aplicación debe ser lo más modulable posible. Esto permitirá que, ante un cambio en los requerimientos y necesidades de la tienda, la aplicación se adapte de manera fácil y rápida.
- Diseñar un sistema de administración que permita al dueño del comercio llevar un control de los clientes, los productos, los pedidos y las facturas, así como de los proveedores de productos y del transporte de los mismos.
- Implementar un sistema de importación y exportación de ficheros Excel que permita el manejo de grandes volúmenes de datos.
- Implementar un sistema de mensajería que permita la comunicación entre los clientes y el administrador.
- Generación de las facturas en PDF para que se puedan enviar junto a los productos.

2.3. Motivación

En la actualidad, las tiendas *online* están en pleno auge y son un pilar fundamental para el ámbito comercial, cada vez más personas deciden realizar sus compras a través de internet. Este hecho, junto con la motivación de realizar un proyecto ambicioso, fue lo que hizo que desarrollase esta aplicación.

Este desarrollo me permitía poner en práctica la mayoría de los conocimientos adquiridos en el transcurso del grado. También, me permitía adquirir nuevos conocimientos que me servirán para adentrarme en el mundo laboral del comercio electrónico. Suponía una motivación más enfrentarme a nuevos retos aprendiendo tecnologías nuevas como Node.js, Angular, TypeScript o MongoDB y me servía para demostrarme que soy capaz de resolver cualquier conflicto que se plantee durante el transcurso del desarrollo con esfuerzo y dedicación, aumentando así mi confianza en afrontar nuevos proyectos de cara a mi carrera profesional.

Capítulo 3

Estado del arte

Para el desarrollo de este Trabajo Fin de Grado se han utilizado diferentes tecnologías, las cuales se explicarán a continuación.

3.1. HTML5

HTML5 (HyperText Markup Language) [26] es la quinta y última versión de HTML, lenguaje estándar utilizado para la definición y estructuración de los contenidos de las páginas web.

Se trata de una versión muy diferente a versiones anteriores, ya que su desarrollo se dio por discrepancias entre diferentes miembros del consorcio W3C. En el año 2004, WATWG (Web HyperText Application Technology Working Group) sacó una primera versión, para el asombro de todos. Pero fue en el año 2009, y en colaboración con WATWG, cuando W3C comenzó con el desarrollo de la especificación de HTML5.

Fue creado con la intención de sustituir a la versión anterior de HTML (HTML4), pero también se pretendía que la aparición de esta nueva versión relevará otros lenguajes como XHTML1 y DOM Nivel 2.

Esta versión, además de incluir nuevas etiquetas y atributos, proporciona una plataforma de desarrollo para la creación de aplicaciones web complejas. Entre las nuevas funcionalidades se encuentran los siguientes puntos:

- Incorporación de elementos multimedia para reproducir sonidos y vídeos desde el propio navegador. Estos elementos multimedia son las etiquetas `< audio >` y `< video >`.

- Integración de gráficos vectoriales escalables (SVG).
- Incorporación del elemento `< canvas >`, el cual permite dibujar y recrear animaciones en el navegador.
- Almacenamiento local en el lado del cliente, con `localStorage` y `SeassionStorage`.
- Incorporación de etiquetas que permiten la estructuración de los documentos HTML, sin recurrir al uso de `< div >`. Entre estas etiquetas se encuentran `< header >` y `< footer >`, muy útiles para la creación de la cabecera y el pie de la página HTML.
- Incorporación de `MathML` para el manejo de fórmulas matemáticas en los documentos web.

3.2. CSS3

CSS3 (Cascading Style Sheets) [8] es la última versión de CSS. Es un lenguaje de programación que permite definir y crear el aspecto visual de los documentos web.

Surgió de la necesidad de separar el contenido de los documentos HTML5 de su estilo, permitiendo crear documentos web más simples. Además, permite que un mismo documento web tenga varias hojas de estilo, facilitando así, por ejemplo, el diseño *responsive*, ya que una misma información necesita estilos distintos para los diferentes tipos de pantalla. Del mismo modo que un documento puede tener varias hojas de estilo, una misma hoja puede ser reutilizada en varios documentos web, simplificando el trabajo de los desarrolladores.

Es un lenguaje simple, que se basa en reglas para definir los aspectos visuales del documento web. Como resumen, su propio nombre es capaz de proporcionar la información necesaria que define el lenguaje:

- Cascading: El estilo que se aplica al elemento padre del documento web también se aplica a los elementos hijos que contenga, propagándose en forma de cascada.
- Style: Es un lenguaje que define el estilo y el aspecto visual de los documentos HTML.
- Sheets: Los estilos se añaden en ficheros, denominados hojas, cuya extensión es `.css`. Estas hojas pueden ser utilizadas en diferentes documentos HTML, permitiendo así la simplicidad y flexibilidad en los documentos web.

Algunas de las novedades más importantes que introduce esta versión de CSS son: transiciones, sombras en cajas, gradientes, esquinas redondeadas, múltiples imágenes en fondo, opacidad o transparencia de los colores, entre otras.

El encargado de definir y mantener las especificaciones de este lenguaje es World Wide Web Consortium (W3C).

3.3. JavaScript

JavaScript es un lenguaje de programación interpretado, no necesita ser compilado para poder ejecutarse. Es orientado a objetos y eventos. JavaScript se creó para funcionar en un entorno limitado, es decir, el código JavaScript no puede acceder a los recursos del propio ordenador. A raíz de esto, los navegadores también definieron sus propias normas de seguridad, las cuáles determinan que un *script* no puede establecer conexión con páginas que pertenezcan a diferentes dominios.

Su nombre puede indicar que tiene relación con Java pero la realidad es muy distinta, ya que tanto la semántica como los propósitos son diferentes. Está basado en prototipos en vez de usar clases, que simulan muchas de las características que tienen las clases en los lenguajes de programación orientados a objetos como, por ejemplo, la herencia.

Unas de las principales características que lo difiere de otros lenguajes es el ámbito de las variables. En JavaScript, este ámbito es la función donde han sido declaradas y no el bloque en el cuál se declaran. Con la versión 6 se introdujo esta mejora, permitiendo que la variable fuese usada en el bloque, siempre y cuando se definiese utilizando el término *let*.

Es un lenguaje imperativo y estructurado. También es débilmente tipado y dinámico, es decir, una misma variable puede adoptar dos tipos de datos diferentes en distintos momentos, sin tener que redefinir la variable. Está formado en su totalidad por objetos, pudiéndose modificar las propiedades de dichos objetos en tiempo de ejecución, dotándolo así del dinamismo que lo define.

Es un lenguaje rápido y ligero, por lo que se convierte en uno de los lenguajes más usados en el mundo del desarrollo de aplicaciones web. Su uso, principalmente, está en el lado del cliente, pero con la aparición de tecnologías como Node.js, el uso de JavaScript en el lado del servidor está sufriendo un repunte significativo. El hecho de ser un lenguaje multiplataforma, que permite

utilizarse en Windows, Linux, Mac, así como en cualquier navegador, hace que aumente su popularidad y su uso.

Surgió de la necesidad de los desarrolladores de crear páginas web complejas y dinámicas, ya que HTML únicamente permitía crear documentos web estáticos. Fue desarrollado por Brendan Eich de la compañía Netscape y su primer nombre fue Mocha. Poco tiempo después, cambiaría su nombre original y pasaría a llamarse LiveScript. En el año 1995, Netscape y SUN Microsystems (creadores del lenguaje Java) reintroducirían el lenguaje en el mundo de la programación, como JavaScript. En el año 1997 se adopta como un estándar ECMA, denominándose ECMAScript. Desde entonces ha ido evolucionando y mejorando, creándose bibliotecas y *frameworks* que facilitan la programación. En el año 2015, se publicó ECMAScript 2015 o JavaScript 6, introduciendo características propias de las clases utilizadas en la programación orientada a objetos. La última versión hasta el momento es la versión 8, segunda versión que tiene un proceso de desarrollo abierto.

3.4. TypeScript

TypeScript [25] es un lenguaje de programación de alto nivel orientado a objetos, de código abierto y libre.

Su desarrollo estuvo a cargo de Microsoft, y en él participó Anders Hejlsberg (diseñador y creador de C-Sharp, Delphi o Turbo Pascal). Surgió de la necesidad de los desarrolladores de crear aplicaciones robustas y de gran tamaño que con JavaScript no se podían mantener debido a su escasa escalabilidad.

Su uso está permitido tanto en el lado del servidor con Node.js, como en el lado del cliente con Angular 2, *framework* de JavaScript.

TypeScript es un lenguaje escrito encima de otro lenguaje, por lo que comúnmente se dice que es un superconjunto de JavaScript. Esto significa que TypeScript se compila a JavaScript, permitiendo que cualquier código desarrollado en JavaScript funcione sin tener que modificar el código ni la aplicación. Mejora y arregla algunos problemas que tiene JavaScript y es por ello, que pone a disposición de los desarrolladores las funcionalidades disponibles en JavaScript ES6 y JavaScript ES7. Añade objetos basados en clase, facilitando la programación orientada a objetos. En su código se puede incluir ficheros de definición que contienen información de las

bibliotecas JavaScript. Además, incluye un tipado estático. El poder definir variables y funciones tipadas sin perder la esencia de JavaScript ayuda al desarrollador a evitar errores en tiempo de ejecución.

Se publicó por primera vez en el año 2012 y, actualmente, se encuentra en la versión 2.0, la cual introduce numerosas características con respecto a la primera versión, entre las que destaca la capacidad de evitar la asignación de variables con un valor nulo.

Su compilador fue desarrollado en TypeScript, compilado a JavaScript y con licencia Apache 2.

3.5. Bootstrap 4

Bootstrap 4 [13] es la versión más actual y estable de Bootstrap. Es el *framework* más popular de CSS. Se trata de un conjunto de herramientas que permite a los desarrolladores mejorar el aspecto visual de los documentos HTML y de las aplicaciones web de manera rápida y sencilla.

Su nombre original fue Blueprint, fue creado y diseñado por Matt Otto para la compañía Twitter como herramienta de trabajo de uso interno. En el año 2011, Twitter lo publica con la licencia de software libre MIT y pasa a ser de código abierto.

El uso de este *framework* en las aplicaciones web permite generar diseños que se adaptan dinámicamente a las diferentes resoluciones de pantalla y a los distintos dispositivos, favoreciendo el diseño *responsive*. Para ello, Bootstrap dispone de un sistema de cuadrilla o rejilla estándar de 940 píxeles, pero dispone de cuatro variaciones de tamaño para adaptarse a los distintos dispositivos. Estas cuatro variaciones son *sm*, *md*, *lg* y *xl*.

Es modular y consiste en un conjunto de hojas de estilo LESS que combinado con plantillas HTML y con extensiones JavaScript implementan los diferentes componentes de Bootstrap.

Incluye los elementos más usados en los documentos HTML, como son formularios, barras de progreso y de navegación, cuadros, mensajes de alertas o botones con características especiales entre otros. Todos ellos tienen un estilo predefinido, pero que se puede modificar de manera sencilla.

Para usar Bootstrap en las aplicaciones web es necesario incluir en el documento HTML la hoja de estilo de Bootstrap CSS. De la misma forma, si se quiere utilizar algunos de los compo-

nentes JavaScript predefinidos de Bootstrap se debe incluir la biblioteca JQuery de JavaScript.

La versión 4 significa una mejora en diversos componentes permitiendo un diseño más *responsive*. Se han modificado componentes como la navegación, que incluye la nueva clase *flexbox*. Se ha creado un nuevo componente Card que sustituye a los componente Wells, Paneles y Thumbnails. Se han modificado los elementos de paginación y las modales. Y se incorpora el componente Grid. Además, de estas significativas mejoras, se han incluido numerosas clases de uso común.

Es compatible con todos los navegadores modernos, Firefox, Chrome, Safari, Opera, EDGE, Internet Explorer 10, pero no es compatible con las versiones anteriores a Internet Explorer 10.

3.6. Angular

Angular [9] es un *framework* de código abierto diseñado por Google. Permite crear aplicaciones web de una sola página, lo que se denomina SPA (Single Page Application). Utiliza TypeScript y HTML para el desarrollo de las aplicaciones web.

Angular permite separar el *frontend* del *backend* y sigue un modelo MVC (Modelo-Vista-Controlador), arquitectura de software que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Además, Angular dota a las aplicaciones de escalabilidad, ya que mantiene el código ordenado. De esta forma, las modificaciones y las actualizaciones de las aplicaciones son rápidas y sencillas.

Una de las principales ventajas de este *framework* y de las páginas SPA es la velocidad de carga entre las diferentes vistas de la aplicación. Cuando se cambia de vista no se recarga la página si no que éstas se cargan de manera dinámica, rápida y reactiva.

La arquitectura de una aplicación Angular se basa en clases de 4 tipos distintos (módulos, componentes, servicios y directivas). Estas clases se identifican a través de decoradores, los cuales permiten cargar los metadatos necesarios, que determinan a Angular como debe usar dichas clases.

Módulos.

Los módulos suministran el contexto de compilación de los componentes, es decir, es aquí donde se definen los diferentes componentes que va a formar la aplicación, las dependencias, las clases que actúan como servicios en la aplicación y las rutas de navegación que establecen

las vistas de la aplicación.

Toda aplicación angular tiene un módulo principal, que es el raíz. Este módulo suele recibir el nombre de AppModule y es el que inicia el sistema de arranque de la aplicación.

Al igual que en JavaScript, los módulos de Angular permiten importar y exportar funcionalidades proporcionadas por otros módulos, por lo que una aplicación puede tener más de un módulo, siendo cada uno de ellos independiente. La manera en que se organizan estos módulos ayuda a la realización de aplicaciones más complejas y a la reutilización de código. Además, permite que la carga inicial sea mínima, ya que cada módulo se carga a petición, es decir, cuando se va a utilizar.

Componentes.

Las aplicaciones Angular tienen un componente que es nexo de unión entre los diferentes componentes que forman la aplicación y el modelo de objeto del documento de la página (DOM). Los componentes son los que tienen la lógica y los datos de la aplicación. Son los que controlan las diferentes plantillas HTML que se cargan cuando se modifican las URLs dentro de la aplicación. Estas plantillas HTML son las vistas que forman la interfaz de usuario.

De la misma forma que las aplicaciones diseñadas con Angular pueden tener más de un módulo, también pueden tener subcomponentes, que se pueden relacionar entre sí mediante dos tipos de vinculación de datos, eventos y propiedades. A través de los eventos, la aplicación responde a las entradas del usuario actualizando los datos. A través de las propiedades, se envían valores calculados de los datos de la aplicación a los HTMLs que forman las vistas. Esta vinculación de datos se puede producir en ambas direcciones, por tanto, igual que los datos de la aplicación pueden modificar los HTMLs, los cambios en el DOM pueden modificar los datos de la aplicación.

El decorador @Component es el que determina que una clase actúe como un componente.

Servicios

Los servicios son clases en las que se definen datos o funcionalidades generales que no están asociadas a una determinada vista. Son usados para compartir datos y operaciones entre componentes. También es donde se suele realizar toda la operativa de las peticiones a la API de las aplicaciones. Deben llevar el decorador @Injectable, pues es el que permite obtener los metadatos necesarios para que otras clases puedan inyectar sus dependencias.

Directivas

Las directivas son clases en las que se definen los términos claves que se usan en las plantillas. Cuando se carga una vista, Angular procesa estos términos clave que modifican el HTML y el DOM de la aplicación. Por tanto, una plantilla, además de utilizar HTML para definir la vista, usa marcas propias de angular que son estas directivas.

Existen dos clases de directivas: (i) las de atributo, que modifican el comportamiento del componente, y (ii) las estructurales, que únicamente modifican la apariencia.

Angular dispone de un enrutador, módulo que contiene un servicio para definir las rutas de navegación de la aplicación. Este módulo se ajusta a las convenciones de los navegadores. Es decir, tanto si se pone la URL en la barra de direcciones, como si se pincha un enlace en la aplicación, como si se hace clic en el botón de retroceso o avance, se navega a la página correspondiente. El enrutador es el que muestra u oculta una vista. Cada vez que se modifica la URL, el enrutador se encarga de añadir la ruta en el historial del navegador, de ahí que los botones de retroceso y avance funcionen. Cada ruta de navegación está asociado a un componente.

Angular dispone de varias versiones, aportando todas ellas mejoras en dicho *framework*. La primera versión de Angular, se conoce como AngularJS, y es la más distinta a las demás. El resto son actualizaciones de Angular 2. Cuando se definió la segunda versión se decidió modificar el nombre del *framework* y dejarlo como Angular. La última versión estable es la versión 8.

3.6.1. Angular CLI

Angular CLI (Command Line Interface) [10] es una herramienta de línea de instrucciones creada por el equipo de Angular. Es muy útil a la hora de iniciar una aplicación web diseñada con Angular, ya que con una sola instrucción, se genera el esqueleto de carpetas y archivos necesarios de la aplicación. Además, contiene herramientas predefinidas que ayudan al desarrollo y mantenimiento de este tipo de aplicaciones. Al crear una aplicación web con Angular CLI, dentro de la estructura de archivos, se crea un archivo de configuración, en el cual se añaden las dependencias necesarias para que la aplicación web compile y ejecute. Este archivo de configuración se va modificando conforme se van creando los componentes, servicios o directivas. Para ello, Angular CLI tiene instrucciones que permiten crear estas clases de manera sencilla, creando los distintos archivos que conforman un componente, un servicio o una directiva. Entre las herramientas predefinidas destaca el compilador, el sistema de *testing* y el servidor web.

3.6.2. Angular Material

Angular Material [11] es una biblioteca creada por Google para las aplicaciones web desarrolladas con Angular. Al igual que Bootstrap, ayuda a diseñar aplicaciones web atractivas, con estilos y formas únicas, ya que contiene multitud de componentes con estilos prefijados. Estos componentes permiten a las aplicaciones web ser rápidas, consistentes y versátiles. Además, con el uso de esta biblioteca las aplicaciones web se adaptan mejor a la mayoría de resoluciones de pantalla y de dispositivos, facilitando el diseño *responsive* de las aplicaciones. También, con su uso se crean aplicaciones web accesibles, ya que los componentes cumplen con los estándares de accesibilidad. Además, esta biblioteca facilita la internacionalización de la aplicación, puesto que permite la utilización de diferentes idiomas. Angular Material es compatible con todos los navegadores modernos.

3.6.3. Angular Editor

Angular Editor es un componente de Angular [1], que permite escribir texto con formato y estilo propio dentro de un documento HTML. El estilo y el formato lo determina el usuario. El componente transforma el texto escrito, con el formato y el estilo definido, al lenguaje HTML visualizándose en pantalla conforme el usuario determina. Angular Editor es compatible únicamente con las últimas versiones de Angular (de Angular 6 en adelante).

3.7. Node.js

Node.js [19] fue creado en 2009 por Ryan Dahl como sistema de desarrollo de aplicaciones red escalables. Se trata de un entorno de ejecución orientado a eventos asíncronos que utiliza el motor V8 de Google, desarrollado en C++ y de código abierto, para interpretar el código JavaScript, del mismo modo que hace el navegador Chrome.

Se basa en sistemas como Event Machine de Ruby y Twisted de Python. Arranca cuando se ejecuta el *script* de entrada al bucle de inicio de eventos y finaliza cuando no hay más devoluciones de llamadas *callbacks*, utilizando así un único hilo. De este modo, cada nuevo evento se añade al bucle de eventos que se ejecuta al inicio, evitando el bloqueo de procesos que se produce en los sistemas de concurrencias basados en la utilización de hilos de los sistemas operativos.

Estos procesos usan un hilo por cada conexión provocando la ineficiencia de los recursos utilizados. A pesar de que Node.js esté pensado para usar un único hilo se pueden crear hilos diferentes o subprocesos a través de APIs propias de Node.js como `API_child.process.fork()`.

Por otro lado, Node.js es un buen aliado para desarrollar aplicaciones web, ya que a la hora de implementar el protocolo HTTP para Node.js se pensó tanto en la latencia de transmisión como en la transmisión por *streaming* de datos.

3.8. Npm

npm [22] es el sistema de administración de paquetes del entorno de ejecución de Node.js. Contiene un cliente con el que, ejecutando una línea de instrucciones, se instalan los paquetes necesarios para el funcionamiento de los proyectos Node.js. Además, contiene una biblioteca o base de datos con más de 477.000 paquetes disponibles para usarse. Todos estos paquetes están guardados con formato CommonsJs y contienen un archivo para almacenar metadatos con formato JSON. Fue desarrollado por Isaac Z. Schlueter y se desarrolló por completo en JavaScript.

Al instalar un paquete a través de *npm*, se instala bajo el directorio del proyecto `/.node_modules`. Permite instalar con una única instrucción todas las dependencias del proyecto, siempre y cuando se ejecute la instrucción en la raíz del proyecto. *npm* utiliza el archivo con formato JSON, denominado *package.json*, para guardar todos los metadatos relevantes a los módulos, paquetes y dependencias del proyecto. En el archivo *package.json* se guardan las versiones compatibles para los diferentes módulos, permitiendo hacer un versionado del módulo instalado. Gracias a esto, *npm* hace que la instalación de proyectos Node.js descargados de Git sea sencilla.

3.9. Express.js

Express [14] es el *framework* más usado para aplicaciones desarrolladas con Node.js. Se trata de un *framework* flexible, rápido, ligero y minimalista. Pero, a pesar de ello, existen numerosos plugins o *middlewares* compatibles con Express. Estos plugins permiten resolver los problemas que pueden surgir a la hora de implementar aplicaciones con Express. Por todo esto, Express sirve de base de otros muchos *frameworks* de Node.js.

Express permite:

- Crear aplicaciones Node.js de manera sencilla y rápida, ya que proporciona el esqueleto de las aplicaciones.
- Modificar ajustes de la aplicación como la localización de las plantillas que se van a utilizar o el puerto en el que se va a ejecutar.
- El uso de cualquier base de datos, siempre y cuando la base de datos esté soportada por Node.
- La gestión de cookies, usuarios y sesiones a través de su *middleware*.
- Configurar y establecer las rutas entre el *frontend* y la base de datos.
- Mejorar las funcionalidades HTTP que tiene Node.js y que están basadas en Connect. Existen métodos que determinan la función a usar indicando únicamente el verbo de la petición HTTP.

3.10. Nodemon

Nodemon [21] es una herramienta que permite a los desarrolladores de Node.js implementar sus aplicaciones de manera más sencilla, ya que amplía y mejora funcionalidades propias de Node.js.

Cuando se programa en Node.js, los cambios realizados en el código fuente no se ven reflejados inmediatamente, hasta que no se reinicia el servidor de Node.js no se actualiza la aplicación, por lo que cada vez que se prueba un cambio, habría que parar y arrancar el servidor, ralentizando el desarrollo de los proyectos. Nodemon reinicia de manera automática el servidor cada vez que detecta un cambio en el código fuente, solucionando el inconveniente anterior. Los cambios se ven reflejados en la aplicación inmediatamente, permitiendo programar y probar nuevas funcionalidades sin demoras.

3.11. MongoDB

MongoDB [17] es una base de datos no relacional, una base de datos NoSQL, que se basa en documentos para guardar la información.

Es una base de datos gratuita y multiplataforma (Linux, Windows, Os X y Solaris). Las distintas versiones se lanzaban bajo la licencia AGPL, pero a partir de octubre de 2018, se publican bajo la licencia pública SSPL, aunque tiene una línea comercial que contiene características más avanzadas.

A diferencia de las bases de datos relacionales, MongoDB almacena sus datos en documentos y no en tablas. Estos documentos son de esquemas libres, lo que quiere decir que cada entrada puede tener una estructura de datos distinta, con campos que no se tienen porque repetir en otro registro. Por eso, MongoDB tiene mayor flexibilidad y escalabilidad que las bases de datos relacionales. Los documentos se guardan en formato BSON (Binary JSON), lo que permite guardar índices de array, longitudes de campos y datos relevantes, que hacen las búsquedas más rápidas y eficientes. A pesar de que se almacenan en BSON, los desarrolladores siempre trabajarán con documentos JSON.

Los documentos se agrupan en colecciones y estas colecciones son lo que serían las tablas en las bases de datos relacionales.

MongoDB permite indexar cualquier atributo. Por defecto, el índice tiene un formato similar a un valor UUID hexadecimal. Está formado por una semilla basada en la MAC de la interfaz de la red de la máquina y por una marca de tiempo, que ocupa las primeras posiciones del índice. Gracias a esta marca de tiempo, los datos se ordenan por orden de creación. Todos los documentos tienen un índice, ya que es el único campo obligatorio dentro de un documento MongoDB.

MongoDB permite realizar:

- Consultas *ad-hoc*, es decir, se puede realizar búsqueda de datos a través de campos, de rangos o a través de expresiones regulares. Estas consultas nos pueden devolver un dato concreto o pueden devolver el documento entero.
- Balanceo de carga, para ello realiza una partición horizontal de los datos. Es el desarrollador quien decide cómo van a ser distribuidos los diferentes datos. Para llevar a cabo este balanceo de carga, MongoDB puede ejecutarse en diferentes servidores.

- Almacenamiento de archivos.
- Operaciones similares al *GroupBy* de SQL, a través de agregaciones. Estas agregaciones se implementan como pipeline en el que se van transformando los datos a través de etapas hasta conseguir el dato deseado. Contiene una operación MapReduce que se suele utilizar en estas agregaciones.
- Consultas a través del lenguaje JavaScript.

Como desventajas, MongoDB únicamente puede realizar operaciones concurrentes de escritura entre documentos distintos, ya que cada vez que se realiza una operación de escritura se bloquea el documento. Y, además, puede tener problemas de rendimiento si los datos superan los 100GB.

3.12. Mongoose

Mongoose [18] es un ODM (Object Document Mapping) de MongoDB, es decir, sirve para mapear objetos de JavaScript a documentos propios de MongoDB.

Es una biblioteca de JavaScript que se instala a través de una única instrucción *npm* en los proyectos que utilizan Node.js y MongoDB.

Para mapear los objetos a documentos, lo que hace es crear esquemas con datos fuertemente tipados. Estos esquemas se traducen en un modelo Mongoose, y es este modelo Mongoose el que se traduce en el documento MongoDB.

Contiene una infinidad de funciones para realizar búsqueda de datos, validación de datos, guardado y eliminado de datos, entre otras. Todas estas funciones se basan en las funciones más comunes que Mongo tiene para realizar estas operaciones.

Permite crear esquemas muy flexibles, de tal forma que se puede organizar la información de diferente manera, según convenga al desarrollador. Tanto es así, que permite crear esquemas con referencias a otros esquemas, asemejándose de esta forma, a lo que sería una relación entre tablas de una base de datos relacional como, por ejemplo, MySQL.

3.13. Bcryptjs

Bcrypt [12] es una biblioteca que se usa para el cifrado de contraseñas y datos importantes. Fue creada por Niels Provos y David Maxieres y está desarrollada con JavaScript.

Utiliza un valor aleatorio, denominado *salt*, para la generación del *hash* del dato que se desea cifrar. Este valor se guarda con el *hash* en la base de datos. Por eso, para el mismo valor se obtiene diferentes *hashes*. Gracias a esto, se evitan los ataques de fuerza bruta o los ataques denominados Rainbow table (tablas de textos y sus *hash* asociados). Bcrypt permite elegir el valor del SaltRound, ya que a mayor valor numérico del *salt* mayor seguridad, pero a mayor seguridad mayor coste de tiempo a la hora de cifrar los datos. Por lo tanto, lo idóneo es elegir un *salt* lo suficientemente grande para que no se pueda obtener el dato cifrado por fuerza bruta pero no lo suficientemente alto como para que el procesamiento se demore mucho tiempo. Por defecto, Bcrypt pone un *salt* de 10, un valor intermedio para que no se produzcan los dos problemas principales.

Bcrypt permite realizar comparaciones entre valores. Para realizar la comparación no descifra el dato a comparar, como se hace en otros sistemas, si no que cifra el valor con el *salt* del dato cifrado.

3.14. Passport.js

Passport [23] es un *middleware* que permite la autenticación en aplicaciones Node.js. Es de código abierto. Se puede incorporar en cualquier aplicación basada en Express, ya que trabaja en conjunto con Express y Connect. Permite gestionar y administrar las sesiones de los usuarios. Las características que definen Passport son:

- Tiene numerosas estrategias de autenticación. Por ejemplo, permite autenticarse utilizando bases de datos, en las que se almacenan las sesiones o puede autenticarse con el inicio de sesión único de OpenID y OAuth, usados por ejemplo, en Facebook.
- Permite desarrollar nuevas estrategias de autenticación.
- Permite usar sesiones persistentes.
- Gestiona el proceso de autenticación indicando si ha sido satisfactoria.

3.15. Git

Git [15] es un software de control de versiones. Fue diseñado por Linus Torvalds. Es un software libre que se distribuye a través de la versión 2.0 de la licencia GNU GPL.

Git permite el trabajo en equipo mediante su sistema de ramas, que permite el desarrollo no lineal de los proyectos. En cada rama se puede trabajar sobre una funcionalidad distinta, agilizando de esta forma el desarrollo del proyecto y haciendo que el trabajo en equipo sea más sencillo y óptimo. Además, permite gestionar de manera eficiente aplicaciones y proyectos de gran envergadura.

Los cambios en los proyectos se realizan de manera inmediata, ya que cada desarrollador tiene una copia en local (en su equipo) del proyecto. Esto permite navegar por el historial de cambios sin necesidad de conectarse a un servidor. Es decir, se puede trabajar en local hasta finalizar una tarea y, posteriormente, subirlo al repositorio. Así, todo el equipo obtiene el cambio en su proyecto local.

Cuando se usa Git en un equipo de trabajo se definen flujos que determinan como se va a realizar el desarrollo del proyecto. Es habitual que exista una rama *master*, que será la que se despliegue en producción. De esta rama *master* se obtiene una nueva rama, por ejemplo *develop*. En esta rama se añaden los cambios y las funcionales nuevas. Se mezcla con la rama *master* una vez testeados los cambios. Para funcionalidades específicas, se crean ramas que se obtienen de la *develop*, denominándose por ejemplo *features*. De esta forma, se permite el desarrollo en paralelo de los distintos integrantes del grupo. Una vez que la funcionalidad esté finalizada y probada se junta con la rama *develop*. Cuando el resto del equipo actualiza su rama *develop*, obtienen los cambios de las nuevas funcionalidades sin que su trabajo se vea afectado. Cuando hay un error en producción que deber ser solucionado inmediatamente, se crea una rama a partir de la *master*, llamada *hotfix*, donde se desarrollan las soluciones al error. Una vez implementada la solución al error, se mezcla con la rama *master*.

3.16. Visual Studio Code

Visual Studio Code es un editor de código fuente, lanzado en 2015 por Microsoft. Se trata de un editor multiplataforma, pudiéndose instalar en Windows, Linux y MacOS.

Soporta una gran cantidad de lenguajes de programación. El resaltado de sintaxis, la capacidad de finalización del código que se escribe, la refactorización del código, el modo depuración, así como la integración con Git facilita el desarrollo de aplicaciones.

Para hacer más atractiva su interfaz y para facilitar su uso, permite personalizar los temas del editor y los atajos del teclado.

Es un editor de código abierto bajo licencia de software libre MIT, lo que quiere decir que se puede descargar de GitHub y realizar cualquier modificación en su código. Se basa en Electron, *framework* utilizado para el desarrollo de aplicaciones de escritorio usando tecnologías web, utiliza Chromium para el motor gráfico y Node.js para ejecutar JavaScript.

3.17. Mongo Compass

Mongo Compass [16] es una herramienta gráfica, multiplataforma (Linux, Mac y Windows), que permite a los desarrolladores interactuar con un base de datos MongoDB. Las principales ventajas que tiene su uso son:

- Permite analizar los esquemas, visualizando la estructura de sus documentos, el tipo y los rangos de los campos de dichos documentos.
- Permite gestionar de manera sencilla los índices.
- Se pueden realizar todas las operaciones CRUD¹ de manera visual. De la misma forma, se pueden realizar búsquedas o aplicar filtros para encontrar datos dentro de los documentos del esquema.
- Ayuda a resolver problemas de rendimientos, ya que se pueden obtener análisis y gráficos de lo que tarda una *query* en ejecutarse.
- Se pueden crear agregaciones de manera intuitiva y sencilla, gracias a los esqueletos de código y al autocompletado. Estas agregaciones se exportan a código fuente para que se incorporen dentro del código de la aplicación que usa MongoDB.

¹Acrónimo de “Crear, Leer, Actualizar y Borrar” (del original en inglés: Create, Read, Update and Delete).

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

La aplicación implementada en este trabajo tiene una arquitectura *Three – Tier*, es decir, una arquitectura de tres niveles. El primer nivel de esta arquitectura es la capa de presentación o cliente. Este nivel está compuesto por las diferentes vistas de la aplicación que forman la interfaz de usuario. Es el encargado de recoger la información que produce el usuario al interactuar en la aplicación y el que envía, a través de peticiones, la información al segundo nivel. El segundo nivel es el servidor, donde se encuentra la lógica de negocio. Se encarga de recibir todas las peticiones de usuario enviadas por el cliente y de su posterior respuesta. Es el único nivel que se comunica con el resto, puesto que, también puede comunicarse con el tercer nivel, acceso a base de datos o servidor de base de datos. Este tercer nivel es el que maneja la información. En la figura 4.1 se representa un ejemplo de este tipo de arquitectura.

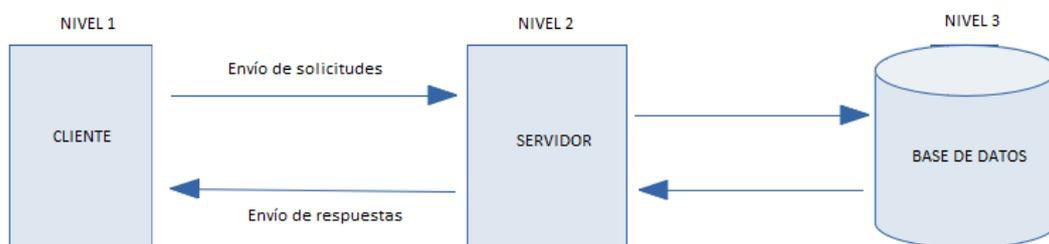


Figura 4.1: Ejemplo arquitectura Three-Tier.

Existen distintos tipos de arquitectura *Three – Tier*. La escogida para esta aplicación es

la denominada *MEAN*. Este tipo de arquitectura se denomina así porque utiliza MongoDB, Express, Angular y Node.js para el desarrollo de las aplicaciones. En la figura 4.2 se puede observar como es la comunicación entre el cliente y el servidor en una arquitectura de tipo MEAN.

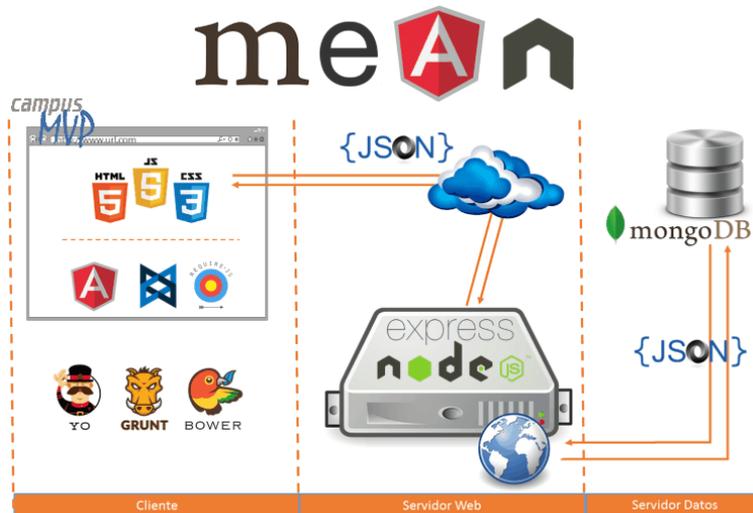


Figura 4.2: Ejemplo arquitectura de tipo MEAN.

Para la implementación del cliente se ha usado Angular, ya que permitía crear una aplicación Single-Page, haciendo uso del modelo-vista-controlador (MVC). Este tipo de aplicaciones se caracterizan por la rapidez y la fluidez de navegación, ya que no es necesario recargar la página cada vez que se cambia de vista. Angular utiliza TypeScript como lenguaje de programación, por lo que el cliente está programado con esta tecnología, además, de usar HTML5, CSS3 y Bootstrap. El cliente utiliza el servidor que proporciona Angular CLI para servir los recursos.

El servidor se ha implementado usando Node.js y Express, consiguiendo así, crear una aplicación rápida y escalable. Se ha programado utilizando JavaScript. Por su parte, el servidor se encuentra alojado en el servidor proporcionado por Node.js.

Para establecer la comunicación entre el cliente (servido por Angular CLI) y el servidor (servido por Node.js) se ha recurrido a la utilización de un *proxy*. En JavaScript, cuando se hacen peticiones a un dominio distinto al del origen del *script*, como es el caso, surge el problema de Cross Domain, el cual impide que se establezca la conexión entre el cliente y el servidor. En el caso de la aplicación, el *proxy* actúa de intermediario reenviando todo el tráfico que recibe Angular CLI en el contexto */nevado* al servidor e intercepta las respuestas que el servidor envía

al cliente. Con Angular CLI se configura de manera sencilla, ya que para configurarlo solo se necesita un fichero de configuración y ejecutar su servidor a través de la instrucción: `ng-serve -proxy-config proxy.conf.json`.

La aplicación necesita una base de datos para almacenar tanto los datos de los clientes, como los datos de los productos, facturas y pedidos. Se ha utilizado una base de datos no relacional, concretamente MongoDB.

4.2. Modelo de la base de datos

El gestor de base de datos utilizado es MongoDB, que es una base de datos no relacional. Se ha utilizado este tipo de base de datos por la flexibilidad que proporciona a la hora de trabajar con documentos. Además, como se ha comentado en la sección 3.11, los documentos almacenados no tienen porque tener las mismas propiedades, lo que resulta muy útil para el almacenamiento de los productos, ya que cada producto tiene propiedades diferentes. Están indicadas en aplicaciones en las que el volumen de datos crece constantemente. En una tienda *online*, tanto el número de usuarios como el número de pedidos está en constante crecimiento. El formato de los documentos de las distintas colecciones de la base de datos está definido en el subdirectorio `modelos` del servidor. Las colecciones de la aplicación se describen a continuación:

- **Usuarios:** Colección donde se almacenan los usuarios registrados de la tienda *online*. Los campos que tienen sus documentos son:
 - **correo:** Dirección de correo electrónico con el que se registra el usuario, debe ser único y siempre debe de estar presente en los documentos guardados. (String).
 - **rol:** Perfil del usuario registrado, puede ser cliente o admin. Dependiendo de este perfil se pueden visualizar unas vistas u otras. (String)
 - **bloqueado:** Indica si un usuario se ha bloqueado. El administrador de la tienda *online* puede bloquear usuarios en caso de detectar anomalías en su comportamiento. (Boolean)
 - **confirmacion:** Indica si un usuario está confirmado. Si el usuario no está confirmado no podrá realizar compras. (Boolean)

- **contrasena:** Contraseña cifrada introducida por el usuario para entrar en su cuenta. Es un campo que siempre debe ir relleno. (String)
 - **clientes:** Identificador del usuario en la tabla clientes. Cuando un usuario se registra se crea un documento nuevo en la colección clientes. (ObjectId)
- **Cientes:** Colección donde se almacena la información personal de un usuario registrado. Los campos son:
- **nombre:** Nombre del usuario registrado. (String)
 - **apellidos:** Apellidos del usuario registrado. (String)
 - **teléfono:** Teléfono personal donde dirigirse en caso de tener problemas con su pedido. (String)
 - **fechaNacimiento:** Fecha de nacimiento del usuario registrado. (String)
 - **direccionFactura:** Dirección de facturación. Calle, número, portal, escalera, piso y letra. (String)
 - **codigoPostalFactura:** Código postal de la dirección de facturación. (String)
 - **puebloFactura:** Municipio de la dirección de facturación. (String)
 - **provinciaFactura:** Provincia de la dirección de facturación. (String)
 - **direccionEnvio:** Dirección donde se envía el pedido. Sólo tendrá valor si la dirección de envío es diferente a la dirección de facturación. (String)
 - **codigoPostalEnvio:** Código postal de la dirección de envío. Sólo tendrá valor si la dirección de envío es diferente a la dirección de facturación. (String)
 - **puebloEnvio:** Municipio de la dirección de envío. Sólo tendrá valor si la dirección de envío es diferente a la dirección de facturación. (String)
 - **provinciaEnvio:** Provincia de la dirección de envío. Sólo tendrá valor si la dirección de envío es diferente a la dirección de facturación. (String)
 - **pedidos:** Número de pedidos realizados por el usuario en la tienda. (Number)
- **Artículos:** Colección donde se almacena la información de los productos.

- **codigoArticulo:** Identificador del artículo. Es único y debe estar presente en todos los documentos guardados. (String)
 - **categoria:** Grupo en el que se encuadra el artículo. (String)
 - **modelo:** Modelo del artículo. (String)
 - **marca:** Marca del artículo. (String)
 - **pvpTarifa:** Precio máximo del artículo (Number)
 - **pvp:** Precio del artículo con IVA. (Number)
 - **estado:** Indica si el producto es visible o no en el catálogo. (Boolean)
 - **stock:** Existencias del artículo en el almacén. (Number)
 - **imagen:** Ubicación donde se encuentra la foto del producto en el servidor. (String)
 - **características:** Descripción del artículo con las características principales. (String)
 - **otros:** Información adicional del artículo. (String)
 - **fechaPublicacion:** Fecha en la que se guarda por primera vez el artículo en el sistema. (Date)
 - **fechaModificacion:** Fecha en la que algún campo del documento es actualizado. (Date)
 - **autor:** Nombre de quien añade el artículo al sistema. (String)
 - **ventas:** Número de veces que el producto ha sido comprado. (Number)
 - **ofertas:** Identificador del descuento que se le aplica al producto si estuviese en oferta. (ObjectId)
 - **envioGratis:** Indica si el artículo se envía sin ningún coste. (Boolean)
 - **estrellas:** Número de estrellas puestas en las reseñas. (Number)
 - **usuarios:** Número de usuarios que han escrito una reseña sobre el producto. (Number)
 - **comentarios:** Array de identificadores de las reseñas del producto. (ObjectId)
- **Categorías:** Colección donde se almacenan las diferentes categorías donde se agrupan los productos.

- **nombre:** Nombre de la categoría. Es único y debe estar presente en todos los documentos. (String)
 - **imagen:** Ubicación donde se encuentra la foto en el servidor. (String)
 - **categoríaGeneral:** Grupo donde se encuadra la categoría. (String)
- **Marcas:** Colección donde se almacenan las marcas de los artículos.
 - **nombre:** Nombre de la marca. Es único y debe estar presente en todos los documentos. (String)
 - **imagen:** Ubicación donde se encuentra el logo de la marca en el servidor. (String)
- **Ofertas:** Colección donde se almacenan las ofertas disponibles en la tienda *online*.
 - **idOferta:** Identificador de la oferta, es único y siempre debe estar relleno. (String)
 - **descuento:** Descuento que se aplica en la oferta. Siempre en % (Number)
 - **fechaInicio:** Fecha en la que se empieza a aplicar la oferta. (Date)
 - **fechaFin:** Fecha límite de la oferta. (Date)
 - **descripcion:** Breve descripción del motivo de la oferta. (String)
- **Comentarios:** Colección donde se almacenan las reseñas escritas por los usuarios.
 - **calificacionGeneral:** Puntuación general que se le otorga al producto. (Number)
 - **titulo:** Título de la reseña (String)
 - **comentario:** Valoración del usuario al producto. (String)
 - **recomendacion:** Indica si el usuario aconsejaría comprar el producto. (Boolean)
 - **calidadPrecio:** Puntuación que se le otorga al producto en cuanto a calidad precio. (Number)
 - **alias:** Nombre del usuario que escribe la reseña. (String)
 - **email:** Correo del usuario que escribe la reseña. (String)
- **Método de Pago:** Colección donde se almacenan los métodos de pago con los que se pueden pagar los pedidos.

- **nombre:** Nombre del método de pago (String)
- **imagen:** Ubicación donde se encuentra el logo del método de pago. (String)
- **Transportes:** Colección donde se almacenan las empresas de transportes utilizadas para el envío de los pedidos.
 - **nombre:** Nombre de la empresa de transporte. (String)
- **Pedidos:** Colección donde se almacena la información relativa a los pedidos.
 - **idPedido:** Identificador del pedido, debe ser único. (String)
 - **fechaEntrada:** Fecha en la que se realiza el pedido. (Date)
 - **fechaEntrega:** Fecha aproximada de la entrega del pedido. (Date)
 - **estado:** Indica la situación del pedido: pendiente, preparando, entregado y finalizado. (String)
 - **total:** Precio total del pedido. (Number)
 - **formasPago:** Método de pago del pedido. (String)
 - **cobrado:** Indica si el pedido está pagado o no. (Boolean)
 - **usuario:** Identificador del usuario que realizó el pedido. (ObjectId)
 - **articulos:** Artículos que contiene el pedido. Array de objetos formados por el identificador del producto y cantidad del producto comprado. (Array)
- **Facturas:** Colección donde se almacena la información relativa a las facturas de los pedidos realizados.
 - **nFactura:** Identificador de la factura, debe ser único. (String)
 - **fecha:** Fecha en la que se crea la factura. (Date)
 - **cliente:** Identificador del usuario que realiza la compra. (ObjectId)
 - **total:** Precio total de la factura. (Number)
 - **formasPago:** Método de pago con el que se paga el pedido. (String)
 - **idPedido:** Identificador del pedido asociado a la factura. (String)

- **articulos:** Artículos que contiene el pedido. Array de objetos formados por el identificador del producto y cantidad del producto comprado. (Boolean)
- **Mensajes:** Colección donde se almacenan los mensajes enviados por parte de los usuarios al administrador.
- **idUsuario:** Identificador del usuario que escribe el mensaje. (ObjectId)
 - **nombreUsuario:** Nombre del usuario que escribe el mensaje. (String)
 - **correoUsuario:** Correo del usuario que escribe el mensaje. (String)
 - **mensaje:** mensaje que se envía al administrador. (String)
 - **leído:** Indica si un mensaje se ha leído o no. (Boolean)
 - **fecha:** Fecha en la que se envió el mensaje. (Date)

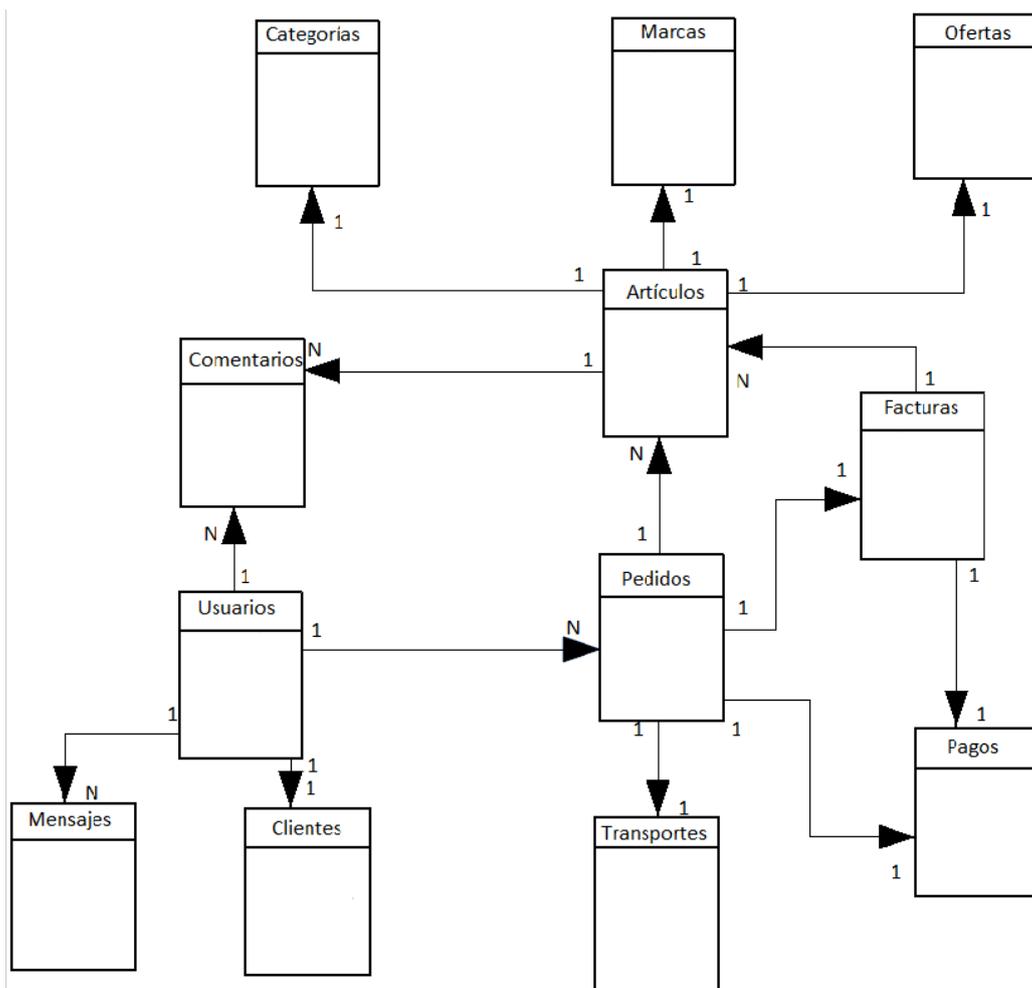


Figura 4.3: Diseño de la base de datos utilizando una base de datos relacional.

4.3. Diseño e implementación del servidor

El servidor es el encargado de recibir las peticiones del cliente y de su posterior respuesta. Para ello, se ha implementado un API REST. Esta API REST es la encargada de devolver los datos requeridos en cada petición del cliente. El API REST se explica en la sección 4.3.2. Por otro lado, el servidor es el que realiza las peticiones a base de datos para la obtención, el guardado, borrado y actualización de los datos almacenados en ella.

La implementación del servidor se ha realizado usando Node.js y Express, usado este último para la estructuración de los archivos del servidor y para el enrutamiento de las llamadas a la API REST. Node.js está orientado a eventos asíncronos que se ejecutan en un único hilo, como se explicó en la sección 3.7. Para el manejo de los eventos asíncronos es necesario recurrir a la programación asíncrona y, en este tipo de programación, es común recurrir al uso de los denominados *callbacks*, funciones que ejecutan otra función una vez terminada la operación. Para evitar estos *callbacks* y hacer el código más legible, se ha recurrido al uso de los operadores *async* y *await*. Estos operadores tienen como finalidad simplificar la manera de programación cuando se programa de manera asíncrona, ya que permiten ejecutar promesas y esperar el resultado sin utilizar *then* y *catch*, muy usados en JavaScript para este tipo de programación, y que lo único que les diferencia de los *callbacks* es la simplicidad en el código.

4.3.1. Estructura del servidor

- **Server.js**: archivo donde se establece el puerto de la conexión de la aplicación y desde el cual se inicia el servidor. Además, carga el archivo **App.js**, archivo principal del servidor.
- **App.js**: archivo principal del servidor. Es donde se establece la conexión con la base de datos y donde se configura el *middleware* para servir el API REST y los datos estáticos.
- **Package.json**: archivo donde se especifican las dependencias de las bibliotecas utilizadas en el servidor y la versión de cada una de ellas. Además, se define el nombre y la descripción de la aplicación.
- **/.node_modules**: directorio donde están instaladas las bibliotecas usadas en el servidor. Al recurrir a Express para la organización del servidor, muchas de las bibliotecas ya vienen

definidas. Pero muchas otras se han instalado por necesidades de la aplicación. Para su instalación se ha usado la instrucción *npm install*.



Figura 4.4: Estructura de los directorios del servidor.

En la figura 4.4 se puede observar como está organizado el servidor.

- **/config**: directorio que contiene la configuración del servidor. El archivo **nevadoServer.properties** contiene los contextos de los distintos *endpoints* usados en el API REST. De este modo, en el caso de cambiar de contexto, únicamente se modificaría en este archivo y no en cada uno de los *endpoints* afectados por la modificación. El archivo **passport.js** permite la autenticación de los usuarios. Además, permite que únicamente el usuario con rol de administrador pueda acceder a la parte administración de la tienda *online*.

- **/modelos:** directorio que contiene la estructura de los documentos de las colecciones usadas en la base de datos de la aplicación.
- **/routes:** directorio en el que se encuentran todos los *endpoints* de la API. Consta de dos subdirectorios, uno para la parte que solo puede ver el administrador de la tienda *online* (**/admin**) y otro para la parte de la tienda *online* que ven los usuarios **/front**. Además, contiene los archivos **APIREST.JS**, **APIRESTADMIN.JS**, **APIRESTORDERS.JS** y **APIRESTUSER.JS**. Estos archivos son los que determinan a que archivo se ha de dirigir cada contexto del API.
- **/services:** contiene la lógica del negocio de la aplicación, es decir, se especifica que se debe hacer en cada llamada a la API. En estos archivos se produce el acceso a la base de datos para el guardado, borrado, búsqueda y actualización de los datos almacenados. Está organizado de la misma forma que el directorio **/routes**, un subdirectorio para la administración y otro para la tienda *online* que ven los usuarios.
- **/utils:** directorio que contiene las funciones genéricas usadas en los diferentes servicios del servidor. Entre algunas funcionalidades, destacan el formateo de fechas, la creación de PDFs y la configuración del correo para poder enviar *emails* a los usuarios de la tienda *online*.

4.3.2. API REST de la aplicación

La API es cada uno de los *endpoints* que tiene la aplicación para que el cliente y el servidor se comuniquen. Se utiliza el protocolo HTTP para establecer estas conexiones, por lo que las operaciones soportadas son GET, POST, PUT y DELETE. La respuesta por parte del servidor a estas llamadas tienen un formato JSON.

Los *endpoints* de la aplicación implementada en este trabajo son:

- **POST /nevado/auth/register:** registra nuevos usuarios en el sistema. Antes de registrarlos, comprueba si existe el *email* en el sistema, en caso de existir manda un error indicando que el usuario está dado de alta. Al mismo tiempo que se crea un registro nuevo en la colección de usuarios, se crea un nuevo registro en la colección de clientes.

- **POST /nevado/auth/register/idUsuario:** Actualiza la información de confirmación del usuario.
- **POST /nevado/auth/login:** Permite el inicio de sesión. Para hacer este inicio de sesión, se comprueba el *email* y la contraseña. Si son erróneas devuelve un error que se muestra en pantalla.
- **GET /nevado/auth/logout:** Cierra la sesión del usuario.
- **GET /nevado/auth/forgetPassword/email:** Envía un correo al usuario, que ha olvidado la contraseña, con el enlace para restituirla. Antes de enviar el correo se comprueba que el usuario esté registrado en la aplicación. Si no lo está devuelve error y no envía el correo.
- **POST /nevado/auth/forgetPassword/idUsuario:** Permite reestablecer la contraseña del usuario.
- **GET /nevado/user/idUsuario:** Obtiene la información personal de un usuario para mostrarla en pantalla. El usuario tiene que estar autenticado para ver esta información.
- **POST /nevado/user/idUsuario:** Permite actualizar los datos personales de un usuario. Sólo el usuario y el administrador de la aplicación pueden modificar estos datos.
- **GET /nevado/orders/idUsuario:** Devuelve la lista de pedidos que el usuario ha realizado. El usuario debe haber iniciado sesión en la tienda *online* para ver esta información.

API parte de administración de la aplicación

La administración es lo que comúnmente se denomina trastienda. Permite gestionar la tienda *online*, es decir, añadir productos, proveedores, ver la lista de clientes y administrar los pedidos que hay. Únicamente puede acceder a esta parte aquellos usuarios que tienen rol de administrador. Para controlar esto, se ha usado *Passport.js*. Passport permite añadir en cada *endpoint* la autenticación del usuario. Esta autenticación será explicada en la sección 4.3.3.

- **GET /nevado/admin:** Muestra la página principal de la parte de administración.
- **GET /nevado/admin/orders:** Devuelve la lista de pedidos realizados por los clientes de la tienda *online*. Estos pedidos están ordenados por la fecha de entrada. Se puede aplicar filtros, para que únicamente devuelva los pedidos que se necesitan.

- **GET /nevado/admin/orders/export:** Exporta la lista de pedidos realizados por parte de los clientes en formato Excel.
- **POST /nevado/admin/orders/idPedido:** Actualiza el estado del pedido. Cuando un cliente finaliza un pedido siempre se crea con estado “pendiente”. El administrador modifica el estado cuando se empieza a preparar para permitir al usuario conocer el estado de su pedido.
- **POST /nevado/admin/orders/cobrado/idPedido:** Actualiza el campo cobrado de un pedido. Esta opción se ha implementado pensando en los pagos contra-reembolso, ya que estos pagos se realizan en el momento que el cliente ha recibido su producto en casa.
- **GET /nevado/admin/orders/pending:** Devuelve el número de pedidos que están con estado “pendiente”. Este número se muestra en el *header* de la parte de administración, agilizando la preparación de los envíos.
- **GET /nevado/admin/orders/products:** Recupera la información de los productos que contiene un pedido.
- **GET /nevado/admin/invoices:** Devuelve la lista de facturas del sistema. Está ordenada por número de factura y, además, permite aplicar filtros para la búsqueda de facturas concretas.
- **GET /nevado/admin/invoices/export:** Exporta la lista de facturas almacenadas en el sistema en formato Excel.
- **POST /nevado/admin/invoices/pedido:** Genera la factura correspondiente a un pedido. La factura siempre se genera al finalizar el pedido, exceptuando cuando el método de pago es contra-reembolso, que se generará a mano.
- **POST /nevado/admin/invoices/products:** Devuelve la información relativa a los productos que contiene la factura de un pedido.
- **POST /nevado/admin/invoices/client:** Devuelve la información que se necesita de un cliente para generar la factura.
- **POST /nevado/admin/invoices/pdf:** Genera el PDF de la factura.

- **GET /nevado/admin/invoices/idFactura:** Comprueba si la factura está generada. Sólo se puede generar una factura por pedido.
- **GET /nevado/admin/products:** Retorna la lista de productos que tiene la tienda *online*. Permite filtrar los productos por código, marca, categoría, modelo y precio.
- **GET /nevado/admin/products/export:** Exporta la lista de productos de la tienda *online* en formato Excel.
- **GET /nevado/admin/products/idProducto:** Retorna la información de un producto.
- **POST /nevado/admin/products:** Añade un nuevo producto en el sistema.
- **POST /nevado/admin/products/idProducto:** Actualiza la información de un producto.
- **DELETE /nevado/admin/products/idProducto:** Elimina un producto del catálogo de la tienda *online*.
- **GET /nevado/admin/brands:** Devuelve la lista de marcas registradas en la tienda *online*.
- **GET /nevado/admin/brands/export:** Exporta la lista de marcas registradas en la tienda *online* en formato Excel.
- **DELETE /nevado/admin/brands/idMarca:** Elimina una marca de la tienda *online*.
- **POST /nevado/admin/brands:** Añade una nueva marca al registro de la tienda *online*.
- **GET /nevado/admin/categories:** Devuelve la lista de categorías registradas en la tienda *online*.
- **GET /nevado/admin/categories/export:** Exporta la lista de categorías registradas en la tienda *online* en formato Excel.
- **DELETE /nevado/admin/categories/idCategoría:** Elimina una categoría de la tienda *online*.
- **GET /nevado/admin/categories/products:** Devuelve los productos que tienen una misma categoría.

- **POST /nevado/admin/categories:** Añade una nueva categoría al registro de la tienda *online*.
- **GET /nevado/admin/clients:** Devuelve el listado de usuarios registrados en la tienda *online*. Además, permite filtrar el listado por nombre, apellidos, dirección de correo electrónico, DNI y teléfono.
- **GET /nevado/admin/clients/export:** Exporta el listado de usuarios registrados en la tienda *online* en formato Excel.
- **DELETE /nevado/admin/clients/idUsuario:** Elimina un usuario de la aplicación. El usuario registrado puede cancelar su cuenta cuando lo desee a través de un correo electrónico al administrador. El administrador, por la Ley de protección de datos y política de privacidad, deberá borrarlo del registro del cliente.
- **GET /nevado/admin/payments:** Devuelve la lista de métodos de pagos disponibles en la tienda *online*. Se puede filtrar por el nombre del método de pago.
- **POST /nevado/admin/payments:** Añade un nuevo método de pago a la tienda *online*.
- **GET /nevado/admin/payments/export:** Exporta la lista de métodos de pagos disponibles en la tienda *online* en formato Excel.
- **DELETE /nevado/admin/payments/idPago:** Elimina un método de pago del sistema.
- **GET /nevado/admin/transports:** Devuelve la lista de empresas de transporte que se utilizan para realizar los envíos. Se puede filtrar la lista por el nombre de la empresa.
- **GET /nevado/admin/transports/export:** Exporta la lista de empresas de transporte que se utilizan para realizar los envíos en formato Excel.
- **POST /nevado/admin/transports:** Añade una nueva empresa de transporte.
- **DELETE /nevado/admin/transports/idEmpresaTransporte:** Elimina una empresa de transporte del sistema.
- **GET /nevado/admin/offers:** Devuelve la lista de ofertas que están disponibles en la tienda *online*.

- **POST /nevado/admin/offers:** Añade una nueva oferta.
- **GET /nevado/admin/offers/export:** Exporta la lista de ofertas disponibles en formato Excel.
- **POST /nevado/admin/offers/idOferta:** Permite modificar las fechas en la que la oferta está disponible, así como la descripción de la misma.
- **GET /nevado/admin/offers/idOferta:** Elimina una oferta del sistema.
- **GET /nevado/admin/offers/Products:** Devuelve los productos que tienen ofertas.
- **GET /nevado/admin/messages:** Retorna la lista de mensajes enviados por los usuarios.
- **GET /nevado/admin/messages/export:** Exporta la lista de mensajes enviados por los usuarios en formato Excel.
- **DELETE /nevado/admin/messages/idMensaje:** Elimina un mensaje.
- **POST /nevado/admin/messages/idMensaje:** Permite poner el mensaje como leído o no leído.
- **POST /nevado/admin/messages/send:** Envía la respuesta a un mensaje a través de un correo electrónico.
- **GET /nevado/admin/import/template:** Descarga la plantilla Excel que hay que usar para poder importar datos de manera masiva en la tienda *online*.
- **POST /nevado/admin/import/upload:** Importa los datos del archivo Excel seleccionado.
- **GET /nevado/admin/import/getImportData:** Muestra el *log* de la importación.

API parte pública de la aplicación

La parte pública de la aplicación la forman las vistas que no necesitan un rol específico para interactuar en ellas. Los *endpoints* de esta parte son los siguientes:

- **GET /home:** Muestra la página principal de la aplicación.
- **GET /home/topVentas:** Devuelve los productos más vendidos para mostrarlos en el apartado Top Ventas de la página principal.

- **GET /home/ofertas:** Devuelve los productos que tienen ofertas para mostrarlos en el apartado ofertas de la vista principal.
- **GET /home/largeAppliances:** Muestra las subcategorías que tiene la categoría Gran electrodoméstico.
- **GET /home/smallKitchenAppliances:** Muestra las subcategorías que tiene la categoría Pequeño electrodoméstico de cocina.
- **GET /home/image:** Muestra las subcategorías que tiene la categoría Imagen.
- **GET /home/sound:** Muestra las subcategorías que tiene la categoría Sonido.
- **GET /home/telephony:** Muestra las subcategorías que tiene la categoría Telefonía y electrónica.
- **GET /home/personalCare:** Muestra las subcategorías que tiene la categoría Cuidado personal.
- **GET /home/computing:** Muestra las subcategorías que tiene la categoría Informática.
- **GET /home/product/idProducto:** Devuelve la información relativa a un producto para mostrar esta información en la vista de detalle del producto.
- **GET /home/brand:** Devuelve el logo de la marca del producto pasado como parámetro para mostrarlo en la vista detalle del producto.
- **GET /home/similarProducts:** Permite ver productos similares en la página de detalle de productos. Ofreciendo diferentes alternativas al producto seleccionado.
- **GET /home/category/nombreCategoria:** Devuelve todos los productos de una subcategoría. Estos productos se mostrarán en la vista de la subcategoría seleccionada.
- **GET /home/brands:** Devuelve los productos de una subcategoría filtrados por marca.
- **GET /home/paymentMethods:** Retorna los métodos de pago con los que el cliente puede pagar su pedido.
- **GET /home/cart:** Permite ver los productos que hay en el carrito de compra.

- **GET /home/termUse:** Muestra la página de condiciones de uso.
- **GET /home/faq:** Muestra la página de preguntas frecuentes.
- **GET /home/contact:** Muestra la página de ¿quiénes somos?.
- **GET /home/privacyPolicy:** Muestra la página de política de privacidad.
- **POST /home/review:** Añade una reseña en un producto.
- **POST /home/saveOrder:** Permite finalizar el pedido.
- **POST /home/message:** Envía un mensaje al administrador de la aplicación.

4.3.3. Registro y autenticación de usuarios

La aplicación consta de dos perfiles de usuario, un perfil cliente y un perfil administrador. El perfil cliente se asigna a un usuario cuando se registra en la aplicación. El perfil administrador únicamente lo tiene el dueño o administrador de la tienda *online*. Además, el perfil administrador permite acceder a la trastienda.

Proceso de registro de un usuario

Cuando un usuario decide registrarse en la tienda *online* debe acceder a */nevado/login*, seleccionar la pestaña Registrarse y rellenar los campos obligatorios para el registro. Cuando el usuario pulsa sobre el botón registrarse se realiza una petición al servidor, en la que viajan los datos introducidos por el usuario. El servidor comprueba si el *email* introducido ya se encuentra en el sistema. Para ello, realiza una búsqueda en la base de datos. Si el *email* ya está registrado se devuelve un error, indicando que el usuario ya está registrado. Si, por el contrario, el *email* no se encuentra en nuestro sistema entonces guarda los datos del usuario en la base de datos, creando un nuevo registro en la colección de usuarios. Antes de crear el registro, se deben proteger los datos más susceptibles del usuario. La contraseña es la que permite realizar compras y acceder a la información personal del usuario, por lo que no se puede guardar en la base de datos en texto plano. Según la Ley de protección de datos, en su artículo noveno, el encargado del tratamiento de los datos susceptibles debe garantizar la seguridad de los mismos adoptando las medidas técnicas y organizativas necesarias. Para cumplir con este artículo, se cifra la contraseña utilizando la biblioteca Bcryptjs. Esta biblioteca permite realizar un cifrado

robusto frente ataques de fuerza bruta, como se ha mencionado en la sección 3.13. Para llevar a cabo el cifrado de la contraseña se realizan los siguientes pasos:

1. Se genera el *salt*, valor aleatorio que es usado para dar mayor robustez a el cifrado. Para generar este valor se utiliza la función *genSalt*, función propia de la biblioteca de Bcrypt. El *salt* generado tiene una longitud de 10 caracteres, valor por defecto de Bcrypt. Se ha optado por este valor porque es lo suficientemente grande para evitar que ataques de fuerza bruta consigan descifrar la contraseña. Además, con este valor el proceso de cifrado no se demora mucho tiempo.
2. Se cifra la contraseña utilizando la función *hash*, también de Bcrypt. Para llevar a cabo el cifrado, esta función utiliza el *salt* generado en el paso anterior.

Una vez cifrada la contraseña, se crea el nuevo registro en la colección usuarios. Además, se crea una nueva entrada en la colección clientes, que almacena la información personal del cliente. Para dar por finalizado el proceso de registro, se envía un correo al usuario. El correo contiene un enlace de confirmación de la cuenta. De este modo, se garantiza que el correo introducido por el usuario es válido y está dado de alta en un sistema de mensajería. Si el usuario no confirma el correo no podrá realizar compras en la tienda *online*.

Proceso de autenticación de un usuario

Un usuario debe autenticarse para poder realizar determinadas operaciones dentro de la tienda *online*, por ejemplo, realizar sus compras, acceder a sus datos personales o acceder a la información de sus pedidos. Para realizar esta autenticación se ha utilizado la biblioteca *Passport.js*. Esta biblioteca permite realizar este proceso de manera sencilla y rápida. Passport almacena los datos necesarios para la autenticación en una cookie de sesión. Para configurar dicha cookie de sesión se ha usado *express – session*, que es un módulo de Express. La configuración se realiza cuando se arranca el servidor de Node.js, por lo que se encuentra situada en el fichero *app.js*. Los parámetros que se han configurado son el tiempo de expiración de la cookie, dónde se guardan las sesiones y el valor de la clave *secret*. Este *secret* es una cadena de texto que se utiliza como identificador y que permite validar la sesión. Passport tiene muchas estrategias para autenticar a un usuario, en la aplicación se ha optado por utilizar *Passportlocal*, la cual utiliza el *email* y la contraseña para realizar la autenticación.

Para acceder a una cuenta de usuario, se debe navegar a */nevado/login*, acceder a la pestaña entrar, rellenar el *email* y la contraseña y pulsar sobre el botón entrar. Al pulsar sobre el botón, se realiza una petición al servidor, en la que viajan la contraseña y el *email* introducido por el usuario. El servidor ejecuta la estrategia de Passport que se ha implementado. Esta estrategia consiste en buscar el *email* en el sistema. Si no lo encuentra envía un error, indicando que el usuario no está registrado. Si lo encuentra, procede a comparar la contraseña enviada con la almacenada. Para realizar esta comparación se ha utilizado la función de *bcrypt compare*, que cifra la contraseña introducida y la compara con la almacenada. Si las contraseñas no coinciden se envía un error. Si coinciden, Passport crea una sesión de usuario que serializa y guarda en la base de datos. Además, estos datos de la sesión se añaden a la cookie, la cual se envía en cada una de las peticiones realizadas al servidor. De esta forma, no hay que enviar el *email* y la contraseña en cada llamada. El servidor únicamente se encargará de validar la cookie enviada.

Para acceder a determinadas vistas de la aplicación como, por ejemplo, las vistas de administración, el usuario, además de estar logueado, deberá tener el perfil de administrador. Para hacer esta comprobación, en cada llamada realizada bajo el contexto */admin* se ejecutará una función que valida que el usuario tenga este perfil. Si el perfil no coincide, no se permitirá el acceso a esta parte y se redireccionará a la página principal.

4.3.4. Sistema de subida de archivos en el servidor

La aplicación permite realizar subidas de archivos al servidor. Los archivos que se permiten subir son imágenes y archivos *.zip*. La subida de imágenes se produce al registrar un nuevo producto o al actualizarse. Además, se suben imágenes al añadir a la tienda *online* nuevos proveedores, nuevas categorías y nuevas formas de pago. Las imágenes tienen que tener un formato determinado para poder subirse (*.jpeg*, *.jpg* o *.png*). Si el formato es diferente se produce un error, que se muestra por pantalla. Los archivos *.zip* se suben al servidor cuando se realiza la importación de productos, categorías o marcas. El sistema de importación se explica en la sección 4.3.6.

Para realizar esta subida de archivos en el servidor, se ha utilizado la biblioteca *Multer* de Node.js. *Multer* [5] facilita el manejo de datos codificados como *multipart/form-data*. Para ello, se incluye la biblioteca y se configura pasándole como parámetro el *path* donde se almacenarán los archivos subidos. En el *route* (endpoint correspondiente) se ejecuta la función *single*,

pasándole como parámetro el nombre del campo del formulario donde se envió el fichero.

```
var multer = require('multer');  
  
var upload = multer( dest: properties.get('nevado.server.import.folder') );  
routesImport.post('/upload', passportConfig.isAuthenticated, upload.single('file'), ... );
```

4.3.5. Generación de PDFs

Cuando se realiza un pedido es necesario enviar la factura, ya que es esta la que actúa como ticket de compra. Por ello, se ha implementado una funcionalidad que permite generar el PDF de una factura. Para la generación de los PDFs se ha utilizado la biblioteca *HTML – PDF* [6] de Node.js. A través de un HTML, generado con la información recogida en la base de datos, se crea el PDF. Para ello, se utiliza la función *create* de la biblioteca, a la cual se le pasa el HTML generado y se llama a la función *toFile*, a la que se le indica donde se crea el archivo PDF y qué nombre va a tener dicho archivo.

Producto	Cantidad	Precio
1253052 WTA 8612 XSW A+++ Orbegozo	2	255.2
124578 WTA 8612 XSW A+++ Cecotec	1	260
TOTAL:		770.4

Figura 4.5: Ejemplo PDF factura.

4.3.6. Sistema de importación y exportación

Una tienda *online* requiere tener en su catálogo numerosos productos. Añadir cada uno de ellos al sistema puede llegar a ser pesado y tedioso. Por eso, se ha implementado un sistema de importación, que permite cargar masivamente datos en la base de datos. En este sistema se pueden cargar productos, marcas y categorías.

Para realizar la importación el administrador de la tienda debe ir a `/admin/import`, seleccionar el tipo de datos que se van a importar, es decir, debe indicar si son productos, categorías o marcas y pulsar el botón importar. Cuando se pulsa el botón importar se realiza una petición al servidor, en la que viaja el archivo a importar. El tipo de archivo que se permite importar es `.zip`. El `zip` contiene un Excel con los registros que se quieren añadir a la base de datos y las imágenes asociadas a cada registro. Cuando se recibe la petición, se guarda el archivo `.zip` en el servidor, como se explica en la sección 4.3.4 y, posteriormente, se realiza la importación. Para ello, se descomprime el archivo `.zip`, utilizando la biblioteca *Adm – zip* [3]. Se extrae el documento Excel y se parsea dicho documento para convertirlo en un array de objetos JSON. Para realizar el parseo del documento Excel se usa la biblioteca *SheetJS – XLSX* [7]. El array va a tener tantos objetos JSON como registros tenga el documento Excel. A cada objeto JSON se le añade la ubicación de la imagen asociada a cada registro. Antes de guardarlo en la base de datos, se realizan una serie de comprobaciones como, por ejemplo, que todos los campos obligatorios estén rellenos o, en el caso de los productos, que el precio sea numérico o la marca y categoría exista en la base de datos. Una vez terminada esta comprobación, se insertan en la base de datos todos los registros, utilizando la instrucción *insertMany* de Mongoose, realizando así una única *query*. Cuando el proceso termina, se muestra en pantalla, a través de un sistema de *log*, el número de registros importados y cuales de ellos han fallado, indicando el número de línea y el error producido.

Para facilitar la creación de los documentos Excel y evitar errores a la hora de importarlos, se ha implementado una opción de descarga de plantilla. Para descargar esta plantilla, basta con seleccionar el tipo de dato que se quiere importar y pulsar sobre el botón plantilla, descargando un documento Excel con las cabeceras de los campos que se han de rellenar. Este sistema de descarga de plantilla es similar al sistema de descarga de los documentos Excel que se encuentran en las distintas vistas de la parte de administración.

Para la creación de los documentos Excel, se utiliza la biblioteca de Node.js *excel4node* [4].

El documento Excel se va rellenando con los metadatos de la cabecera y los datos devueltos por la base de datos. En el caso de la plantilla, sólo se rellena la cabecera del documento Excel. Este documento se envía en la respuesta al cliente, y éste lo descarga utilizando *saveAs* de la biblioteca de Angular *File – saver* [2].

4.3.7. Envío de correos

Para el envío de correos se ha utilizado la biblioteca *Nodemailer* [20] de Node.js. Lo primero que se hace es crear un objeto con la función *createTransport*. A esta función se le pasan las credenciales de la cuenta desde la que se va a realizar el envío. Posteriormente, se crea un nuevo objeto con los datos del usuario que recibe el correo, el asunto y el mensaje del correo. Y, por último, se envía el correo utilizando la función *sendMail* de la propia biblioteca. El envío de correos se ha usado en el registro del usuario, en el restablecimiento de la contraseña y en la respuesta de las preguntas de los usuarios al administrador.

4.4. Diseño e implementación del cliente

El cliente se ha desarrollado utilizando Angular y está programado en TypeScript. Está formado por diferentes vistas, que forman la interfaz de usuario. Estas vistas se cargan de manera dinámica e instantánea, ya que cuando se modifica la URL no se recarga la página. Por lo tanto, el cliente se ha implementado como Single-Page. Además, con el uso de Angular, el cliente tiene una arquitectura modelo-vista-controlador.

El cliente es el encargado de recoger la información introducida por el usuario cuando interactúa en la aplicación. Para atender los eventos lanzados por el usuario, se ha alojado en el servidor proporcionado por Angular CLI.

4.4.1. Estructura del cliente

- **/node_modules:** Directorio donde se encuentran instaladas las dependencias del cliente. Estas bibliotecas se descargan y se instalan mediante la instrucción *npm install*, ya que Angular CLI usa Node.js para ejecutar su servidor.

- **package.json**: Archivo donde se especifican las dependencias utilizadas en el proyecto. Estas dependencias se declaran junto con la versión utilizada.
- **proxy.conf.json**: Archivo de configuración del *proxy* utilizado como intermediario entre el cliente y el servidor.
- **tsconfig.json**: Archivo de configuración del compilador de los archivos TypeScript.

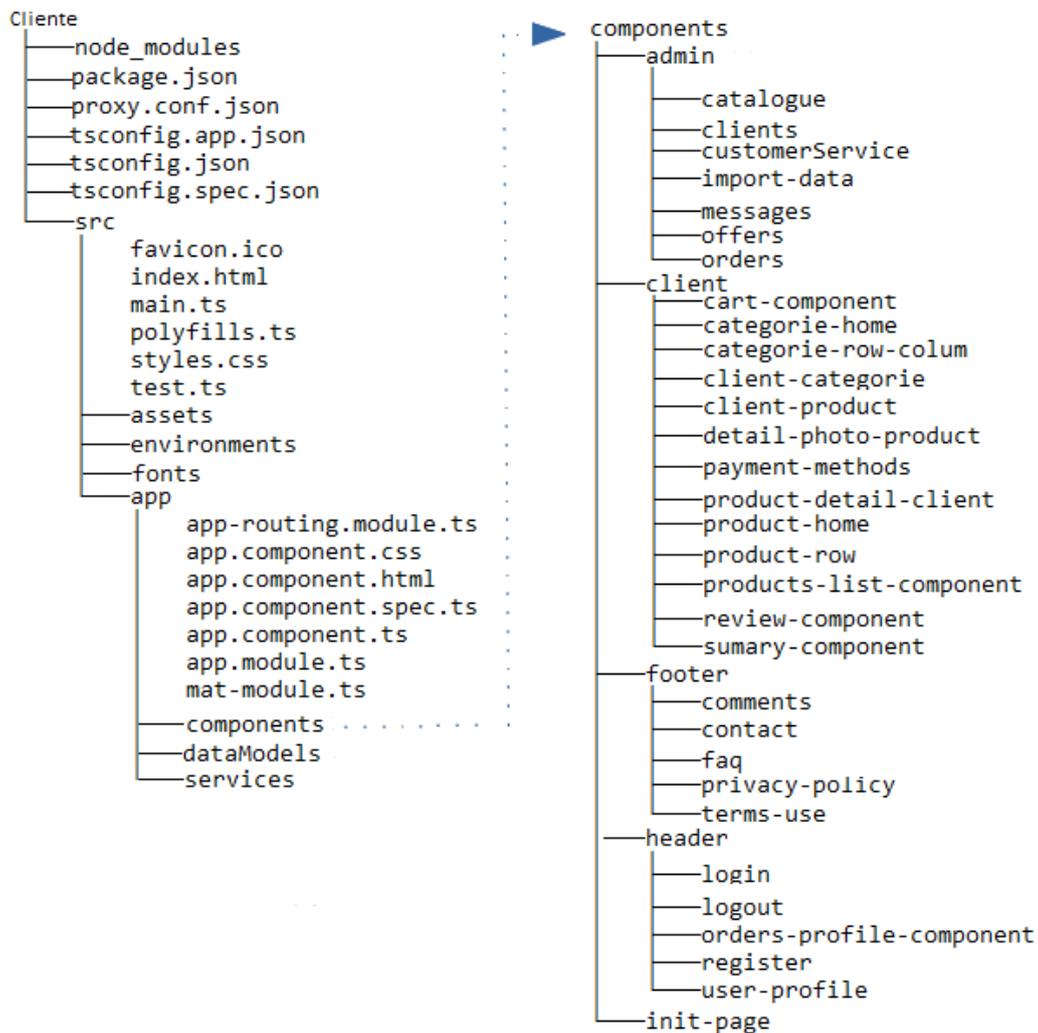


Figura 4.6: Estructura de los directorios del cliente.

En la figura 4.6 se puede observar como está organizado el cliente.

- **/src**: Directorio donde se encuentra la lógica del cliente, en él se sitúan los componentes, las directivas y los servicios utilizados para interactuar en la interfaz de usuario. Está

formado por:

- **index.html**: Archivo HTML principal de la interfaz de usuario. Es el punto de entrada a la aplicación. En él se carga el componente principal, los estilos generales y los *scripts* necesarios para el correcto funcionamiento del cliente.
- **main.ts**: Archivo TypeScript principal. En él se establece el módulo principal del cliente, siendo este **AppModule**.
- **styles.css**: Archivo donde se definen los estilos generales de la interfaz de usuario.
- **/fonts**: Directorio donde se almacenan las fuentes utilizadas en la interfaz de usuario.
- **/environments**: Directorio donde se establece la configuración de los distintos entornos (desarrollo, test, producción). En el caso de la aplicación, no se ha configurado ningún entorno, ya que siempre se ha trabajado en local.
- **/assets**: Directorio donde se almacenan los recursos estáticos como el logo o las imágenes del *carousel* implementado en la página *home* de la aplicación.
- **/app**: Directorio que contiene los componentes, servicios y directivas utilizados para implementar el cliente. Está compuesto por:
 - **app.module.ts**: Módulo principal del cliente. En este archivo se especifican todos los componentes que forman la interfaz de usuario.
 - **app.component.html**: Archivo HTML del componente principal. Está formado por dos componentes fijos para todas las vistas, *Header* y *Footer*, y una parte variable donde se cargan los distintos componentes. Estos componentes varían según se modifique la URL.
 - **app.routing-module.ts**: Archivo donde se asocian los componentes, que se cargan en la parte variable del componente principal, con cada una de las rutas de la aplicación.
 - **mat.module.ts**: Archivo donde se declaran los módulos de Angular Material que se pueden utilizar.
 - **/components**: Directorio que contiene los componentes que forman el cliente. Se divide en cinco subdirectorios. **/admin** (componentes de la trastienda), **/client** (componentes de la parte pública de la tienda), **/footer** (componentes del

pie de página), **/header** (componentes de la cabecera), **/init-page** (componentes que forman la vista *home* de la tienda). Cada uno de los componentes tiene asociado un archivo HTML, un archivo CSS y un archivo .ts, que es el controlador del componente.

- **/dataModels**: Directorio que contiene los modelos de datos utilizados en el cliente. En este directorio se encuentra el archivo **properties.ts**, que contiene los contextos de las peticiones realizadas al servidor. De este modo, si uno de los contextos se modifica únicamente habría que modificar este fichero y no todos los archivos donde se encuentra el contexto a modificar.
- **/services**: Directorio que contiene los servicios utilizados en los controladores de los componentes. Estos servicios se pueden usar desde cualquier componente. Sirven para compartir datos y funcionalidades entre componentes, y para definir las peticiones a la API. Se divide en tres subdirectorios: **/admin** (servicios usados en la trastienda), **/front** (servicios usados en la parte pública de la tienda), **/faq** (servicio usado en el componente preguntas frecuentes). Además, contiene los archivos **auth-service.service** (contiene las peticiones a la API del registro), **user-service.service** (servicio donde se definen las variables de usuario que se guardan en el localStorage).

4.4.2. Vistas

Las vistas de la aplicación forman la interfaz de usuario. Cada vista está formada por uno o más componentes. Cada uno de estos componentes tiene asociado un controlador, una hoja CSS y un documento HTML. Trabajar con Angular hace que el código esté ordenado, por lo que las aplicaciones son más escalables. Para aumentar la escalabilidad de la aplicación se han creado componentes reutilizables, es decir, componentes que se usan en varias vistas de la aplicación. Crear estos componentes reutilizables significa aumentar la mantenibilidad de las aplicaciones, ya que a la hora de producirse un error o una actualización en uno de los componentes reutilizables supone modificar únicamente el componente reutilizable, evitando modificar cada uno los componentes afectados por el error o la actualización.

Los componentes que se reutilizan en la aplicación son:

- **Client-product:** Tarjeta resumen del producto. Muestra la información más importante del producto, marca, modelo, precio y envío gratuito, si lo tuviese. Permite añadir el producto al carrito de compra pulsando el botón “comprar”, sin necesidad de pasar por la vista de *Detalle del producto*. Se utiliza n veces en el *Home* de la aplicación, ya que se usa para mostrar los productos más vendidos y los productos que tienen oferta. También, se utiliza tantas veces como productos tenga la subcategoría en la vista *Listado de productos*. Además, se utiliza en la vista *Detalle del producto* para mostrar los productos de la sección *Artículos relacionados*.



Figura 4.7: Ejemplo de una tarjeta resumen del producto (Componente client-product).

- **Product-row:** Fila de tarjetas resumen del producto. Permite mostrar varias tarjetas resumen en una misma fila. Se utiliza en el *Home* de la aplicación, tanto en top ventas como en ofertas, y en la vista *Detalle del producto* en el apartado *Artículos relacionados*.



Figura 4.8: Ejemplo fila de tarjetas resumen del producto (Componente product-row).

- **Product-detail-client:** Detalle del producto. Permite visualizar la información del producto de manera detallada. Se utiliza cada vez que se pulsa sobre la imagen o el modelo

del producto en el componente *client-product*.



Figura 4.9: Ejemplo detalle del producto (Componente Product-detail-client).

- **Client-categories:** Tarjeta subcategoría. Permite visualizar el nombre y la imagen con la que se caracteriza una subcategoría. Se utiliza tantas veces como subcategorías tenga la categoría general.



Figura 4.10: Ejemplo tarjeta subcategoría (Componente Product-detail-client).

Vistas parte pública de la tienda

Todas las vistas de la parte pública de la tienda tienen dos componentes comunes: *Header* (cabecera de la página) y *Footer* (pie de página).

- **Header:** Es una barra de navegación que se adapta en función de si se ha iniciado sesión. Si el usuario no ha iniciado sesión, se muestra un enlace para ver el carrito y otro para ha-

cer *login*. Si el usuario ha iniciado sesión, se muestran enlaces para cerrar sesión, acceder a la información personal, ver los pedidos realizados y visualizar el carrito.

- **Footer:** Pie de página que contiene los enlaces para ver las condiciones de uso, la política de privacidad, las preguntas frecuentes y la página de contacto. A diferencia del *Header* los enlaces siempre son los mismos.

Vistas que no necesitan autenticarse en la aplicación para poder visualizarse:

- **Login:** Página donde el usuario puede registrarse o acceder a su cuenta. Está formado por dos pestañas, uno para el registro y otro para el *login*. Cada pestaña es un formulario donde el usuario debe rellenar los datos de acceso (en el caso del *login*) y los datos para darse de alta (en el caso del registro). En ambos formularios se comprueba que los campos estén rellenos y que los datos sean válidos. Cuando se crea una cuenta, la contraseña y la contraseña de confirmación deben coincidir. Además, se informa al usuario de la fortaleza de la contraseña. Si se produce un error en el registro o en el acceso a la cuenta, se muestra un mensaje en pantalla a través del sistema de avisos, explicado en la sección 4.4.6.
- **Recuperar contraseña:** Vista que permite restablecer la contraseña, enviando un enlace al correo introducido en el formulario. Previamente, se comprueba que el correo es válido para un usuario de la aplicación. El enlace redirecciona a una nueva vista con dos campos a rellenar, la nueva contraseña y su confirmación. Cuando se pulsa el botón “Cambiar contraseña” se actualiza la contraseña.
- **Home:** Vista principal de la parte pública de la aplicación. Está formada por un menú que permite la elección de categorías generales, un *carousel*, que sirve de sistema de anuncios, y dos listados de productos, el *Top ventas* y *Ofertas*. Ambos listados están desarrollados con los componentes reutilizables *Product-row* y *client-product*. Al pulsar sobre la foto o modelo del producto se navega a la vista *Detalle del producto*. Si se pulsa el botón comprar se añade un producto a la cesta. Las categorías generales son enlaces que permiten navegar a la vista de la categoría general seleccionada. Este elemento de la vista se convierte en un *dropdown* cuando el tamaño de la vista es pequeño.
- **Categoría:** Vista en la que se muestran las subcategorías de una categoría general. Las

subcategorías se visualizan a través del componente reutilizable *client-categories*. Si se pulsa sobre la foto se navega al listado de productos de la subcategoría.

- **Listado de productos de una subcategoría:** Vista en la que se muestra el listado de los productos de una subcategoría. Este listado está paginado permitiendo al usuario elegir la cantidad de productos a mostrar, siendo 12 el número máximo. Se diseñó así para evitar cargar todos los productos de la base de datos, haciendo que la *query* sea rápida y eficiente. Para la creación del listado se ha utilizado el componente *grid* de Angular Material, que a su vez muestra los productos utilizando el componente *client-product*. Los productos se pueden ordenar por: los más vendidos, los mejores valorados o de más caros a más baratos y viceversa. Además, se puede filtrar por marca (únicamente se muestran las marcas de los productos de la subcategoría seleccionada), por precio (en un rango de 0 a 1000), por disponibilidad, por oferta y por características. Para implementar el menú de filtros se ha utilizado *mat-accordion* de Angular Material. Para realizar el filtrado y la ordenación de los productos se ha recurrido al uso de la función *aggregate* de Mongoose, que permite transformar los datos por etapas hasta conseguir el resultado esperado.
- **Detalle del producto:** Vista que permite conocer la información detallada de un producto: Foto, marca, modelo, estrellas, envío (si es gratuito o no), promociones, disponibilidad, características y reseñas.

La información de la disponibilidad cambia de color en función de la cantidad de productos en stock. Si no hay stock se muestra en rojo, si el stock es inferior a 5 se muestra en amarillo y si es mayor se muestra en verde.

Las características se visualizan con el formato que se definió (con *html-editor*) cuando el producto se almacenó en el sistema.

Las reseñas pueden ser filtradas por el número de estrellas de cada valoración. Para escribir una nueva valoración de un producto, se utiliza la vista *Añadir reseña* y es necesario haber iniciado sesión.

Cuando se pulsa el botón comprar se añade el producto al carrito, guardando los datos necesarios en el *LocalStorage* en formato JSON.

- **Carrito:** Muestra los productos que hay en el carrito, el total del carrito y la forma de en-

trega. Cuando la cantidad elegida de un producto es mayor que uno, la forma de entrega es la misma para todos. Si el carrito contiene diferentes productos, se permiten diferentes formas de entrega (envío a domicilio o recogida en tienda). Además, se permite la eliminación de los productos de forma individual o colectiva, al pulsar “Vaciar cesta”. Para la implementación del carrito se ha utilizado *LocalStorage*, donde se almacenan los productos seleccionados por el usuario en formato JSON.

- **Preguntas frecuentes:** Vista que permite conocer la respuesta de las preguntas más frecuentes formuladas por los usuarios. Cuando el usuario ha iniciado sesión puede enviar preguntas al administrador a través del formulario que aparece en esta vista. El administrador puede ver los mensajes desde la trastienda y contestarlos enviando un correo.

Vistas donde el usuario necesita estar logueado para visualizarlas:

- **Añadir reseña:** Vista que permite publicar la valoración de un producto. Es un formulario con los campos: valoración general, título, comentario, recomendación a otros usuarios y valoración del producto en cuanto a calidad precio. La valoración general, en forma de número de estrellas, se suma al computo total de estrellas de producto que, finalmente, se muestra en la vista *Detalle del producto* junto con la reseña.
- **Perfil de usuario:** Vista donde se visualiza la información personal del usuario, es decir, nombre, apellidos, *email*, teléfono, fecha de nacimiento, DNI, dirección de facturación, código postal, municipio, provincia de la dirección de facturación y la dirección, el código postal, el municipio y la provincia de envío. Todos los campos son obligatorios para realizar compras, excepto los de la dirección de envío que serán obligatorios si la dirección de facturación es distinta. Si algún campo no se rellena se mostrará un mensaje de error en la pantalla a través del sistema de aviso, explicado en la sección 4.4.6. Esta vista se muestra cuando el usuario accede a ella a través del enlace de la barra de navegación y cuando se tramita el pedido, ya que el usuario debe comprobar los datos antes de finalizar con el pedido.
- **Pedidos del usuario:** Vista que permite conocer al usuario los pedidos realizados y el estado de cada uno de ellos. Es una tabla en la que se visualiza el identificador del pedido, la fecha del pedido, la fecha aproximada de entrega y el estado. La tabla está paginada y

el usuario puede decidir la cantidad de pedidos que desea mostrar por página. Además, se le permite filtrar por identificador de pedido, estado o fecha de pedido.

- **Métodos de Pago:** Vista donde el usuario decide el método de pago del pedido. Se puede elegir entre tarjeta de crédito, Paypal y contra-reembolso. Cuando se selecciona tarjeta de crédito aparece un formulario con el tipo de tarjeta, el nombre del titular, la fecha de expiración, el número de tarjeta de pago y el CCV. Además, se comprueba que el número de tarjeta sea válido para el tipo de tarjeta seleccionado, que el CCV sea de tres posiciones y que la fecha de expiración tenga un formato correcto. Cuando se selecciona el método de pago Paypal aparece el botón “Paypal pagar”, que permite realizar el pago en el sitio web de Paypal. Cuando se selecciona contra-reembolso únicamente aparece el botón “Finalizar pedido”. Al finalizar el pedido se genera la factura, excepto en la opción contra-reembolso que se genera manualmente por el administrador una vez que se cobre el pedido. El pago con tarjeta de crédito finaliza tras la validación de la tarjeta. No se ha implementado la conexión con el banco porque requería datos y cuentas reales.

Vistas de la trastienda

La vista principal de la trastienda está compuesta por:

- **Header:** Es el componente *header* descrito anteriormente. Cuando el perfil del usuario es administrador, se muestra un indicador con el número de pedidos en estado pendiente.
- **Menú:** El menú se muestra en el lateral izquierdo y permite seleccionar las diferentes secciones de administración. Cuando se reduce el tamaño de pantalla, el menú cambia de posición y se muestra debajo del *header*.
- **Espacio de trabajo:** Es un contenedor que rellena todo el espacio disponible y donde se muestran las distintas secciones de administración.

Las secciones de la trastienda son: Clientes, pedidos (pedidos y facturas), catálogo (productos, marcas y categorías), servicios (métodos de pago y transporte), ofertas, importación y mensajería.

Todas las secciones se organizan de la misma forma. Tienen una tabla, creada con *mat-table* de Angular Material, que muestra la información de cada sección de forma paginada.

Cada tabla puede ordenar y filtrar la información. Para ello, se recogen los valores de filtrado mediante un formulario y se envían al servidor, que se encarga de introducirlos en la *query* para devolver exclusivamente los datos que coinciden. Además, las tablas son exportables a Excel (sección 4.3.6). También, hay tablas que permiten realizar acciones exclusivas. Por ejemplo, la tabla de facturas permite la exportación de cada factura en formato PDF (sección 4.3.5). La tabla de mensajería, además de permitir ver los mensajes enviados por los usuarios, permite contestar enviando una respuesta por correo, utilizando la biblioteca *Nodemailer* (sección 4.3.7).

Cada uno de los registros de las tablas tiene opciones para mostrar datos, editar la información o borrar registros. Tanto en la visualización como en la edición o creación de registros, se crea una ventana de diálogo con *mat-dialog* de Angular Material. Estas ventanas muestran y permiten la edición de los datos mediante formularios, creados con Angular Material. Todos los campos tienen control de errores. Además, son validados en el servidor y en caso de error se muestra un mensaje con el sistema de avisos (sección 4.4.6). En algunas de estas ventanas, por ejemplo, productos o marcas, se permite la subida de imágenes (sección 4.3.4) para después mostrarlas en la parte pública de la tienda. En la ventana de productos se ha utilizado la biblioteca *HTML-editor*, que permite escribir texto formateado (listas, colores, tamaños y tipos de letras, entre otras). Este texto formateado corresponde con las características de cada producto, que se mostrará en la parte pública de la tienda.

La sección de importación es completamente diferente a las descritas anteriormente. Esta sección permite la descarga de plantillas XLSX para ser importadas después, lo que permite cargar datos en la aplicación de manera masiva (sección 4.3.6). El resultado final de la importación se muestra en la misma pantalla, permitiendo conocer registros erróneos o duplicados.

4.4.3. Uso de Angular Material

Como ya se vio en la sección 3.6.2, Angular Material es una biblioteca de Angular que permite crear componentes de manera rápida y sencilla. Sus componentes se adaptan a los distintos tamaños de pantalla, por lo que se suelen utilizar para crear diseños *responsive*. En la aplicación se ha utilizado en:

- Las tablas de productos, clientes, pedidos, facturas, marcas, categorías, métodos de pago, empresas de transporte y mensajes. Para ello, se ha utilizado *mat-table*.

- Los diálogos de confirmación de borrado de productos, clientes, marcas, categorías, métodos de pago, empresas de transporte y mensajes. Usando para ello *mat-dialog*.
- Los campos de los formularios utilizados para crear productos, marcas, categorías, métodos de pago y empresas de transporte. Y en los campos de los formularios donde se muestra la información de los pedidos, clientes, facturas y mensajes. Para crear estos campos se ha utilizado *mat-form-field*.
- Los *checkbox* de las vistas de información de producto, del carrito, de las reseñas, entre otros. Para ello, se ha utilizado *mat-checkbox*
- El listado de productos y las subcategorías. Usando *mat-grid-list*.
- Los filtros del listado de productos. Usando *mat-accordion*.
- Las tabs de las vistas del *Login* y del *Detalle del producto*. Utilizando *mat-tab*.

4.4.4. Descarga de archivos

Para la descarga de los archivos Excel generados en todas las tablas de la trastienda (productos, clientes, etc.) se ha utilizado la biblioteca *file – saver*. Para realizar la descarga hay que llamar a la función *saveAs* pasándole como parámetros los bytes del archivo a descargar y el nombre con el que se va a guardar el archivo descargado.

4.4.5. Integración con Paypal

Para realizar el pago a través de Paypal [24] se ha integrado la aplicación con el API de Paypal. De esta forma, se usa su pasarela de pagos y se evitan errores de seguridad al no tener que trabajar con los datos bancarios ni guardarlos en la base de datos.

Se han seguido los siguientes pasos:

1. Se crea una cuenta de Paypal para la tienda *online* en el sitio web de desarrolladores.
2. En la cuenta creada anteriormente, se crea un *sandbox* que permite realizar y recibir pagos. Al crear el *sandbox* se obtiene un ID, que se utilizará para identificar la cuenta en cada pago.

3. Se modifica el HTML del componente **Métodos de pago** añadiendo un tag *script* con la URL del *script* de Paypal. Además, se añade el botón “Paypal Pagar”. Cuando se carga el componente se descarga el *script* con el código necesario, para que el botón “Paypal Pagar” funcione.
4. Se añade un objeto “paypalConfig” en el componente **Métodos de pago**. Este objeto contiene la configuración para realizar los pagos. En él se incluye el entorno (producción o *sandbox*), el ID del *sandbox*, la moneda, la cantidad a pagar y una función (*onAuthorize*) que se ejecuta cuando el pago ha sido validado.

Esta integración permite al usuario efectuar el pago en el sitio web de Paypal. Al pulsar sobre el botón “Paypal pagar” se abre la página de inicio de sesión. Después de hacer *login*, el usuario puede elegir la forma de pago que más le conviene, ya que Paypal permite realizar pagos a través de su propia cuenta Paypal, a través de tarjeta bancaria o a través de transferencia. Cuando el usuario finaliza el pago, se cierra el sitio web de Paypal y se redirecciona a la página principal de la tienda *online*, mostrando al usuario un mensaje indicando que su pedido ha finalizado correctamente. La figura 4.11 muestra este proceso.

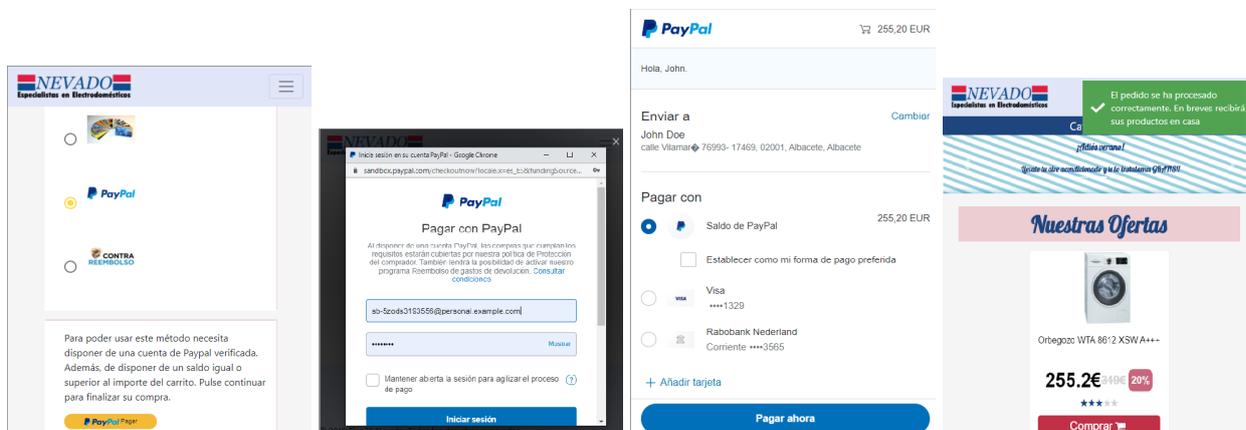
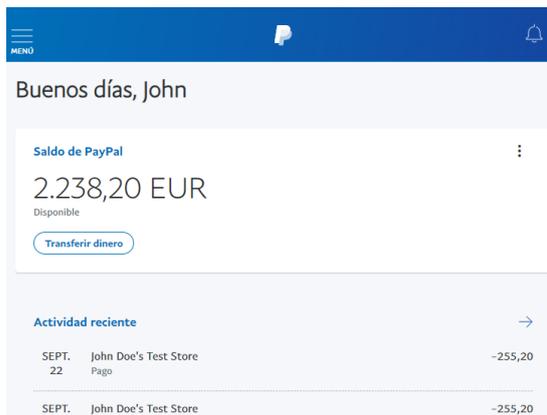


Figura 4.11: Proceso de pago con Paypal.

En la figura 4.12 se puede observar como el pago se descuenta de la cuenta del cliente y como se recibe en la cuenta de la tienda.



(a) Pago



(b) Recibí

Figura 4.12: Ejemplo cargo en la cuenta del cliente y recibí en la cuenta de la tienda.

4.4.6. Sistema de avisos

Se ha creado un sistema de avisos que permite informar al usuario del resultado de las operaciones realizadas en la tienda *online*. Este sistema muestra un *popup* en el margen superior derecho de las vistas. Si la operación ha finalizado de manera errónea, el *popup* será de color rojo y, si por el contrario, ha finalizado correctamente el *popup* será de color verde. La figura 4.13 muestra un ejemplo del formato de los avisos.

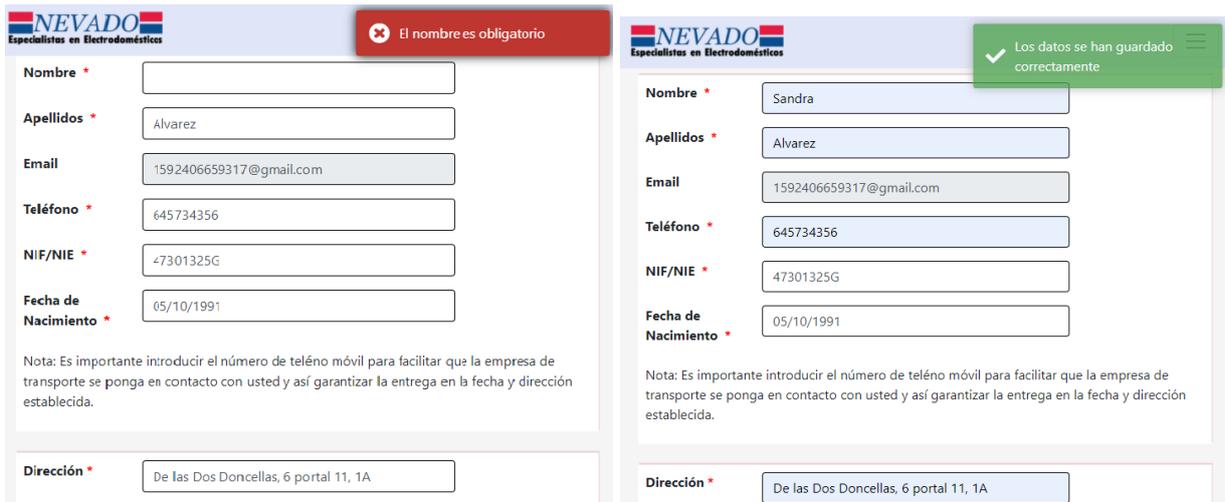


Figura 4.13: Ejemplos mensajes de aviso.

Para implementar este sistema de avisos se ha utilizado el servicio *ToastrService* de la

biblioteca de Angular *ngx-toastr*. Para ello, se llama a las funciones *error*, cuando se produce un error, y *success*, cuando el proceso es correcto. Ambas funciones reciben como parámetros el mensaje que se muestra en pantalla y el tiempo que el *popup* es visible.

4.4.7. Diseño *responsive*

El acceso a las aplicaciones web se realiza desde diferentes dispositivos: ordenadores, tablets, móviles. Cada uno de estos dispositivos tiene una resolución y un tamaño de pantalla distinto, por lo que la visualización de las páginas no es la misma. Por esta razón, las aplicaciones web tienen que adaptar sus diseños a los diferentes tamaños de pantalla.

Para que los usuarios de la tienda *online* puedan acceder a través de diferentes dispositivos, se ha creado un diseño adaptativo, denominado diseño *responsive*. Este tipo de diseño permite visualizar las vistas de la tienda *online* en cualquier pantalla. En las figuras 4.14 y 4.15 se puede observar cómo el diseño se adapta al tamaño y resolución de la pantalla.

Para adaptar las vistas se ha recurrido al uso de componentes Bootstrap como, por ejemplo, la *navbar* de la cabecera de las vistas, y al uso de las nuevas clases *d-flex*, *flex-column*, *flex-row*, entre otras. Estas clases permiten posicionar los elementos en filas o columnas sin tener en cuenta si el elemento es un elemento de bloque. La variación de rejilla que tiene Bootstrap (*sm*, *md*, *lg* y *xl*) ha permitido tener diferentes posicionamientos para los distintos tamaños de pantalla.

Para los elementos creados sin Bootstrap se ha utilizado Media Queries de CSS3, que permiten redefinir los estilos cuando el tamaño de pantalla lo requiere. Por tanto, un mismo elemento puede tener un estilo diferente para cada tamaño de pantalla.

En algunos componentes se ha recurrido a Angular utilizando la nomenclatura *class.clase*, que permite añadir clases CSS al componente en función de unas condiciones determinadas por el tamaño de pantalla. Además, se utilizan diferentes plantillas que permiten modificar la vista en función del tamaño de pantalla. El tamaño de pantalla se calcula en el evento *resize* del objeto *window* de JavaScript.

Figura 4.14: Diseño *responsive* en móvil

Por último, se ha utilizado la biblioteca Angular Material para ciertas vistas, por ejemplo para crear la tabla de pedidos que realiza un usuario. Esta biblioteca permite crear componentes *responsive* de manera sencilla.

Figura 4.15: Diseño *responsive* en ordenador

Capítulo 5

Resultados

El resultado del trabajo es satisfactorio, ya que se ha cumplido con el objetivo principal del mismo: crear una tienda *online* de electrodomésticos. Si se hace una comparativa con otras tiendas *online*, se puede decir que la aplicación desarrollada está a la altura de las que se ven en el mercado *online* de electrodomésticos.

5.1. Consecución de objetivos

Con la implementación de la aplicación se ha conseguido cumplir con los objetivos específicos que se marcaban en el capítulo 2, pues:

- La aplicación es *responsive*. Si se observan las imágenes 4.14 y 4.15, se ve como la aplicación se adapta a diferentes tamaños de pantalla. Para cumplir con este objetivo se ha recurrido al uso de Bootstrap, CSS3 y Angular Material como se ha explicado en el capítulo 4.
- La interfaz de usuario es una interfaz sencilla, en la que los componentes que se utilizan son muy intuitivos. La mayoría de estos componentes se encuentran en casi todas las web de comercio electrónico como, por ejemplo, los botones denominados *dropdown*. Además, se ha usado gran número de iconos que identifican las funcionalidades que realiza cada botón.
- La aplicación cumple con la normativa de protección de datos. Para llevar a cabo este objetivo se ha creado una política de privacidad, donde se informa al usuario como van

a ser tratados sus datos. En cuanto a la seguridad de los datos susceptibles, como son las contraseñas, se ha usado una biblioteca robusta para su cifrado. En la base de datos se guarda esta información cifrada mediante un *hash*. El *hash* esta formado por un *salt* que dificulta conseguir la información mediante ataques de fuerza bruta. En cuanto a los pagos, se realizan a través de APIs externas a la aplicación, para no tener que almacenar los datos bancarios.

- La aplicación es atractiva y competitiva. Se ha implementado las funcionalidades básicas que deben tener las tiendas *online* y, además, se ha implementado un sistema de valoración y votación de productos, que no todas las aplicaciones de comercio electrónico tienen. Esto permite a los usuarios conocer las valoraciones de anteriores compradores. Se ha recurrido al uso de colores, iconos, letras y componentes llamativos para captar la atención de los clientes. De esta manera, se amplía el mercado y la aplicación puede situarse entre las más visitadas del mercado de los electrodomésticos.
- Se ha logrado crear una trastienda de administración sencilla. Permite al administrador de la tienda a gestionar pedidos, crear productos y visualizar clientes de manera rápida, evitando que los pedidos se demoren. Para agilizar la carga masiva de los productos, las marcas y las categorías, se ha implementado un servicio de importación y exportación de ficheros Excel. Además se ha implementado un sistema de mensajería que permite interactuar entre los usuarios y el administrador de la tienda, pudiendo resolver cualquier duda de los usuarios.
- Con el uso de Angular, Express y Node.js se ha conseguido crear una aplicación web modutable y escalable, en la que el mantenimiento y la actualización de componentes es sencilla. Cada vista de la aplicación tiene su propio componente, por lo que modificar una vista significaría simplemente modificar el componente asociado a ella.

Capítulo 6

Conclusiones

6.1. Aplicación de lo aprendido

Al realizar este trabajo he puesto en práctica muchos de los conocimientos adquiridos en el grado que estoy cursando, Grado en Tecnologías de la Telecomunicación, sobre todo, aquéllos que están relacionados con el desarrollo de aplicaciones web y con la programación.

La asignatura que más me ha aportado y ayudado para hacer este trabajo es la cursada en el tercer curso del grado, Desarrollo de Aplicaciones Telemáticas, ya que fue aquí donde aprendí todo lo relacionado con la implementación de aplicaciones web como CSS3, HTML5 y Bootstrap.

En el mismo año cursé la asignatura Ingeniería de Sistemas de la Información. Fue aquí donde tuve el primer contacto con JavaScript, lenguaje de programación utilizado para desarrollar el servidor de la aplicación web de comercio electrónico Nevado. Además, aprendí a manejar Git, que me ha permitido llevar un control de las versiones que iba teniendo del proyecto, de tal forma que si alguna funcionalidad me fallaba podía seguir con otra.

Asignaturas como Fundamentos de la Programación o Programación de Sistemas Telemáticos han significado mucho, ya que aprendí los conocimientos más básicos de la programación.

Para implementar una aplicación, como la desarrollada en este trabajo, hay que tener conocimientos sobre las peticiones HTTP y saber lo que es un API Rest. Estos conocimientos los adquirí en las asignaturas Sistemas telemáticos y Servicios y aplicaciones telemáticas.

Por último, indicar que aunque muchas de las asignaturas que cursé en esta etapa universitaria no han sido necesarias para este trabajo, sí que me han servido para aprender que todo

requiere un esfuerzo y un aprendizaje y que, sin este esfuerzo y aprendizaje, no se consiguen las metas que te propones.

6.2. Lecciones aprendidas

El desarrollo de este trabajo ha supuesto un gran reto, pues es el primer proyecto que he realizado de esta complejidad. Durante el transcurso del grado han sido muchas las prácticas que he realizado, pero en todas tenía las directrices a seguir. A lo primero que me he enfrentado en este trabajo ha sido a la recapitulación de los requisitos que se necesitaban para implementar la aplicación. Este análisis me ha servido para conocer los puntos más importantes del comercio electrónico.

Al analizar los requerimientos he tenido que hacer un análisis de las distintas arquitecturas que podría tener la aplicación. Esto me ha servido para conocer los distintos tipos de arquitecturas que se utilizan en las aplicaciones de este negocio. Como he mencionado en el Capítulo 4, me decanté por la arquitectura MEAN, ya que era la que más se ajustaba a mis necesidades. El haber implementado esta arquitectura me ha permitido conocer nuevas tecnologías: MongoDB, Express, Angular y Node.js. He aprendido a programar de una manera diferente, ya que con Node.js se programa de manera asíncrona. Con Angular, me he tenido que adentrar en el mundo de TypeScript y en el mundo de las aplicaciones modulares y escalables.

Durante el grado, han sido pocas las veces que he usado base de datos, y cuando las he usado ha sido de manera muy básica, con operaciones sencillas y usando siempre SQL. Utilizar MongoDB me ha permitido aprender otra forma de guardar los datos a través de documentos, en vez de en tablas como hasta ahora. Además, esta aplicación me ha permitido realizar operaciones mucho más complejas, en las que en una misma operación tenía que tocar varias colecciones (tablas en base de datos relacionales). Para ello, he tenido que aprender a hacer agregaciones, que son funciones en las los datos se van transformando, a través de etapas, hasta que se obtiene el resultado esperado.

Para realizar un diseño atractivo y que se adapte a los tamaños de pantalla más estándares, he aprendido a usar media *queries* de CSS3 y he profundizado en Bootstrap. Además, he conocido la biblioteca Angular Material de Angular.

La importación y exportación de ficheros, así como la subida de archivos al servidor de

Node.js me ha proporcionado tener conocimientos en las bibliotecas adm-zip, para la decomposición de zip, excel4node para el manipulado de ficheros Excel y multer, para la subida de imágenes al servidor.

Por último, con este trabajo he aprendido que los grandes logros se consiguen con constancia, dedicación y sacrificio y que, a pesar de todos los problemas que pueden aparecer en el camino, hay que levantarse y luchar por conseguirlos. Que la desesperación y los nervios no son buenos aliados. Que hay que confiar en uno mismo y no ponerse límites.

6.3. Trabajos futuros

Todas las aplicaciones web necesitan mantenimiento y la implementada en este trabajo no iba a ser menos. Algunas de las líneas a seguir para mejorar la aplicación son:

- Adaptar la aplicación para que soporte, como segundo lenguaje el inglés. De esta forma, se abriría el mercado al mundo internacional. Para realizar esta mejora, se debería utilizar la biblioteca de angular Ngx-translate. Ngx-translate utiliza ficheros JSON donde se encuentran los textos estáticos de la aplicación traducidos a los idiomas que se desee utilizar. Tiene que existir un fichero por cada idioma y el formato de los ficheros debe ser igual. Para los textos que varían en la aplicación, como, por ejemplo, la descripción de los productos, se debería crear un campo nuevo dentro de la base de datos en el que se encuentre la traducción y a la hora de hacer la petición, mandar el idioma para saber que descripción obtener.
- Ampliar el filtrado de productos. A pesar del número de filtros que tiene la aplicación para la búsqueda de productos, se pueden añadir filtros que se adapten a las distintas categorías. Por ejemplo, si se accede a la categoría de televisores, que aparezca un filtro más para buscar por pulgadas.
- Añadir una lista de favoritos, donde el cliente pueda guardar los productos que le gustan. Así, si vuelve a entrar en la aplicación no tiene que volver a buscar los productos que desea.
- Inicio de sesión a través de Facebook y Google. De este modo, los clientes no tendrían

que darse de alta en la aplicación, bastaría que iniciasen sesión con sus credenciales de Facebook y Google.

En cuanto a la parte de administración se podrían introducir las siguientes mejoras:

- Añadir estadísticas y gráficos que ofrezcan información sobre los productos más vendidos, los menos demandados, así como la zona donde más pedidos se realizan. Tener esta información de manera resumida, en forma de gráficos, sería muy útil para el administrador y dueño, ya que sabría lo que debe reforzar para mejorar el servicio.
- Añadir una opción de impresión de cartelera. En la tienda física, se necesitan carteles que promocionen los productos, que contengan la información y precio de los mismos. Toda esta información está almacenada en la base de datos, por lo que con la impresión de estos carteles, a través de la aplicación, se agilizaría el trabajo en tienda física.
- Mejorar el diseño *responsive* de la parte de administración. La aplicación en la parte de administración es *responsive*, pero en ciertas pantallas, los campos de las tablas salen más amontonados, por lo que se podría reducir el número de campos a mostrar en la tabla para que dicha información se vea mejor.

Apéndice A

Manual de Instalación

Para instalar la aplicación y poder utilizarla, en un equipo Windows y en el entorno local, se deben seguir los siguientes pasos:

1. Descargar *Git* desde:

<https://git-scm.com/download/win>.

2. Instalar *Git*.

3. Clonar el repositorio de *Git* con la instrucción:

`git clone https://github.com/sandri1304/tfg`

4. Descargar *Node.js* y *npm* desde:

<https://nodejs.org/es/>

5. Instalar *Node.js*. La versión utilizada en la aplicación es 10.16.13 LTS. Esta versión incluye *Node.js* y *npm*.

6. Instalar *Nodemon* con la instrucción:

`npm install -g nodemon`

7. Instalar *Angular CLI* con la instrucción:

`npm install -g @angular/cli`

8. Instalar *MongoDB* desde:

<https://www.mongodb.com/download-center/community>

Con la instalación de *MongoDB* se instala *Mongo Compass* (herramienta gráfica que permite interactuar con la base de datos MongoDB).

9. Importar la base de datos.

mongorestore -d Nevado ./Escritorio/Nevado

Para iniciar la aplicación se incluyen dos *scripts* .bat dentro del directorio raíz del proyecto, que permiten arrancar la aplicación. Los *scripts* son:

- *server.bat*, ejecuta el servidor de *Node.js*. Utilizado para arrancar el servidor de la aplicación.
- *client.bat*, ejecuta el servidor de *Angular CLI* con la configuración del *proxy*. Utilizado para arrancar el cliente de la aplicación.

Apéndice B

Manual de usuario

B.1. Registro de un nuevo usuario

Para que un nuevo usuario pueda registrarse en la tienda *online* ningún otro usuario debe tener abierta una sesión. Para realizar el registro el usuario deber pulsar el enlace “Entrar”.



Figura B.1: Enlace entrar para registrar usuarios.

Al pulsar sobre el enlace “Entrar” se visualiza el formulario que permite registrar al usuario en la aplicación. El usuario debe rellenar todos los campos que son obligatorios. Cuando se introduce la contraseña en el formulario se comprueba la fortaleza de la misma.

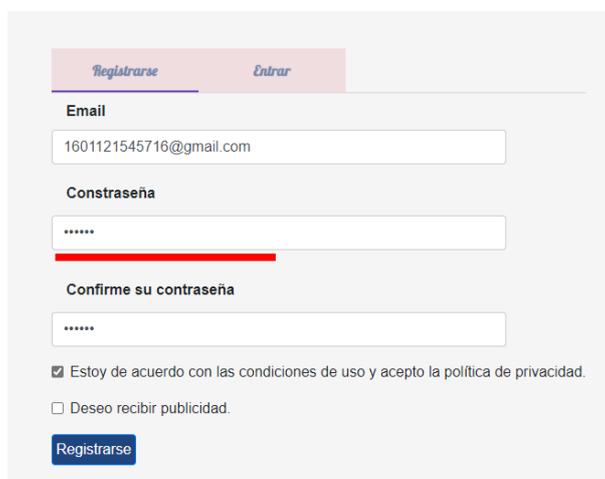
A screenshot of the user registration form. At the top, there are two tabs: 'Registrarse' (selected) and 'Entrar'. The form contains the following fields: 'Email' with the value '1601121545716@gmail.com'; 'Contraseña' with a red strength indicator bar; and 'Confirme su contraseña'. Below the fields, there are two checkboxes: 'Estoy de acuerdo con las condiciones de uso y acepto la política de privacidad.' (checked) and 'Deseo recibir publicidad.' (unchecked). At the bottom, there is a blue 'Registrarse' button.

Figura B.2: Formulario registro usuarios.

Si alguno de los campos no tiene un formato correcto o no está relleno y es obligatorio, se muestra un mensaje de error por pantalla y se impide el registro del usuario.



Figura B.3: Error al registrar un usuario.

Cuando el registro finaliza correctamente se muestra un mensaje indicando al usuario que debe confirmar la cuenta de usuario para completar el registro. Para confirmar la cuenta, se le envía al usuario un enlace a su correo electrónico. El usuario debe pulsar el enlace recibido para completar el registro. Un usuario que no confirma su cuenta no puede realizar compras en la tienda.

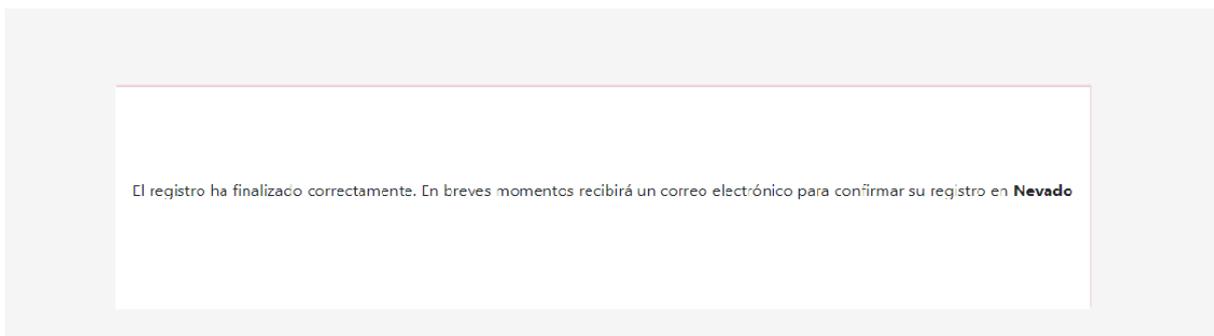


Figura B.4: Mensaje de confirmación de cuenta.

B.2. Inicio de sesión

El usuario debe pulsar el enlace “Entrar” de la barra de navegación.



Figura B.5: Enlace entrar para iniciar sesión.

Al pulsar el enlace “Entrar” se visualizan dos pestañas, una para el registro de usuarios y otro para iniciar sesión. El usuario debe pulsar sobre la pestaña “Entrar”. Al pulsar sobre dicha pestaña se muestra el formulario que permite acceder a la cuenta de usuario. Para iniciar sesión, el usuario debe rellenar la contraseña y el *email* y pulsar el botón “Entrar”.

Figura B.6: Formulario para iniciar sesión.

Si la contraseña o el *email* es incorrecto se muestra un mensaje de error y no se permite el acceso a la cuenta.

Figura B.7: Error al iniciar sesión.

Si tanto la contraseña como el *email* son correctos, se redirecciona al usuario a la página principal de la tienda *online* y se muestra la barra de navegación del usuario logueado, que

contiene un enlace para acceder al perfil de usuario, otro para acceder a los pedidos realizados, otro para mostrar el carrito y otro para cerrar sesión. Si el inicio de sesión se realiza cuando se está tramitando el pedido, al finalizar el *login* se redirecciona a la página *Perfil de usuario*.



Figura B.8: Página principal de la tienda.

B.3. Restablecer contraseña

Para restablecer la contraseña el usuario debe pulsar el enlace “Entrar” de la barra de navegación y seleccionar la pestaña “Entrar”. En esta pestaña debe hacer click sobre el enlace “¿Has olvidado la contraseña?”.

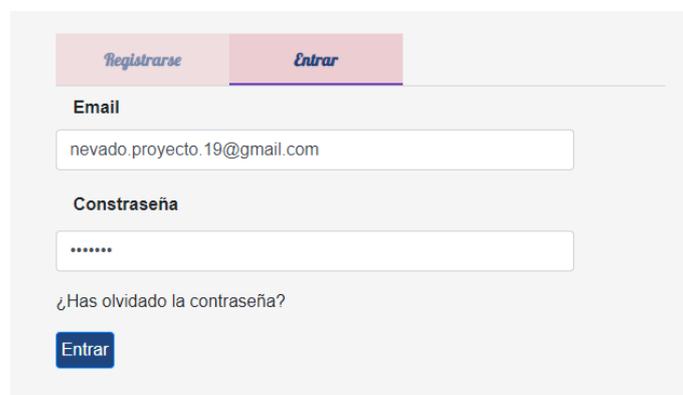


Figura B.9: Enlace para restablecer contraseña.

Al pulsar sobre este enlace se visualiza el formulario que permite restablecer la contraseña. Para ello, el usuario debe introducir el correo electrónico con el que se registró en la aplicación.



The screenshot shows a web form titled "¿Olvidaste la contraseña?". Below the title is a sub-header: "Si Introduce el email para recuper la contraseña. Se le enviará un correo para terminar el proceso de recuperación". There is a text input field labeled "Email" and a blue button labeled "Recuperar contraseña".

Figura B.10: Formulario para restablecer la contraseña.

Si el correo introducido no coincide con ninguno de los registrados en la tienda *online* se muestra un mensaje y se impide restablecer la contraseña.



The screenshot shows the same password recovery form as in Figure B.10, but with an error message. At the top left is the NEVADO logo with the tagline "Especialistas en Electrodomésticos". At the top right, a red banner displays "✘ Email incorrecto.". The form title "¿Olvidaste la contraseña?" and sub-header are present. The "Email" input field now contains "gfda@gmail.com". The "Recuperar contraseña" button is still visible.

Figura B.11: Correo no válido al restablecer contraseña.

Si el correo está registrado en el sistema, se envía un correo electrónico con el enlace que permite modificar las contraseñas. Al abrir el enlace, se muestra el formulario a rellenar para el cambio de contraseña. Este formulario tiene dos campos, "Contraseña nueva" y "Confirmación de la contraseña nueva".

The image shows a web form titled "Recuperación contraseña" (Password Recovery). It contains two input fields: "Contraseña nueva" (New password) and "Confirma la nueva contraseña" (Confirm new password). Both fields are filled with seven dots. Below the fields is a blue button labeled "Cambiar contraseña" (Change password).

Figura B.12: Formulario cambio de contraseña.

Para modificar la contraseña, el usuario debe introducir la misma contraseña en los dos campos y pulsar el botón “Cambiar contraseña”. Si al pulsar el botón, las dos contraseñas no coinciden se muestra un mensaje de error y no se procede al cambio de contraseña. Si las contraseñas coinciden se actualiza en la base de datos y se redirecciona al usuario al *login*.

This screenshot shows the same password recovery form as in Figure B.12, but with an error message. At the top right, a red notification box contains a white 'x' icon and the text "Las contraseñas no coinciden" (Passwords do not match). The "Contraseña nueva" field has seven dots, while the "Confirma la nueva contraseña" field has only four dots. The "Cambiar contraseña" button is still visible.

Figura B.13: Error cambio de contraseña.

B.4. Cerrar sesión

Para cerrar sesión, el usuario debe pulsar el enlace “Salir” de la barra de navegación.

The image shows a navigation bar for the website. On the left is the logo for "NEVADO Especialistas en Electrodomésticos". On the right, there is a user ID "1592406659317", a shopping cart icon with "Pedidos 1", and a "Salir" (Logout) link.

Figura B.14: Enlace Salir.

B.5. Perfil de usuario

Un usuario puede acceder a su información personal siempre que haya iniciado sesión en su cuenta. Para ello, debe pulsar sobre el enlace que lleva su nombre en la barra de navegación.



Figura B.15: Barra de navegación acceso al perfil de usuario.

Al pulsar sobre el enlace se visualiza un formulario con sus datos personales. La primera vez que acceda a su perfil de usuario todos los campos estarán vacíos, excepto el correo electrónico. El usuario puede modificar sus datos personales tantas veces como lo desee.

The image shows a web form titled 'Introduce tus datos personales' (Enter your personal data). The form contains several input fields with the following pre-filled data: 'Nombre' (Alejandro), 'Apellidos' (Cortes Gomez), 'Email' (1601136589233@gmail.com), 'Teléfono' (645734859), 'NIF/NIE' (47302514F), and 'Fecha de Nacimiento' (31/08/1988). Below these fields is a note: 'Nota: Es importante introducir el número de teléfono móvil para facilitar que la empresa de transporte se ponga en contacto con usted y así garantizar la entrega en la fecha y dirección establecida.' The second section of the form contains address fields: 'Dirección' (Méndez Álvaro, 4, 1-B), 'Código Postal' (28904), 'Población' (Getafe), 'Provincia' (Madrid), and 'País' (España). At the bottom, there is a checkbox labeled '¿Desea utilizar otra dirección para el envío?' (Do you want to use another address for shipping?). The form has 'Volver' (Back) and 'Guardar' (Save) buttons at the bottom.

Figura B.16: Perfil de usuario.

El usuario puede tener una dirección de facturación distinta que la dirección donde se van a enviar los pedidos. Para agregar la dirección de envío, debe marcar la opción “¿Desea utilizar otra dirección para el envío?”. Al marcar esta opción aparece un formulario donde se rellenan los datos de la nueva dirección. Si la opción está marcada, los campos de la dirección de envío son obligatorios.

¿Desea utilizar otra dirección para el envío?

Dirección *

Código Postal *

Población *

Provincia *

País *

Volver Guardar

Figura B.17: Formulario para los datos de la dirección de envío.

Para modificar sus datos, debe rellenar los campos que son obligatorios y pulsar el botón “Guardar”. Si un campo es incorrecto se muestra un error y no se actualizan los datos.

NEVADO
Especialistas en Electrodomésticos

16 El nombre es obligatorio

Nombre *

Apellidos *

Email

Teléfono *

NIF/NIE *

Fecha de Nacimiento *

Nota: Es importante introducir el número de teléfono móvil para facilitar que la empresa de transporte se ponga en contacto con usted y así garantizar la entrega en la fecha y dirección establecida.

Dirección *

Código Postal *

Población *

Provincia *

País *

¿Desea utilizar otra dirección para el envío?

Volver Guardar

Figura B.18: Error al actualizar los datos personales.

B.6. Realizar compra

El usuario debe acceder a la página principal de la tienda *online*.



Figura B.19: Página principal de la tienda.

Si en la página principal encuentra el producto que quiere comprar, puede tramitar directamente el pedido o ir a la página *Detalle del producto*. Si en esta página no encuentra lo que desea debe seleccionar una categoría del menú de categorías generales.



Figura B.20: Menú categorías generales.

Al seleccionar una categoría se redirecciona a la página de *subcategorías* de la categoría general. En esta página debe elegir la subcategoría donde se encuentra el producto que busca.



Figura B.21: Subcategorías.

Para seleccionar la subcategoría, el usuario debe pulsar sobre la imagen. Al pulsar, se visualiza el listado de productos que tiene la subcategoría seleccionada.

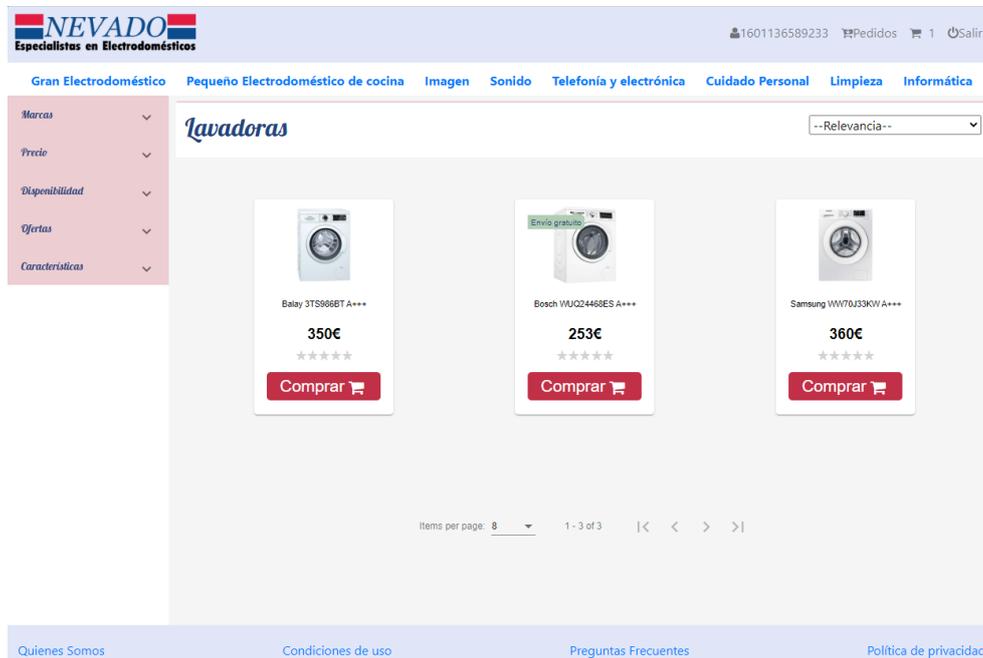


Figura B.22: Productos de la subcategoría.

El usuario puede ordenar los productos de más baratos a más caros, de más caros a más baratos, los más vendidos y los mejores valorados.

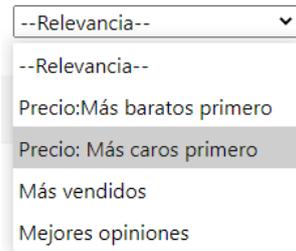


Figura B.23: Ordenación de productos.

Además, puede filtrar los productos por marca, precio, disponibilidad, ofertas y características.



Figura B.24: Filtros.

Desde esta vista, el usuario puede añadir directamente el producto al carrito pulsando sobre

el botón “Comprar” o ir al detalle del producto. Si el usuario decide ir al detalle del producto debe pulsar sobre la foto del producto. En la vista de *detalle del producto* se visualiza la imagen del producto, el logo de la marca, el modelo, las estrellas y opiniones que tiene el producto, el precio del envío, la disponibilidad, el precio, las características y las opiniones de los usuarios.

The screenshot shows the product detail page for a Bosch washing machine. The header includes the NEVADO logo and navigation links for various product categories. The main content area features a large image of the washing machine, the Bosch logo, and the model number WUQ24468ES A+++ in a stylized font. Below the model number, there is a star rating of 4.5 and a link to write a review. The shipping information indicates free delivery. The availability section states that the product can be received at home in two days. The price is shown as 202.4€ with a 20% discount from the original price of 253€. A red 'Comprar' button is prominently displayed. At the bottom, there are tabs for 'Características' and 'Reseñas', with the 'Reseñas' tab selected. The 'Reseñas' section shows a single review with a 4.5-star rating and a brief description of the product's features.

NEVADO
Especialistas en Electrodomésticos

1601136589233 Pedidos 1 Salir

Gran Electrodoméstico Pequeño Electrodoméstico de cocina Imagen Sonido Telefonía y electrónica Cuidado Personal Limpieza Informática

BOSCH *Oferta*

WUQ24468ES A+++

★★★★★ 1 opinión | [Escriba una reseña](#)

Envío: Envío Gratuito

Disponibilidad: Recíbelo en casa en dos días. Plazo de entrega estimado

Promociones: Descuento finalización proyecto

202.4€ ~~253€~~ 20%

Comprar

[Características](#) [Reseñas](#)

Lavadora EcoSilence: silenciosa y eficiente.

- A+++ -30%: consume un 30% menos que la mejor clase de eficiencia energética del mercado.
- Gran display con indicación de tiempo restante y fin diferido.

Figura B.25: Detalle del producto.

Para visualizar las opiniones del producto debe pulsar sobre la pestaña “Reseñas”. Estas reseñas se pueden filtrar por número de estrellas. Para ello, el usuario debe pulsar sobre la fila de estrellas que desea buscar.

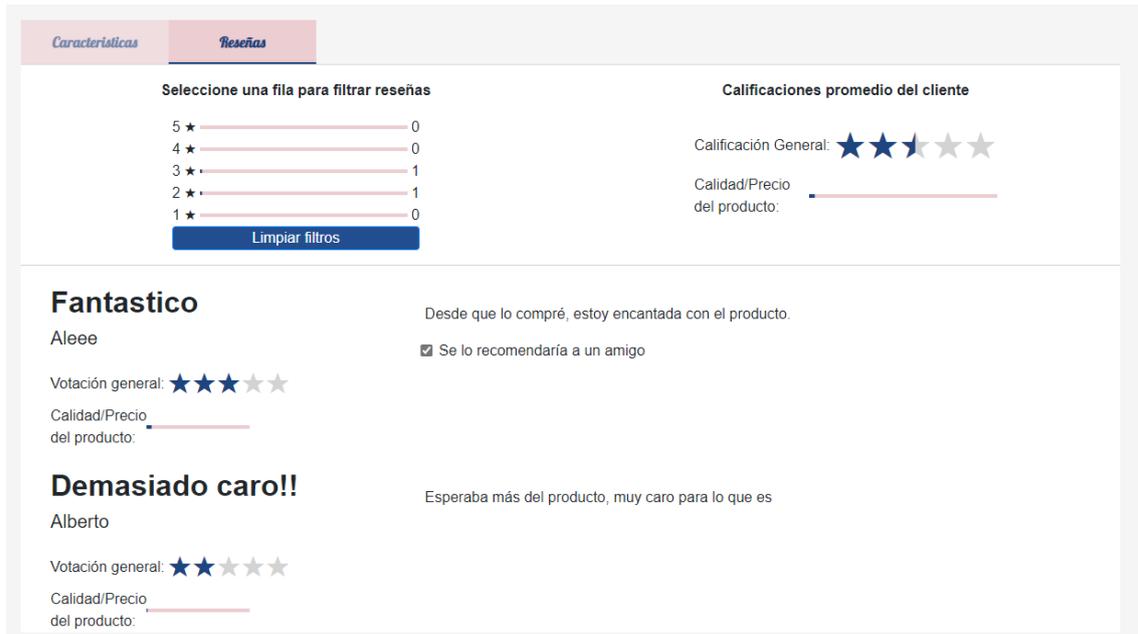


Figura B.26: Reseñas del producto.

Para añadir el producto al carrito, el usuario debe pulsar sobre el botón “Comprar”. Al pulsar sobre este botón, se redirecciona a la vista de *carrito*, donde el usuario ve los productos que tiene guardados en el carrito. Puede aumentar o disminuir la cantidad de un mismo producto pulsando sobre los botones “+” o “-”, elegir la forma de envío (envío a domicilio o recogida en tienda), siempre y cuando los productos sean diferentes, y eliminar todos los productos del carrito pulsando sobre el botón “Vaciar cesta”. Si el usuario quiere seguir comprando, debe pulsar el botón “Seguir comprando”. Si desea finalizar el pedido debe pulsar sobre el botón “Tramitar Pedido”.

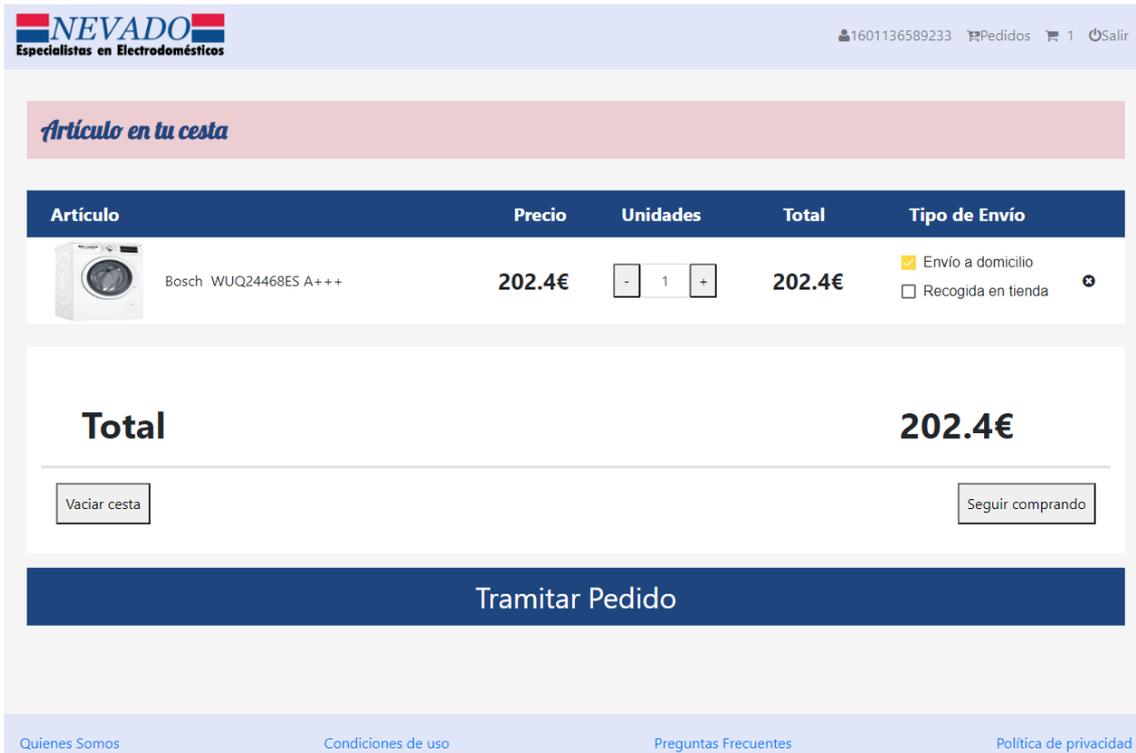


Figura B.27: Carrito.

Al pulsar sobre el botón “Tramitar pedido” se visualizan los datos personales del usuario (*perfil de usuario*), ya que el usuario debe comprobar que la información guardada es correcta.

NEVADO
Especialistas en Electrodomésticos

1601136589233 Pedidos 1 Salir

Introduce tus datos personales

Nombre *

Apellidos *

Email

Teléfono *

NIF/NIE *

Fecha de Nacimiento *

Nota: Es importante introducir el número de teléno móvil para facilitar que la empresa de transporte se ponga en contacto con usted y así garantizar la entrega en la fecha y dirección establecida.

Dirección *

Código Postal *

Población *

Provincia *

País *

¿Desea utilizar otra dirección para el envío?

Volver Continuar Pedido

Figura B.28: Perfil de usuario.

Para finalizar el pedido, el usuario debe pulsar sobre el botón “Continuar pedido”. Al pulsar

sobre este botón, se visualizan los métodos de pago disponibles: Tarjeta de crédito, Paypal y contra reembolso.

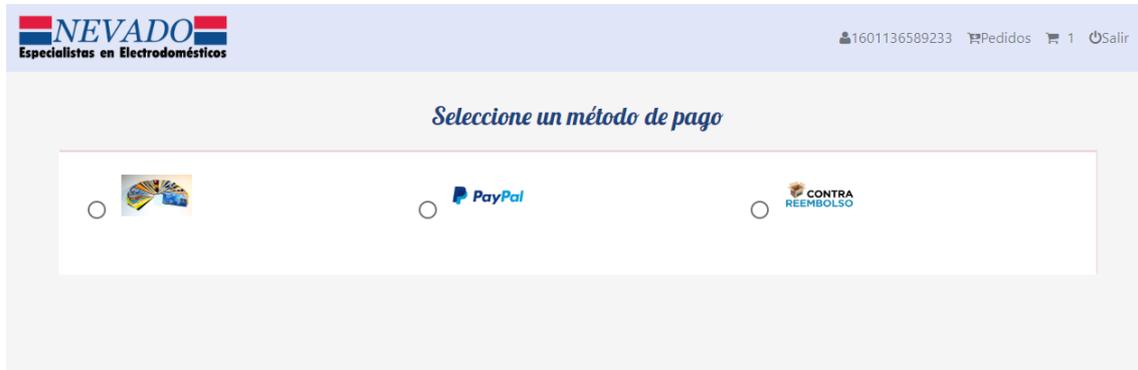


Figura B.29: Métodos de pago.

Al seleccionar Tarjeta de crédito, se muestra un formulario donde se han de rellenar los datos relativos a la tarjeta de crédito. Los datos de la tarjeta de crédito deben ser correctos, ya que se comprueba que su formato sea válido.

Figura B.30: Formulario tarjeta de crédito.

Si al finalizar el pedido, pulsando sobre el botón “Finalizar Pedido” se produce un error, como, por ejemplo, el número de tarjeta es erróneo o el CCV tiene una longitud superior a tres, se muestra un mensaje en la pantalla y no se finaliza el pedido.

NEVADO
Especialistas en Electrodomésticos

16

El número de la tarjeta de crédito no tiene el formato correcto

Seleccione un método de pago

Tipo tarjeta *
 MasterCard

Nombre Titular*
 Alejandro Cortes Gómez

Fecha Expiración*
 09/2022

Número Tarjeta*
 5405256026253

CCV*
 154

Finalizar Pedido

Figura B.31: Error al finalizar pedido con tarjeta de crédito.

Si los datos son correctos, se procesa el pedido y al usuario se le redirecciona a la página principal de la tienda.

Si el usuario elige como método de pago Contra reembolso, debe marcar la opción contra reembolso y pulsar sobre el botón “Finalizar Pedido”, ya que el pago se realiza cuando el producto llegue a su domicilio. Al finalizar el pedido, se redirecciona al usuario a la página principal de la tienda, como con tarjeta de crédito.

Seleccione un método de pago

El pago se realizará en el momento de la entrega del producto. Pulse continuar para finalizar su comprar.

Finalizar Pedido

Figura B.32: Método de pago. Contra reembolso.

Si el usuario decide que va a realizar el pago a través de Paypal, debe marcar la opción Paypal y pulsar sobre el botón “Paypal pagar”.

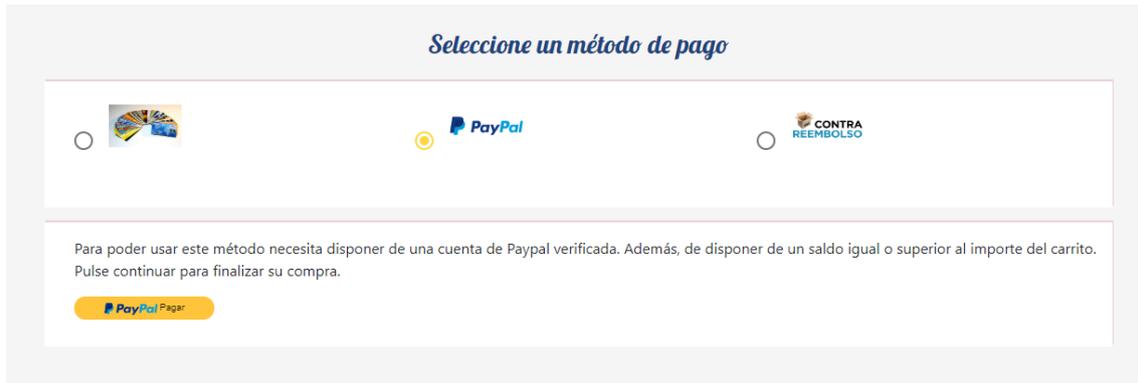


Figura B.33: Método de pago. Paypal.

Cuando pulsa sobre este botón, se abre la ventana de inicio de sesión del sitio web de Paypal, donde debe introducir sus credenciales.

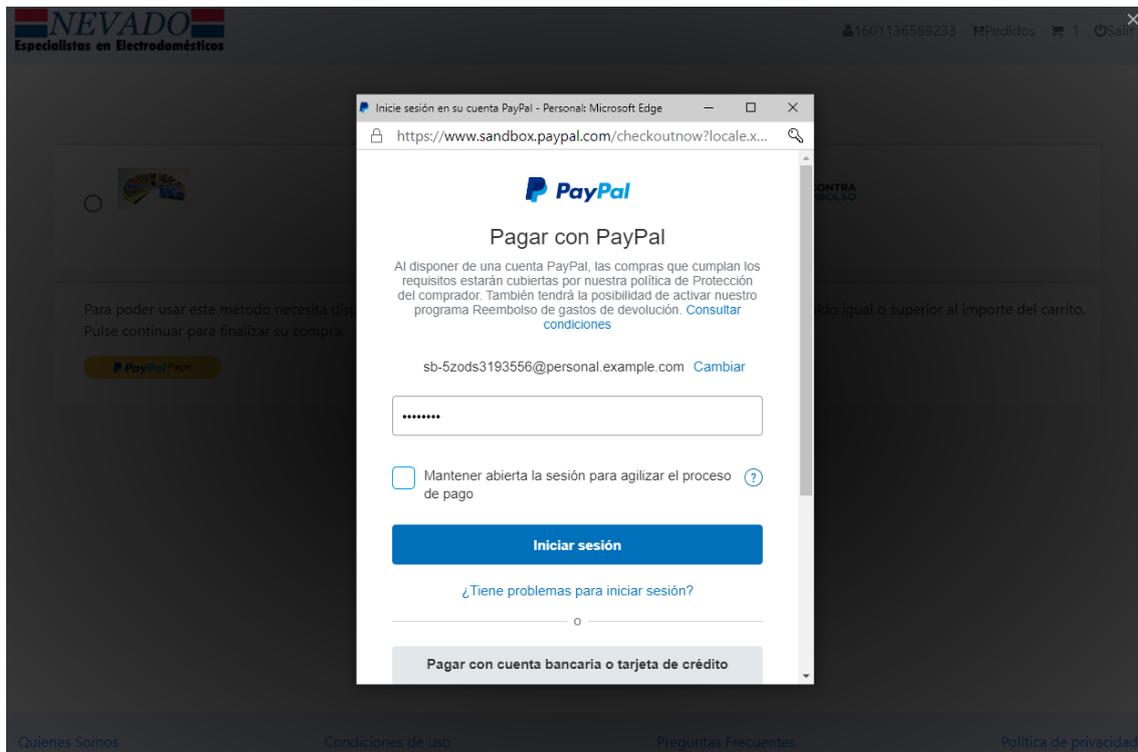


Figura B.34: Inicio sesión en Paypal.

Una vez el usuario haya iniciado sesión en su cuenta de Paypal, debe elegir como realizar el pago, ya que Paypal permite pagar a través de tarjeta de crédito, transferencia o su cuenta de Paypal. Seleccionada la opción deseada, se pulsa sobre “Pagar ahora” para finalizar el pedido. Si todo ha ido correctamente, se cerrará el sitio web de Paypal y se redirecciona al usuario a la

página principal de la tienda.

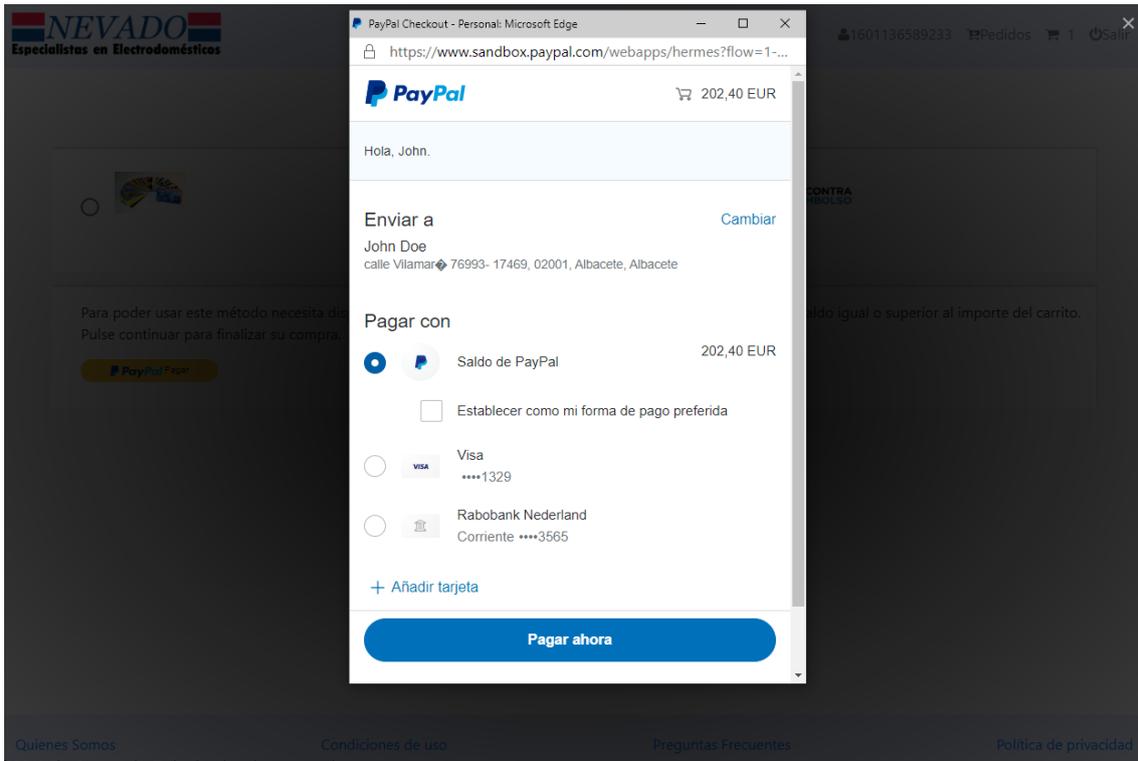


Figura B.35: Pago con Paypal.

B.7. Tus pedidos

El usuario puede ver los pedidos que ha realizado en la tienda *online* y conocer el estado de cada uno de ellos. Para ello, debe pulsar sobre el enlace “Pedidos” de la barra de navegación.



Figura B.36: Enlace pedidos.

Al pulsar sobre el enlace, se muestran los pedidos del usuario.

Id. Pedido	Fecha Entrada del Pedido	Fecha de Entrega aprox	Estado
46975	27-09-2020	30-09-2020	Pendiente envío
47886	27-09-2020	30-09-2020	Pendiente envío

Figura B.37: Tabla de pedidos de un usuario.

El usuario puede filtrar los pedidos por identificador del pedido, estado y fecha en la que se realizó el pedido. Además, puede seleccionar los pedidos que quiere mostrar en la tabla.

B.8. Añadir reseña

El usuario puede escribir reseñas de los productos de la tienda. Pero, para ello, debe haber iniciado sesión (ver sección B.2). Una vez que el usuario ha iniciado sesión, debe ir a la vista del *detalle del producto* y pulsar sobre “Escriba una reseña”.

Figura B.38: Enlace para escribir una reseña.

Al pulsar sobre el enlace se visualiza el formulario para publicar la reseña. El usuario debe rellenar la calificación general, el título, el comentario, si recomendaría el producto y la valoración del producto en cuanto a calidad precio. Además, de indicar su alias y su correo.

Figura B.39: Ventana para publicar la reseña.

Para publicar la reseña debe pulsar sobre el botón “Publicar Reseña”.

B.9. Preguntas frecuentes

Para acceder a la vista de *preguntas frecuentes*, el usuario debe dirigirse al pie de página de la aplicación y pulsar sobre el enlace “Preguntas frecuentes”.

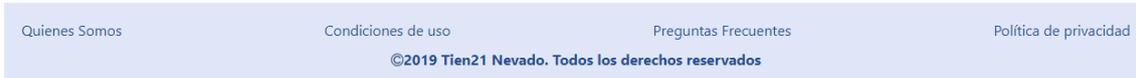


Figura B.40: Enlace preguntas frecuentes.

Para ver las respuestas de las preguntas, el usuario debe pulsar sobre la pregunta. Al pulsar sobre la pregunta se muestra la respuesta.



Figura B.41: Preguntas frecuentes.

Desde esta vista, el usuario puede enviar una pregunta al administrador de la tienda. Para ello, debe rellenar el formulario de “Envía tu pregunta”, pulsar sobre el botón “Enviar” y haber iniciado sesión en la tienda.

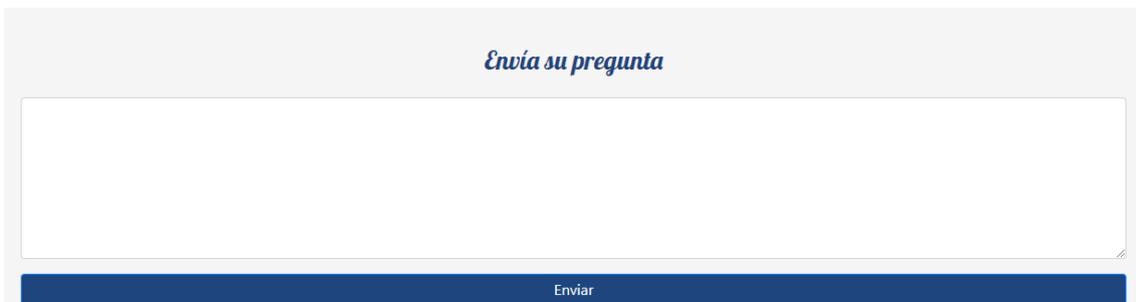


Figura B.42: Envía su pregunta.

B.10. Quiénes somos

En esta página se encuentra la ubicación de la tienda física, el teléfono y el correo donde dirigirse si el usuario tiene algún problema. Para acceder a esta página el usuario debe pulsar sobre el enlace de “Quiénes somos” del pie de página.



Figura B.43: Enlace quiénes somos.

Al pulsar sobre el enlace se muestra la página de *Quiénes somos*.

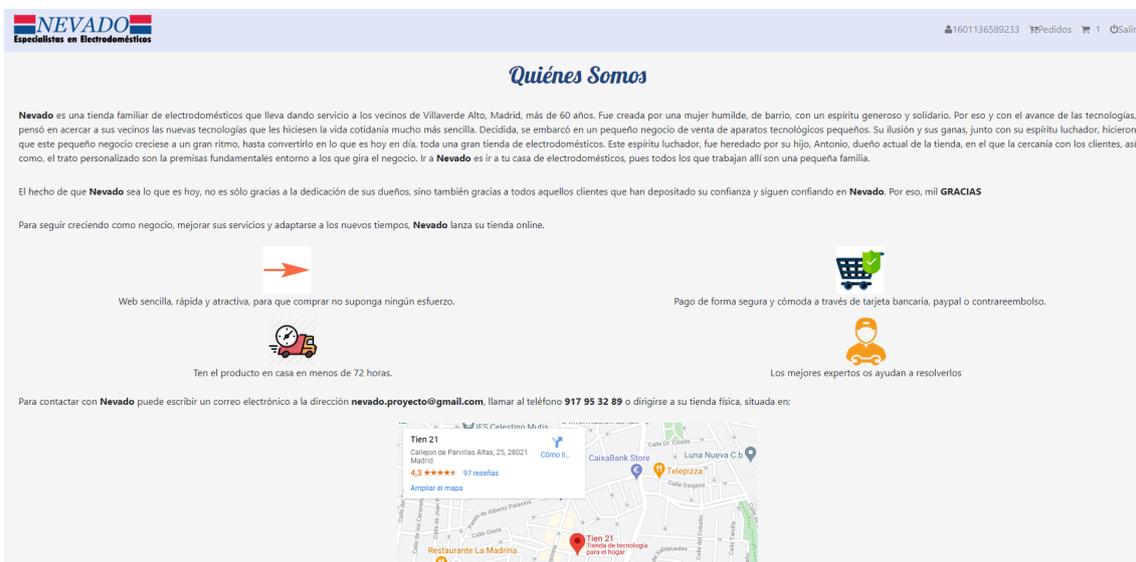


Figura B.44: Quiénes somos.

B.11. Condiciones de uso

En esta página se definen las condiciones de uso. Se explica como deben efectuarse los pedidos, como se realizan los envíos, como hay que hacer las devoluciones y como es la garantía de los productos.

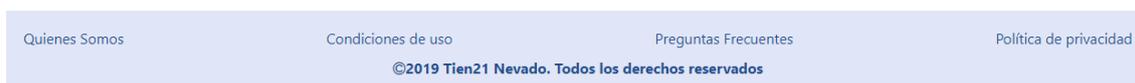


Figura B.45: Enlace condiciones de uso.

Al pulsar sobre el enlace se muestra la página de *Condiciones de uso*.

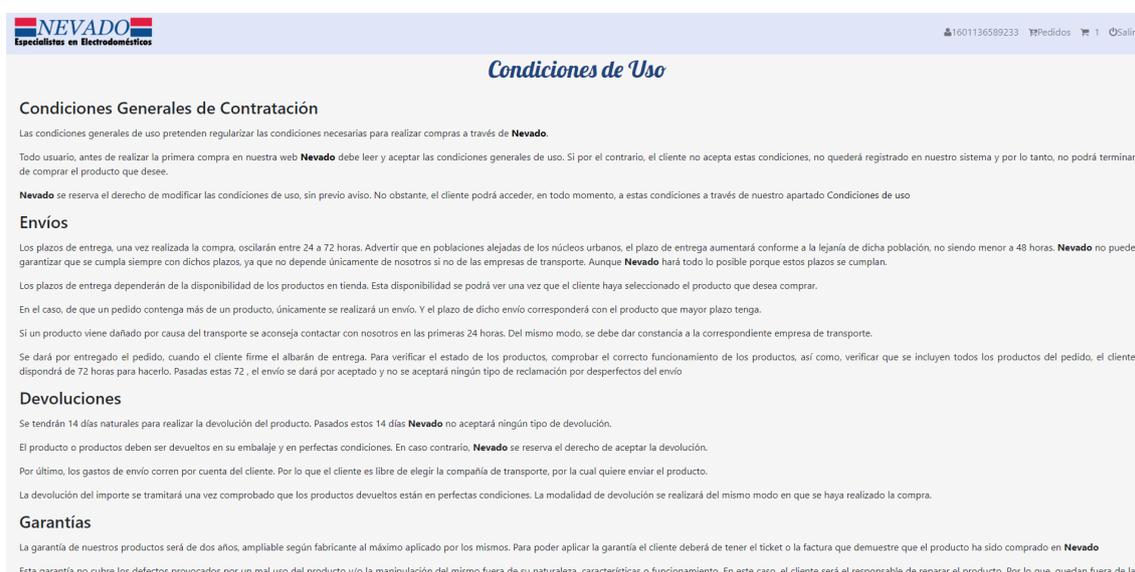


Figura B.46: Condiciones de uso.

B.12. Política de privacidad

En esta página se define la política de privacidad. Por la Ley de Protección de Datos la tienda está obligada a informar a los usuarios como van a ser tratados sus datos. Y son estos, los que deciden si lo aceptan o no. Para acceder, el usuario debe pulsar sobre el enlace “Política de privacidad”.

Figura B.47: Enlace política de privacidad.

Al pulsar sobre el enlace se muestra la página de *Política de privacidad*.

NEVADO
Especialistas en Electrodomésticos

1601136589233 Pedidos 1 Salir

Política de Privacidad

La responsabilidad de los datos y del tratamiento de los mismos corren a cargo de **Nevado**. Cuya dirección física es la calle Parvillas Altas, 36, Madrid. El correo electrónico para cualquier consulta sobre la privacidad y el tratamiento de sus datos es nevado.proyecto@gmail.com

¿Cómo se obtienen sus datos?

La obtención de sus datos se realiza a través de los distintos formularios que contiene este sitio web. Estos formularios son:

- Formulario de registro
- Formulario de acceso
- Formulario de confirmación de pedido
- Formulario del perfil de usuario

El usuario declara que los datos personales facilitados a través de este sitio web son veraces. Debe saber que será el único responsable de los posibles daños ocasionados a **Nevado** si los datos cumplimentados en los formularios de este sitio web son falsos o pertenecen a terceras personas causando daño, engaño o perjuicio.

El envío de datos personales a través de este sitio web por usuarios menores a 13 años queda prohibido. En el caso de detectarse usuarios menores de 13 años no se tratarán los datos.

¿Qué datos son tratados por Nevado?

Los datos tratados por **Nevado** y, teniendo en cuenta la actividad que realiza son los siguientes:

- Llevar a cabo la venta de productos y servicios.
- Facturación y contabilidad.
- Cobros e impagos.
- Presupuestos y contratos.
- Atención al cliente.
- Resolución de dudas o problemas.

¿Para qué puede tratar Nevado los datos personales?

Figura B.48: Política de privacidad.

B.13. Trastienda

Para entrar en la trastienda, el administrador inicia sesión tal como se indica en la sección B.2. Cuando se inicia sesión, la cabecera de la trastienda muestra un enlace para cerrar sesión y un icono con el número de pedidos pendientes de procesar.



Figura B.49: Barra de navegación del administrador.

El menú que se muestra en la figura B.50 permite acceder a cada una de las secciones a administrar.



Figura B.50: Menú.

B.13.1. Pedidos

Al iniciar sesión se muestra la sección *Pedidos*. Esta página es la principal de la trastienda y en ella se muestran todos los pedidos que han realizado los usuarios. Los pedidos están ordenados, por defecto, por la fecha en la que se realizó. Para modificar el orden de los registros

se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellorando los campos del formulario situado encima de la tabla y pulsando el botón “Buscar”. En la parte inferior de la tabla se sitúa la barra de paginación, que permite cambiar de página y seleccionar el número de registros a mostrar por página.

Id. Pedido	Fecha Entrada del Pedido	Fecha de Entrega aprox	Estado	Id. Cliente (email)
6341	28-09-2020	28-09-2020	Pendiente envio	1601318419856@gmail.com
46975	27-09-2020	27-09-2020	Finalizado	1601136589233@gmail.com
47686	27-09-2020	27-09-2020	Pendiente envio	1601136589233@gmail.com
3528	22-09-2020	22-09-2020	Pendiente envio	1601136589233@gmail.com

Figura B.51: Sección Pedidos.

En la parte superior derecha de la tabla se encuentra la barra de herramientas.



Figura B.52: Barra de herramientas.

El primer icono de la izquierda permite recargar los datos de la tabla. El icono central limpia los valores de los filtros cargando de nuevo todos los pedidos. El icono de la derecha permite exportar la tabla a un documento Excel.

Cada registro tiene dos iconos:



Figura B.53: Iconos del registro.

El icono de la derecha permite generar la factura para aquellos pedidos pagados a contra reembolso. El resto de facturas se generan automáticamente cuando el usuario finaliza el pago.

El icono de la izquierda permite mostrar la ficha del pedido.

Código	Categoría	Marca	Modelo	Cantidad
12345672	Lavadoras	Samsung	WW70J33KW A+++	1

Figura B.54: Ficha del pedido.

En la ficha del pedido se muestra el listado de productos que el usuario ha comprado, la fecha de entrega, el correo del usuario, si está cobrado y el estado del pedido. El administrador puede marcar el pedido como cobrado o no cobrado y actualizar el estado (preparando, entregado o finalizado). Para guardar los datos se pulsa el botón “Actualizar pedido”.

B.13.2. Facturas

En la sección *Facturas* se muestra las facturas generadas en el sistema. Las facturas están ordenadas, por defecto, por número de factura. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos del formulario situado encima de la tabla y pulsando el botón “Buscar”. En la parte inferior de la tabla se sitúa la barra de paginación, que permite cambiar de página y seleccionar el número de registros a mostrar por página.

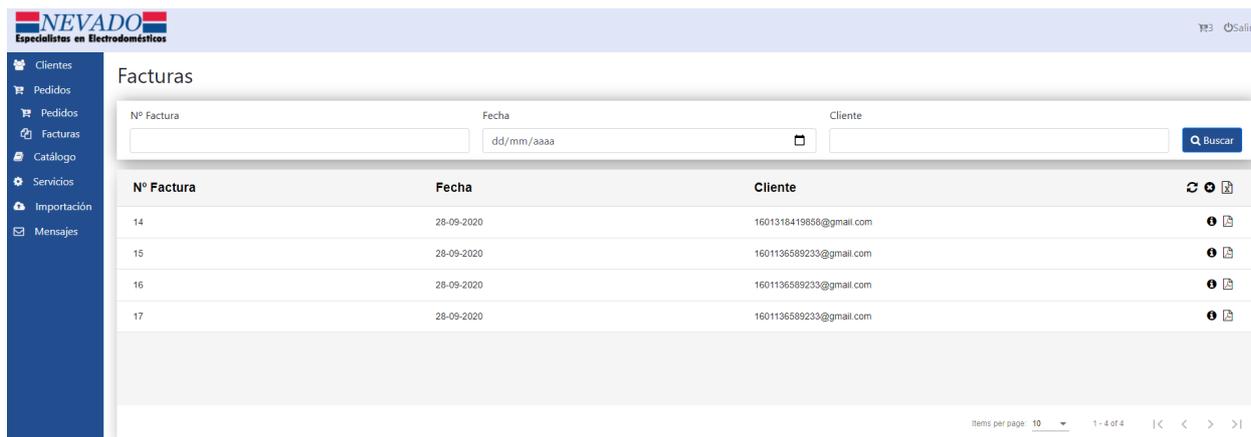


Figura B.55: Sección Facturas.

En la parte superior derecha de la tabla se encuentra la barra de herramientas.



Figura B.56: Barra de herramientas de Facturas.

El primer icono de la izquierda permite recargar los datos de la tabla. El icono central limpia los valores de los filtros, cargando de nuevo todas las facturas. El icono de la derecha permite exportar la tabla a un documento Excel.

Cada registro tiene dos iconos:



Figura B.57: Iconos de una factura.

El icono de la derecha permite descargar el PDF de la factura. Véase la figura 4.5.

El icono de la izquierda permite mostrar la ficha de la factura.

The screenshot shows a receipt card titled 'Factura 14'. At the top, there are three fields: 'Factura 14', 'Fecha 28/09/2020', and 'Cliente 1601318419858@gmail.com'. Below this is a table with the following data:

Código	Artículo	Marca	Modelo	Cantidad	Precio
12345671	Lavadoras	Bosch	WUQ24468ES A+++ 2	202.4	
Total					404.8 €

At the bottom of the table, there is a pagination bar showing 'Items per page: 5' and '1 - 1 of 1' with navigation arrows. An 'Ok' button is located at the bottom right of the card.

Figura B.58: Ficha de la factura.

En la ficha de la factura se muestra el listado de productos que el usuario ha comprado, la fecha de facturación, el correo del usuario y el número de la factura. El administrador no puede realizar ninguna modificación sobre la factura.

B.13.3. Clientes

En la sección *Clientes* se muestra los clientes registrados en la tienda. Los clientes están ordenados, por defecto, por nombre. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos del formulario situado encima de la tabla y pulsando el botón “Buscar”. En la parte inferior de la tabla se sitúa la barra de paginación, que permite cambiar de página y seleccionar el número de registros a mostrar por página.

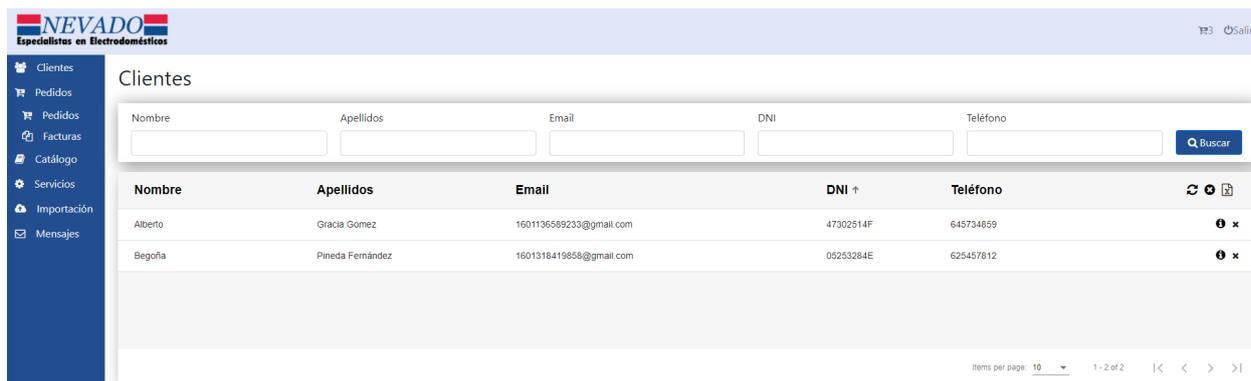


Figura B.59: Sección Clientes.

En la parte superior derecha de la tabla se encuentra la barra de herramientas.



Figura B.60: Barra de herramientas de Clientes.

El primer icono de la izquierda permite recargar los datos de la tabla. El icono central limpia los valores de los filtros, cargando de nuevo todos los Clientes. El icono de la derecha permite exportar la tabla a un documento Excel.

Cada registro tiene dos iconos:



Figura B.61: Iconos de un cliente.

El icono de la derecha permite eliminar un cliente. El cliente puede pedir que su información personal sea borrada del sistema debido a la Ley de Protección de Datos. Antes de borrar el cliente, el sistema muestra una ventana de confirmación.

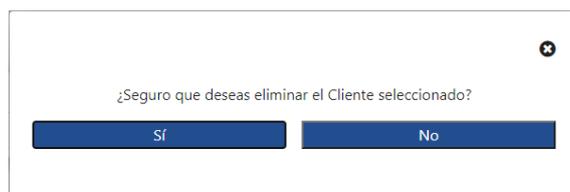


Figura B.62: Ventana de confirmación para eliminar un cliente.

El icono de la izquierda permite mostrar la ficha del cliente.

The image shows a screenshot of a web application interface for a customer profile. At the top, there is a dark blue header bar with the text 'Ficha del cliente' and a small close icon on the right. Below the header, the profile information is displayed in a light gray, card-like layout with rounded corners and thin borders. The data is organized as follows:

- Nombre:** Alberto
- Apellidos:** Gracia Gomez
- DNI:** 47302514F
- Fecha Nacimiento:** 31/08/1988
- Teléfono:** 645734859
- Email:** 1601136589233@gmail.com
- Dirección Envío:** (The field is empty)
- Dirección Factura:** Méndez Alvaro, 4, 1-B, 28904 Getafe, Madrid
- Nº de Pedidos:** 2

At the bottom of the card, there is a solid dark blue horizontal bar.

Figura B.63: Ficha del cliente.

En la ficha del cliente se muestra los datos personales, incluyendo dirección de facturación, dirección de envío y el número de pedidos que ha realizado en la tienda. El administrador no puede realizar ninguna modificación sobre los datos del cliente.

B.13.4. Productos

En la sección *Productos* se muestra los productos almacenados en la tienda. Los productos están ordenados, por defecto, por código de producto. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos del formulario situado encima de la tabla y pulsando el botón “Buscar”. En la parte inferior de la tabla se sitúa la barra de paginación, que permite cambiar de página y seleccionar el número de registros a mostrar por página.

Código ↓	Categoría	Marca	Modelo	PVP	Stock	
12345679	Lavadoras	Balay	3TS986BT A+++	350	2	
12345672	Lavadoras	Samsung	WW70J33KW A+++	360	4	
12345671	Lavadoras	Bosch	WUQ24468ES A+++	253	1	

Figura B.64: Sección Productos.

En la parte superior derecha de la tabla se encuentra la barra de herramientas.



Figura B.65: Barra de herramientas de productos.

El primer icono, empezando por la izquierda, permite recargar los datos de la tabla. El segundo icono permite añadir un nuevo producto. El tercer icono limpia los valores de los filtros, cargando de nuevo todos los productos. El último icono permite exportar la tabla a un documento Excel.

Cada registro tiene dos iconos:



Figura B.66: Iconos de un Producto.

El icono de la derecha permite eliminar un producto. Antes de borrar el producto, el sistema muestra una ventana de confirmación. Véase la figura B.62

El icono de la izquierda permite mostrar la ficha del producto.

B.13.5. Marcas y Categorías

En la sección *Marcas y Categorías* se muestran las marcas y categorías de los productos almacenados en la tienda, cada una en una tabla diferente. Tanto las marcas como las categorías están ordenados, por defecto, por nombre. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos de los formularios situados encima de las tablas y pulsando el botón “Buscar” de cada tabla. En la parte inferior de las tablas se sitúan la barras de paginación, que permiten cambiar de página y seleccionar el número de registros a mostrar por página.

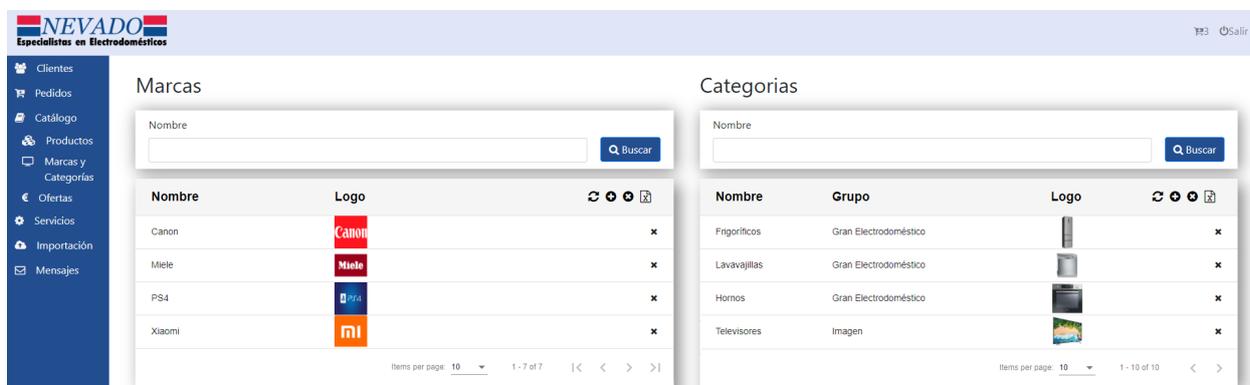


Figura B.68: Sección Marcas y Categorías.

En la parte superior derecha de la tablas se encuentran la barras de herramientas, iguales para ambas tablas.



Figura B.69: Barra de herramientas de marcas y categorías.

El primer icono, empezando por la izquierda, permite recargar los datos de las tablas. El segundo icono permite añadir un nueva marca o una nueva categoría. El tercer icono limpia los valores de los filtros, cargando de nuevo todas las marcas o todas las categorías. El último icono permite exportar la tablas a los documentos Excel correspondientes.

Para crear una marca o categoría, el administrador debe rellenar los campos de los formularios y pulsar sobre el recuadro azul que permite seleccionar y subir una foto al servidor. Para finalizar debe pulsar el botón “Guardar”.

Figura B.70: Añadir una marca y una categoría.

Cada registro tiene un único icono:



Figura B.71: Icono de una marca y una categoría.

El icono permite eliminar la marca o la categoría seleccionada. Antes de borrar el registro seleccionado, el sistema muestra una ventana de confirmación. Véase la figura B.62

B.13.6. Ofertas

En la sección *Ofertas* se muestran las ofertas disponibles en la tienda. Las ofertas están ordenadas, por defecto, por identificador de oferta. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos del formulario situado encima de la tabla y pulsando el botón “Buscar”. En la parte inferior de la tabla se sitúa la barra de paginación, que permite cambiar de página y seleccionar el número de registros a mostrar por página.

Id. Oferta	% de descuento	Fecha de Inicio	Fecha Fin	
123456	20 %	01-09-2020	31-10-2020	✖

Figura B.72: Sección Ofertas.

En la parte superior derecha de la tabla se encuentra la barra de herramientas.



Figura B.73: Barra de herramientas de ofertas.

El primer icono, empezando por la izquierda, permite recargar los datos de la tabla. El segundo icono permite crear una nueva oferta. El tercer icono limpia los valores de los filtros, cargando de nuevo todas las ofertas. El último icono permite exportar la tabla a un documento Excel.

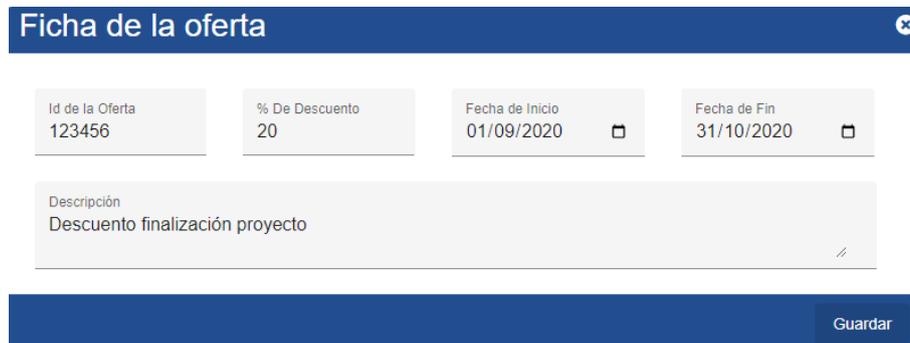
Cada registro tiene dos iconos:



Figura B.74: Iconos de una oferta.

El icono de la derecha permite eliminar una oferta. Antes de borrar la oferta, el sistema muestra una ventana de confirmación. Véase la figura B.62

El icono de la izquierda permite mostrar la ficha de la oferta.



Ficha de la oferta

Id de la Oferta 123456	% De Descuento 20	Fecha de Inicio 01/09/2020	Fecha de Fin 31/10/2020
---------------------------	----------------------	-------------------------------	----------------------------

Descripción
Descuento finalización proyecto

Guardar

Figura B.75: Ficha de la oferta.

En la ficha de la oferta se muestra el identificador de la oferta, el porcentaje de descuento, la fecha de inicio de la oferta, la fecha de fin y una pequeña descripción, que explica en que consiste la oferta. El administrador puede modificar todos los campos. Para guardar los cambios debe pulsar el botón “Guardar”. Esta ventana se utiliza para añadir nuevas ofertas y para editarlas.

B.13.7. Servicios

En la sección *Servicios* se muestran los métodos de pagos disponibles en la tienda *online* y las empresas de transporte que realizan los envíos, cada una en una tabla diferente. Tanto los métodos de pago como las empresas están ordenados, por defecto, por nombre. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos de los formularios situados encima de las tablas y pulsando el botón “Buscar” de cada tabla. En la parte inferior de las tablas se sitúan la barras de paginación, que permiten cambiar de página y seleccionar el número de registros a mostrar por página.

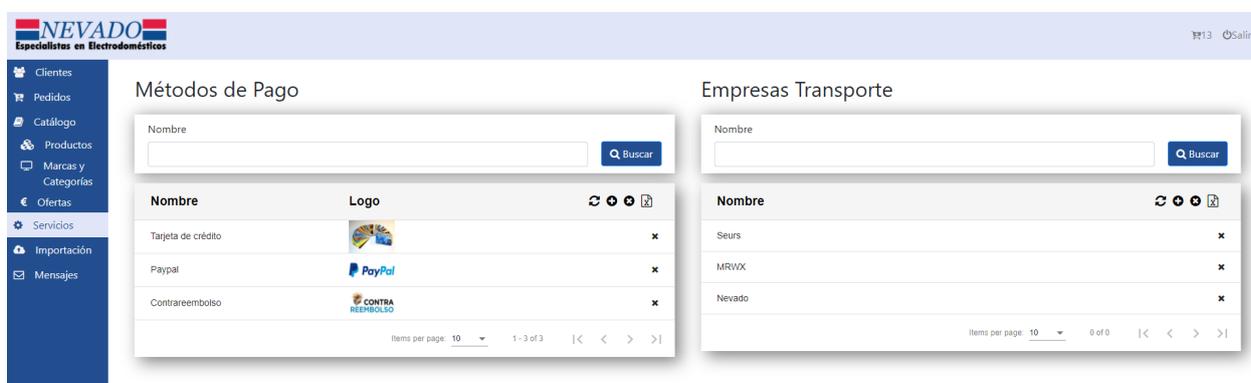


Figura B.76: Sección Métodos de pago y empresas de transportes.

En la parte superior derecha de las tablas se encuentran las barras de herramientas, iguales para ambas tablas.



Figura B.77: Barra de herramientas de métodos de pago y Empresas de transportes.

El primer icono, empezando por la izquierda, permite recargar los datos de las tablas. El segundo icono permite añadir un nuevo método de pago o una nueva empresa de transporte. El tercer icono limpia los valores de los filtros, cargando de nuevo todos los métodos de pago o todas las empresas de transportes. El último icono permite exportar las tablas a los documentos Excel correspondientes.

Para crear un método de pago el administrador debe rellenar el campo del los formulario y pulsar sobre el recuadro azul que permite seleccionar y subir una foto al servidor. Para finalizar

debe pulsar el botón “Guardar”. Para crear una nueva empresa de transporte debe rellenar el campo del formulario y pulsar el botón “Guardar”.

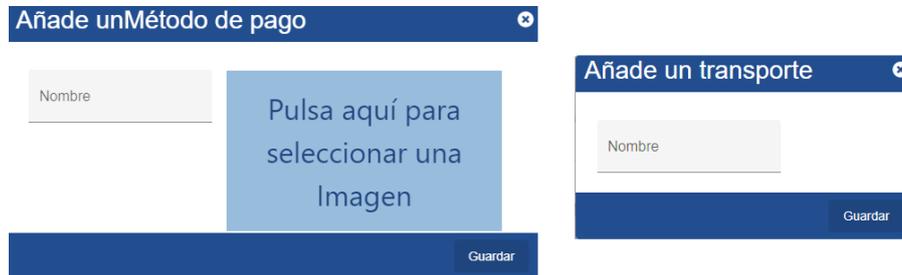


Figura B.78: Añadir un método de pago y una empresa de transporte.

Cada registro tiene un único icono:



Figura B.79: Icono de un método de pago y una empresa de transporte.

El icono permite eliminar la marca o la categoría seleccionada. Antes de borrar el registro seleccionado, el sistema muestra una ventana de confirmación. Véase la figura B.62

B.13.8. Mensajes

En la sección *Mensajes* se muestran los mensajes enviados por los usuarios al administrador. Los mensajes están ordenados, por defecto, por nombre. Para modificar el orden de los registros se pulsa sobre la cabecera de cada columna. El administrador puede filtrar los resultados rellenando los campos del formulario situado encima de la tabla y pulsando el botón “Buscar”. En la parte inferior de la tabla se sitúa la barra de paginación, que permite cambiar de página y seleccionar el número de registros a mostrar por página.



Figura B.80: Sección Mensajes.

En la parte superior derecha de la tabla se encuentra la barra de herramientas.



Figura B.81: Barra de herramientas de mensajes.

El primer icono, empezando por la izquierda, permite recargar los datos de la tabla. El icono central limpia los valores de los filtros, cargando de nuevo todos los mensajes. El último icono permite exportar la tabla a un documento Excel.

Cada registro tiene dos iconos:



Figura B.82: Iconos de un mensaje.

El icono de la derecha permite eliminar un mensaje. Antes de borrar el mensaje, el sistema muestra una ventana de confirmación. Véase la figura B.62

El icono de la izquierda permite mostrar el mensaje.

Figura B.83: Detalle del mensaje.

En el detalle del mensaje se muestra el nombre y correo del usuario que envió el mensaje, la fecha en que se escribió el mensaje, si está leído o no, el mensaje y un campo de respuesta. El administrador puede responder al usuario rellorando el campo respuesta y pulsando sobre el botón “Enviar”. La respuesta se enviará al correo electrónico del usuario. Además, puede marcar el mensaje como leído.

B.13.9. Importación

En la sección *Importación* el administrador puede cargar datos en el sistema de manera masiva. En la parte superior de la página se sitúa un formulario con dos campos y tres botones.

Figura B.84: Sección Importación.

El campo de la izquierda permite elegir el tipo de dato que se desea subir (categorías, marcas

o productos). En el segundo campo se adjunta el fichero .zip a importar. Para subir el archivo al servidor, el administrador debe pulsar el botón “Buscar” y seleccionar el archivo que desea importar. Para finalizar la importación debe pulsar el botón “Importar”. Cuando se realiza la importación se muestra por pantalla los registros totales que se han importado y los que han dado error. Además, se indica el error de cada registro.

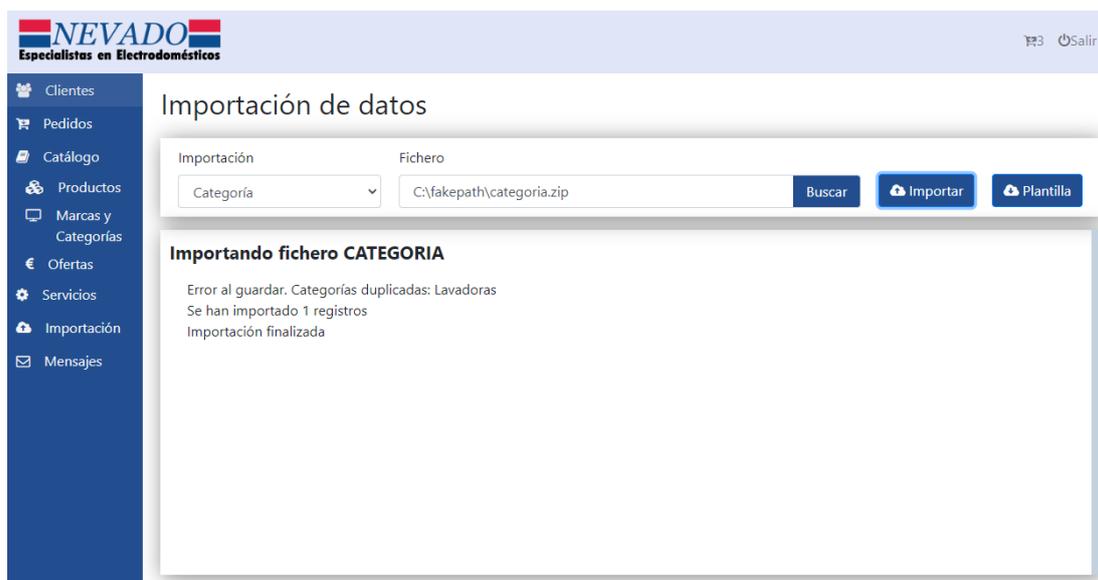


Figura B.85: Log de importación.

El botón “Plantilla” descarga la plantilla del documento Excel a rellenar del tipo seleccionado.

Bibliografía

- [1] Biblioteca Angular Editor.
<https://www.npmjs.com/package/@kolkov/angular-editor>.
- [2] Biblioteca File Saver.
<https://www.npmjs.com/package/angular-file-saver>.
- [3] Biblioteca Node Adm-zip.
<https://www.npmjs.com/package/adm-zip>.
- [4] Biblioteca Node Excel4Node.
<https://www.npmjs.com/package/excel4node>.
- [5] Biblioteca Node Multer.
<https://www.npmjs.com/package/multer>.
- [6] Biblioteca Node.js HTML-PDF.
<https://www.npmjs.com/package/html-pdf>.
- [7] Biblioteca Node.js SheetJS-XLSX.
<https://www.npmjs.com/package/sheetjs>.
- [8] Página CSS3.
http://www.w3schools.com/css/css3_intro.asp/.
- [9] Página de Angular.
<https://angular.io>.
- [10] Página de Angular Cli.
<https://cli.angular.io>.

- [11] Página de Angular Material.
<https://material.angular.io>.
- [12] Página de Bcryptjs.
<https://www.npmjs.com/package/bcryptjs>.
- [13] Página de Bootstrap.
<https://getbootstrap.com/>.
- [14] Página de Expressjs.
<https://expressjs.com/es/>.
- [15] Página de Git.
<https://git-scm.com>.
- [16] Página de Mongo Compass.
<https://www.mongodb.com/products/compass>.
- [17] Página de MongoDB.
<https://www.mongodb.com>.
- [18] Página de Mongoose.
<https://mongoosejs.com>.
- [19] Página de Node.js.
<https://nodejs.org/es/>.
- [20] Página de Nodemailer.
<https://nodemailer.com/>.
- [21] Página de Nodemon.
<https://nodemon.io>.
- [22] Página de npm.
<https://www.npmjs.com/>.
- [23] Página de Passportjs.
www.passportjs.org.

[24] Página de Paypal Developer.

<https://developer.paypal.com>.

[25] Página de Typescript.

<https://www.typescriptlang.org/>.

[26] Página HTML.

https://www.w3schools.com/html/html5_intro.asp.