Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES Y MULTIMEDIA

TRABAJO DE FIN DE GRADO

PLATAFORMA DE COMERCIO ONLINE MAD SERVICES

Autor: Alejandro Fernández Pérez

Tutor: Dr. Jesús María González Barahona

Curso académico 2022/2023

©2023 Alejandro Fernández Pérez
Algunos derechos reservados

Este documento se distribuye bajo la licencia
“Atribución-CompartirIgual 4.0 Internacional”
de Creative Commons, disponible en
https://creativecommons.org/licenses/by-sa/4.0/deed.es

Trabajo de Fin de Grado

PLATAFORMA DE COMERCIO ONLINE MAD SERVICES

Autor: Alejandro Fernández Pérez

Tutor: Dr. Jesús María González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2023, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2023

Dedicado a

mi familia, por apoyarme incluso en los momentos de mayor dificultad, por darme fuerza para que no me rindiera durante el proceso de la carrera de fondo que es la Licenciatura. A mi abuela, por apoyarme y por ser una luchadora innata a sus 93 años. A mis amigos, ya que sin su apoyo, risas y sentido del humor hubiera sido más difícil y más aburrido el proceso de obtención de la Licenciatura. Y a mi tutor, Jesús, por su paciencia conmigo.

Agradecimientos

Y ya llegó el final de la Licenciatura. Parecía que estaba muy lejos, pero como todo, está a la vuelta de la esquina. Como dijo el empresario, diseñador industrial, magnate empresarial, propietario de medios e inversor estadounidense, Steve Jobs: *'el tiempo corre y es limitado, así que no lo malgastes viviendo la vida de otro. Vive tu propia vida. Todo lo demás es secundario'*.

A partir de ahora, me depararán nuevos retos y objetivos. Porque en eso consiste la vida, en marcarte objetivos o retos a batir, e ir aprendiendo, adquiriendo conocimiento y disfrutando por el camino. Los seres humanos debemos ser personas realizadas, si no estaríamos vacíos de contenido y no habría ninguna diferencia con un ser irracional.

Hasta ahora, puedo estar satisfecho con los logros que he ido consiguiendo, gracias a mi constancia, responsabilidad y esfuerzo diario. Pero también, gracias a un apoyo emocional y racional fuerte por parte de mi familia y amigos, sin ellos, todo hubiera sido mucho más difícil y cuesta arriba, ya que la soledad y la falta de apoyo emocional genera mucha más frustración y una mayor probabilidad de fracaso en la consecución de objetivos.

Y no olvidarme de mi Escuela Infantil, Colegio, Instituto y Universidad, que gracias a ellos, he podido desarrollarme profesionalmente, académicamente, emocionalmente y psicológicamente. Donde he conocido a interesantes e importantes personas para mi vida y, que gracias a todo ello, soy quien soy.

Muchas gracias a todos por todo.

Atentamente

Resumen

El Trabajo de Fin de Grado consiste en la programación de una aplicación web de un *Marketplace* de tipo negocios a clientes, que es una Plataforma de Comercio o Centro Comercial online desarrollado en un entorno de tiempo de ejecución de 2D (Dos-Dimensiones) en su primera versión, que sirve para conectar, unificar y dar a conocer las empresas registradas, pudiendo incorporar los productos y/o servicios que quieran. También funciona como una empresa más de comercio electrónico, vendiendo sus productos propios.

Este proyecto se irá desarrollando para tenerlo en 3D (Tres-Dimensiones) y RV (Realidad Virtual) en futuras versiones.

Índice general

Agradecimientos **III**

Resumen **V**

1. Introducción **1**

1.1. Objetivo general 3

1.2. Objetivos específicos 3

1.3. Entorno de trabajo 4

1.4. Estructura de la memoria 4

2. Estado del arte, de la técnica y sistemas relacionados **7**

2.1. Aplicaciones Web 7

2.2. Node JS 12

2.3. Express JS 14

2.3.1. Modelo 15

2.3.2. Vista 18

2.3.3. Controlador 20

2.4. Dependencias de Express JS 21

2.4.1. Axios 21

2.4.2. Bcrypt 22

2.4.3. Body-parser 23

2.4.4. Card-validator 23

2.4.5. Cookie-parser 24

2.4.6. CORS 24

2.4.7. Country-list y Country-list-spanish 25

2.4.8. Dotenv 26
2.4.9. Express-session 26
2.4.10. Fs 27
2.4.11. Helmet 27
2.4.12. Http-errors 27
2.4.13. Luxon 28
2.4.14. Method-override 28
2.4.15. Multer 29
2.4.16. Nif-dni-nie-cif-validation 29
2.4.17. Node-notifier 29
2.4.18. Path 30
2.4.19. Postcode-validator 30
2.4.20. Serve-favicon 31
2.4.21. Sharp 31
2.4.22. Util 31
2.4.23. Validator 32
2.4.24. Nodemon 32
2.4.25. Debug 33
2.5. Otras tecnologías 33
2.5.1. Overleaf 34
2.5.2. Visual Studio Code 35
2.5.3. Thunder Client 37
2.5.4. Git y GitHub 38
2.5.5. Adobe Photoshop 39
2.5.6. Adobe Premiere y Wondershare Filmora 9 40

3. Proceso de desarrollo **41**
3.1. Fase 0: El servidor web o API 42
3.2. Fase 1: Las rutas del servidor web o API 43
3.3. Fase 2: El controlador de la aplicación web 45
3.4. Fase 3: El modelo o manejo de datos de la aplicación web 47

3.5. Fase 4: Configuración de la interfaz de usuario hacia el servidor web o API . . 51
3.6. Fase 5: Vistas Estáticas y su diseño y apariencia 52
3.7. Fase 6: Archivos externos 53
3.8. Fase 7: Despliegue del prototipo en la nube 55

4. Resultados **57**
4.1. Funcionamiento de la interfaz final 57

5. Conclusiones **65**
5.1. Desarrollo temporal 65
5.2. Consecución de objetivos 68
5.3. Lecciones aprendidas 69
5.4. Asignaturas relacionadas 70
5.5. Trabajos futuros 70
5.6. Intrahistoria de la idea 74

Bibliografía **77**

A. Manual de usuario **93**
A.0.1. Dilema práctico 123

Índice de figuras

1.1. Logo de MAD Services 1
1.2. Esquema gráfico de un Marketplace de tipo B2C 2

2.1. Imagen de la Arquitectura TCP/IP 7
2.2. Modelo de comunicaciones Cliente-Servidor de una aplicación web 9
2.3. Tipos de aplicaciones web según la programación y diseño de la interfaz 10
2.4. Tipos de aplicaciones web según las categorías a desarrollar en la interfaz . . . 11
2.5. Tipos de *Marketplace* existentes 12
2.6. Logo de *Node JS* 13
2.7. Características del entorno de *Node JS* 14
2.8. Logo de *Express JS* 14
2.9. Logo de *MySQL* 16
2.10. Logo de *phpMyAdmin* 17
2.11. Logo de *Apache* 17
2.12. Logo de *PHP* 18
2.13. Logo de *PUG* 19
2.14. Logo de *CSS* 19
2.15. Logo de *JavaScript* 21
2.16. Funcionamiento de la tecnología de *Axios* 21
2.17. Tipo de cifrado con la tecnología de *Bcrypt* 22
2.18. Tipo de autenticado con la tecnología de *Bcrypt* 23
2.19. Funcionamiento de la tecnología de *Body-parser* 23
2.20. Funcionamiento de la tecnología de *Card-validator* 24
2.21. Funcionamiento de la tecnología de *Cookie-parser* 24

2.22. Funcionamiento de la tecnología de *CORS* 25
2.23. Listado de países con las tecnologías de *Country-list* y *Country-list-spanish* . . 25
2.24. Funcionamiento de la tecnología de *Dotenv* 26
2.25. Funcionamiento de la tecnología de *Express-session* 26
2.26. Funcionamiento de la tecnología de *Fs* 27
2.27. Funcionamiento de la tecnología de *Helmet* 27
2.28. Tipo de error con la tecnología de *Http-errors* 28
2.29. Funcionamiento de la tecnología de *Luxon* 28
2.30. Funcionamiento de la tecnología de *Method-override* 29
2.31. Funcionamiento de la tecnología de *Multer* 29
2.32. Funcionamiento de la tecnología de *Nif-dni-nie-cif-validation* 29
2.33. Funcionamiento de la tecnología de *Node-notifier* 30
2.34. Funcionamiento de la tecnología de *Path* 30
2.35. Funcionamiento de la tecnología de *Postcode-validator* 30
2.36. Funcionamiento de la tecnología de *Serve-favicon* 31
2.37. Tipos de tamaños generados por la tecnología de *Sharp* 31
2.38. Funcionamiento de la tecnología de *Util* 32
2.39. Funcionamiento de la tecnología de *Validator* 32
2.40. Logo de *Nodemon* 33
2.41. Logo de *Debug* 33
2.42. Logo de *Overleaf* 34
2.43. Logo de *Visual Studio Code* 35
2.44. Encuesta sobre los tipos de código fuente que hay 36
2.45. Logo de *Thunder Client* 37
2.46. Interfaz de *Thunder Client* como simulador de llamadas a la API 37
2.47. Logo de *Git* 38
2.48. Tipos de servicios en la nube de *Git* 39
2.49. Logo de *Adobe Photoshop* 40
2.50. Logos de *Adobe Premiere* y *Wondershare Filmora 9* 40

3.1. Esquema del prototipo del proyecto 41

3.2. Mapeo a un objeto de aplicación agrupando en grupos 44
3.3. Mapeo a un objeto de rutas hacia recursos específicos 44
3.4. Tipos de alertas sonoras en el prototipo 45
3.5. Funcionamiento de la tecnología de *Multer* 47
3.6. Instrucciones 1 de LoadModule de *httpd.conf* de *Apache24* 49
3.7. Instrucciones 2 de LoadModule de *httpd.conf* de *Apache24* 49
3.8. Interfaz gráfica de la base de datos del proyecto en *phpMyAdmin* 51
3.9. Configuración de la interfaz de usuario hacia la API 51
3.10. Base de todas las vistas estáticas del prototipo 52
3.11. Función generar ID aleatoriamente y función consultar ID aleatorio en clientes 53
3.12. Archivos externos del tipo 3 54
3.13. Logo de un Servidor *Cloud* 55
3.14. Archivo BAT compilador y archivo VBS intérprete 56

4.1. Pantalla inicial del modo usuario sin registro de la interfaz final 59
4.2. Pantalla inicial del modo usuario cliente de la interfaz final 60
4.3. Pantalla inicial del modo usuario empresa de la interfaz final 61
4.4. Pantalla inicial del modo usuario miembro de la interfaz final 63

5.1. Logo de *A-Frame* 70
5.2. Tipos de Sistemas de *A-Frame* 72
5.3. Tipos de Entidades de *A-Frame* 72
5.4. Tipos de Componentes de *A-Frame* 72
5.5. Código del servidor web con *Express JS* 73
5.6. Código de la interfaz de usuario con *A-Frame* 73
5.7. Captura de imagen del vídeo de la versión futura de MAD Services 73

Capítulo 1

Introducción

Este proyecto consiste en la realización de una aplicación web[1] de un *Marketplace* de tipo *B2C*[2] (Business to Customers, de sus siglas en inglés y Negocios a Clientes, traducido al español), que es una Plataforma de Comercio online llamada *MAD Services* que hace de intermediaria entre los clientes y las empresas registradas, donde no hay límites en la cantidad de artículos, productos y/o servicios a comprar y vender y donde se venden sus productos propios como un comercio electrónico más. Es decir, se podría definir como un Centro Comercial online.

Figura 1.1: Logo de MAD Services

La aplicación está compuesta por tres componentes[3]: el servidor web, el servidor de bases de datos y el cliente o navegador web. El primero es el componente de tiempo de ejecución principal donde se ejecuta la aplicación. El segundo es el componente encargado de proporcionar datos a la aplicación. Y el tercero es el componente que maneja la lógica de la interfaz de usuario, en cuyo caso, es de tipo *SWA*[4] (Simple Web APP, de sus siglas en inglés y Web APP Simple, traducido al español) porque está diseñada y adaptada sin necesidad de utilizar espacio de almacenamiento en disco de ningún dispositivo electrónico, simplemente ejecutándose en un navegador web. Además, porque se puede actualizar cuando se requiera, su tiempo, su rendimiento y su coste son bajos, su acceso es limitado a las funciones del dispositivo, las experiencias de usuario son diferentes en función del navegador utilizado y sólo funciona con conexión a Internet.

El tipo de aplicación web de *Marketplace* es una idea a desarrollar muy llamativa con respecto a los otros tipos porque no tiene tanta variedad en comparación con los otros, su funcionalidad puede ser múltiple, su implementación puede ser diversa, genera oportunidades de negocio que apoyan la economía de todo tipo de empresas, contribuye a una mayor visibilidad de las marcas, ayuda a posicionar a *Start-Ups* o empresas emergentes y/o a PYMES (Pequeñas y Medianas Empresas, de sus siglas en español), genera fidelización con los clientes, no entiende de fronteras, potencia las ventas cruzadas o las ventas adicionales basadas en ventas primarias del cliente[5], hace ahorrar a las empresas en el posicionamiento, su gestión de pagos suele ser más segura y hace que la inversión de las empresas sea menor al tener la posibilidad de prescindir de su propio comercio electrónico.

La diferencia de esta aplicación web de *Marketplace* con respecto a otras del mismo tipo está en dar protagonismo a cada una de las empresas registradas, destacando su logo y marca empresarial, dejando que cada una de ellas desarrolle la interfaz que muestra al cliente, donde puede introducir los productos y/o servicios que quiera, promocionar su marca a través de redes sociales, introducir uno o varios contactos para establecer comunicación con el cliente, incluso añadir el enlace de acceso a su propia página web externa. Y lo más relevante, poder filtrar la búsqueda en la pantalla donde encontrar todas esas empresas registradas, para que el cliente encuentre la que quiera, según su tipo, dividido por colores y, según su nombre.

En conclusión, la definición de la aplicación como un tipo *B2C*, significa que establece la posibilidad de que las empresas se registren de forma oficial, de acuerdo a su código CIF/NIF, para poder vender sus productos y/o servicios a los clientes registrados, quienes los compran. Es decir, se establece la relación de empresas a clientes únicamente.

Figura 1.2: Esquema gráfico de un Marketplace de tipo B2C de compra-venta entre Empresa y Cliente, sacado de [6]

1.1. Objetivo general

El objetivo general de este Proyecto de Fin de Carrera es la realización de una aplicación web de *Marketplace* de tipo *B2C* para intermediar entre empresas y clientes y vender productos propios, desarrollada en el entorno de tiempo de ejecución en tiempo real de *JavaScript* llamado *Node JS*.

1.2. Objetivos específicos

Otros objetivos no menos relevantes, pero sí más detallados, son:

En primer lugar, el uso del *framework* de *Express JS* para generar la API (Interfaz de Programación de la Aplicación) del proyecto de la aplicación web en el entorno de *Node JS* y que la intermediación y las ventas de productos descritas sean posibles.

En segundo lugar, el uso de vistas estáticas o motores de plantillas *PUG* para crear la interfaz de usuario y que las empresas y los clientes puedan ejecutarla en el navegador web.

En tercer lugar, el diseño de la interfaz de usuario, a través de la tecnología de *CSS*, para que las empresas y los clientes puedan visualizarla con estilos, formatos y una apariencia detallada.

En cuarto lugar, el almacenamiento de datos e información en la base de datos, a través de la tecnología de *MySQL*, para subir productos propios de la aplicación y para que las empresas puedan subir sus productos y/o servicios a vender.

En quinto lugar, la obtención de datos e información almacenados en la base de datos, a través también de la tecnología de *MySQL*, para que los clientes puedan obtener los productos y/o servicios a comprar.

Y por último, la organización del código de programación de la aplicación web por medio de directorios generales y específicos para que su entendimiento sea coherente, rápido y sencillo.

1.3. Entorno de trabajo

El proyecto se ha realizado en un portátil Lenovo, con Sistema Operativo (S.O) *Windows 11 Pro* cuya versión es la *22H2*, procesador *Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz*, memoria RAM de 8 GB (Gigabytes) y tipo de Sistema Operativo (S.O) de 64 bits, cuyo procesador está basado en x64.

Respecto al navegador web, se ha utilizado *Brave*[7], que es un navegador de código abierto basado en *Chromium* y desarrollado por *Brave Software* en 2016. Aunque también se ha utilizado *Mozilla Firefox*[8] para comprobar su funcionamiento en otro navegador diferente y basado en otra tecnología.

1.4. Estructura de la memoria

Para acabar con este primer punto, se definirá la estructura de la memoria de este proyecto.

Como ya se ha podido observar, hasta ahora se ha podido comprobar su **Portada**, su **Índice**, sus **Agradecimientos**, su **Dedicatoria**, su **Resumen** y su **Introducción**. Esto se ha hecho para presentarlo, hacerlo más humano, con alma y para ordenar las ideas, conectar y enganchar al lector.

Después viene el desarrollo de todas las tecnologías usadas y la explicación y definición de una aplicación web y su funcionamiento, que es la base de este proyecto. Este punto se conoce como: **Estado del arte, de la técnica y sistemas relacionados**.

A continuación, está su **Proceso de desarrollo** dividido en ocho fases. La fase 0 explica el servidor web o API en general. La fase 1 explica las rutas que hacen funcionar al servidor web. La fase 2 explica el controlador de la aplicación web dependiente de la API, donde está la lógica de la programación de ésta última. La fase 3 explica el modelo o manejo de datos de la aplicación web, donde se conecta con la base de datos. La fase 4 explica la configuración de la interfaz de usuario y su apariencia hacia el servidor web o API. La fase 5 explica el funcionamiento de las vistas estáticas o motores de plantillas y el diseño, estilo y apariencia de ellas. La

fase 6 explica el funcionamiento de los archivos externos que hay. Y la fase 7 es la explicación del despliegue del prototipo del proyecto en la nube y sus problemas y alternativas.

Luego se sigue con los **Resultados** que consisten en el funcionamiento de la interfaz final y, con las **Conclusiones**, describiendo el desarrollo temporal, la consecución de objetivos, las lecciones aprendidas, las asignaturas relacionadas, el desarrollo de trabajos futuros y una intra-historia del surgimiento de la idea.

Se termina la memoria con una **Bibliografía**, donde se referencian las fuentes de información de donde se han sacado los contenidos para desarrollar la memoria. Aparte de realizar como anexo, un **Manual de usuario** respecto del prototipo realizado, para conocer el funcionamiento práctico de la aplicación web, describiendo en él un dilema práctico.

Capítulo 2

Estado del arte, de la técnica y sistemas relacionados

El Estado del arte, de la técnica y sistemas relacionados consiste en explicar qué es una *aplicación web*[9] y su funcionamiento, base del prototipo del proyecto y, en explicar todas las tecnologías que se usan en ella.

2.1. Aplicaciones Web

Una aplicación web es un software compuesto por dos componentes: un servidor web y un cliente ejecutado en un navegador web. Su funcionamiento consiste en la realización y envío de peticiones solicitud y respuesta bajo el protocolo de aplicación *HTTP*[10], que sigue la Arquitectura TCP/IP de la imagen siguiente:

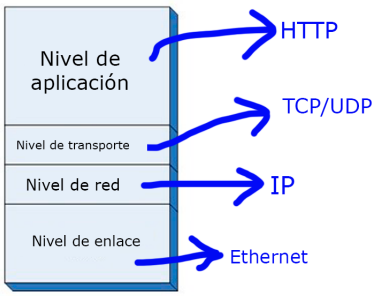


Figura 2.1: Imagen de la Arquitectura TCP/IP diseñada por el autor en Adobe Photoshop

Sus características[11] principales son las siguientes: no necesita ser instalada en ningún

dispositivo electrónico, ya que los datos o archivos utilizados se almacenan en una red o en la nube por parte de un servidor web, cuya función es el alojamiento y envío de información cuando sea requerido. No ocupa espacio de almacenamiento en el disco del dispositivo donde se ejecute, ya que se soporta en el servidor donde esté alojada. No depende de ningún Sistema Operativo (S.O), ya que se ejecuta en el navegador y está accesible en cualquier dispositivo que tenga acceso a *Internet*. Se actualiza por sí solo, de forma sencilla y en concreto, por parte del propio administrador de la aplicación encargado de ello. Sólo se necesita hacer una única copia de seguridad, con posibilidad de restaurar de forma sencilla en caso de que haya cualquier problema. Cuenta con una versión única y compatible con todos los diferentes navegadores. Es muy adaptable. Y su consumo de recursos es bajo.

El protocolo HTTP, que significa *Protocolo de Transferencia de Hipertextos* según sus siglas, es un protocolo de transmisión de información de la *World Wide Web* o de Internet basado en el código que se establece para que el ordenador solicitante y el que contiene toda la información solicitada puedan comunicarse en un mismo idioma a la hora de transmitir información por la red. Es considerado un protocolo que no lleva registro de visitas anteriores, ya que en su lugar, siempre empieza de nuevo.

Una característica relevante sobre el protocolo mencionado es que permite el cifrado de las comunicaciones extremo a extremo a través de *HTTPS*[12], que significa *Protocolo de Transferencia de Hipertextos Seguro* según sus siglas, encargado de cifrar las comunicaciones entre el servidor web y los clientes que se ejecutan en el navegador web. HTTPS es HTTP sobre *TLS*[13], que significa *Seguridad de la Capa de Transporte* según sus siglas.

Resumiendo, las aplicaciones web siguen un modelo de funcionamiento bajo el protocolo HTTP o HTTPS de peticiones solicitud-respuesta entre uno o varios clientes. Es decir, sigue el siguiente modelo de comunicaciones Cliente-Servidor Web:

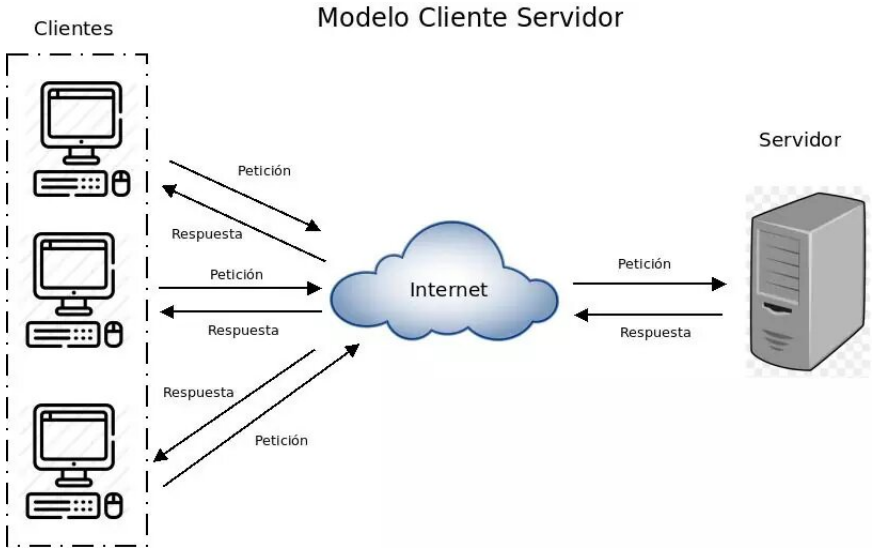


Figura 2.2: Modelo de comunicaciones Cliente-Servidor de una aplicación web, sacado de [14]

La imagen del modelo de arriba consiste en que los clientes envíen o realicen las solicitudes HTTP por medio de Internet al servidor web y éste, las responda a través de las respuestas HTTP. Incluso no sólo se establecen comunicaciones Cliente-Servidor de forma directa, sino que también se pueden establecer comunicaciones a través de un *Proxy*[15], que es una tecnología o dispositivo intermedio que permite establecer la comunicación entre el origen y el destino de manera indirecta, es decir, haciendo que toda la información pase primero por él.

Por último, la parte del cliente de una aplicación web que se ejecuta en un navegador, se puede clasificar en dos tipos:

1. Aplicaciones según cómo están programadas y diseñadas[16], divididas en aplicaciones web simples, progresivas, nativas e híbridas. Las primeras son conocidas por sus siglas en inglés, *SWA* (APPs Web Simples), son las más ligeras de todas porque no requieren de espacio de almacenamiento en disco de ningún dispositivo electrónico, se actualizan de manera permanente, su tiempo, su rendimiento y su coste de desarrollo es bajo, están diseñadas y adaptadas para ser consumidas sin necesidad de ninguna instalación, simplemente accediendo a ellas a través del navegador, su acceso es limitado a las funciones del dispositivo, las experiencias de usuario son diferentes en función del navegador utilizado y deben funcionar con conexión a Internet. Las segundas son conocidas por sus siglas en inglés, *PWA* (APPs Web Progresivas) y son un avan-

ce más reciente de las primeras, cuya diferencia con ellas está en que pueden ser descargadas e instaladas (como archivo ejecutable) en dispositivos electrónicos, tienen mayor rendimiento, ofrecen una funcionalidad similar al de las nativas, pueden funcionar sin conexión a Internet y en segundo plano, pero no se pueden subir a los mercados de aplicaciones. Las terceras son pesadas porque requieren de espacio de memoria en el dispositivo electrónico, tienen buen rendimiento, optimización y rapidez, están diseñadas y adaptadas para ser descargadas e instaladas (como archivo ejecutable) desde los mercados de aplicaciones y con la posibilidad de ser subidas a ellos, no se pueden acceder a ellas a través del navegador, las actualizaciones se pueden bloquear para que no consuman más espacio, pueden ser compatibles con los diferentes sistemas operativos, aprovechan al máximo las diferentes funcionalidades del dispositivo, pueden funcionar sin conexión a Internet y en segundo plano, su coste de desarrollo es alto porque cada sistema tiene un código diferente y complejo y el tiempo de desarrollo es bastante elevado. Las últimas combinan características de las APPs Web Simples y de las nativas, donde tienen un carácter multiplataforma con una sola línea de desarrollo, su tiempo de desarrollo, su coste y su rendimiento son inferiores al de las nativas, se pueden subir a los mercados de aplicaciones, pueden ser compatibles con los diferentes sistemas operativos y el aprovechamiento de las funcionalidades de un dispositivo es limitado.

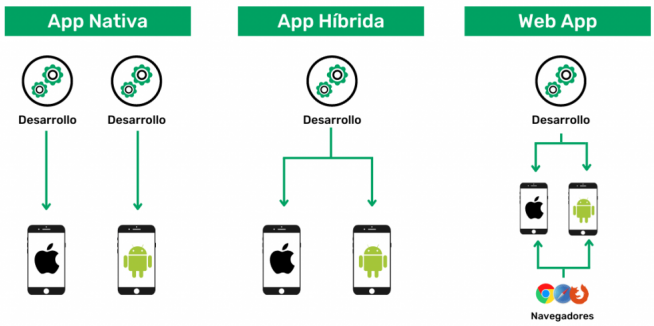


Figura 2.3: Esquema gráfico de los tipos de aplicaciones web según la programación y diseño de la interfaz de usuario, sacado de [17]

2. Aplicaciones según la categoría a desarrollar[18], divididas en aplicaciones web de comercio electrónico, estáticas, dinámicas, portales web, de gestión de contenidos y Plataformas de Comercio Online. Las primeras son tiendas de ventas virtuales que deben permitir pagos electrónicos por medio de métodos de pago actualizados, deben estar implementadas para mí-

nimo dos tipos de perfiles: el de cliente, quien hace las compras y pagos y el de administrador, quien controla que todo funcione de forma correcta, por ejemplo, cualquier tienda online de cualquier marca que venda productos. Las segundas son aquéllas que muestran poca información, sus contenidos no son sencillos de modificar, donde es necesario editar la interfaz de usuario o *front-end* y actualizarla en el servidor web, tampoco suelen tener muchos cambios y los que tienen se deben principalmente a la interfaz de usuario o *front-end*, no cuentan con bases de datos en la parte del servidor o *back-end*, por ejemplo, porfolios, currículums digitales o páginas de presentación de empresas. Las terceras son de las más complejas a nivel técnico porque utilizan bases de datos para cargar información y actualizar los contenidos en cada acceso dinámico, aunque el proceso de actualización es sencillo sin tener que acceder al servidor web para modificarlo, lo que permite la implementación de muchas funcionalidades, como tener implementado el perfil de administrador, quien controla que todo funcione de forma correcta, por ejemplo, cualquier aplicación web donde se inicie sesión. Las cuartas son aquéllas cuyo punto de entrada permite acceder a diversos apartados, por ejemplo, un aula virtual universitario, con posibilidad de acceso al correo electrónico institucional, a foros, a chats y a buscadores. Las quintas son aquéllas que son intuitivas, fáciles de usar, donde su contenido se va actualizando de forma continua, requieren de un gestor de contenidos o CMS que se encarga de administrar, pueden realizar los cambios y las actualizaciones oportunas, por ejemplo, WordPress, Joomla, o Drupal. Las últimas son las conocidas como *Marketplace*, espacios de compra-venta o plataformas virtuales que hacen de intermediarios entre los compradores y los vendedores, donde no hay límites entre la cantidad de artículos que se pueden vender y comprar y se podrían definir como un Centro Comercial online, por ejemplo, Amazon, AliExpress, o el Marketplace de Facebook.

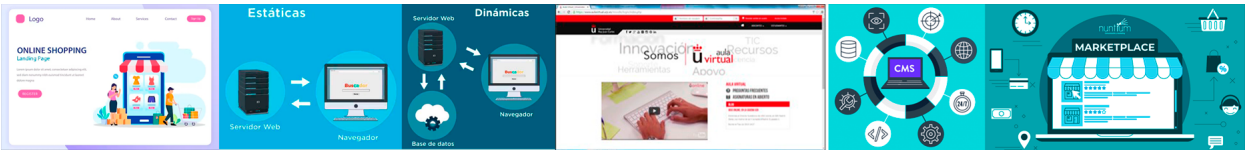


Figura 2.4: Representación gráfica de los tipos de aplicaciones web según las categorías a desarrollar en la interfaz de usuario, realizado y diseñado por el autor en *Adobe Photoshop*

En conclusión, los *Marketplace*[19] o Centros Comerciales online se dividen, a su vez, en tres tipos: los *B2B*, los *B2C* y los *C2C*. Los primeros son plataformas de compra-venta

entre empresas, por ejemplo, Crambo S.A[20]. Los segundos son plataformas de compra-venta entre Empresa y Cliente, por ejemplo, Amazon y AliExpress, pero sin dar visibilidad a las empresas que venden en él, ya que sólo aparecen como productos de la plataforma. Y los últimos son plataformas de compra-venta entre clientes o particulares, por ejemplo, el *Marketplace de Facebook*, *Wallapop* o *Vinted*.

Figura 2.5: Representación gráfica de los tipos de *Marketplace* existentes, sacado de [21]

Una vez explicado qué es una aplicación web y su funcionamiento, se puede pasar a la explicación de las tecnologías usadas en la aplicación web del prototipo del proyecto, definidas desde la más general a la más específica. Además, en cada una de ellas se explica el motivo de su elección con respecto a otras de su competencia. Entre ellas están: **Node JS** para tener un entorno de tiempo de ejecución en tiempo real donde ejecutar el servidor web, el *framework* de **Express JS** generado por *Node JS* para poder desarrollar ágilmente ese servidor mediante un conjunto de módulos, librerías y/o funcionalidades ya creadas, **MySQL** para establecer conexiones con las bases de datos, **phpMyAdmin** para mostrar las bases de datos en una interfaz gráfica visualmente intuitiva, **PUG** para mostrar la interfaz de usuario, **CSS** para dar diseño y apariencia a las vistas estáticas o motores de plantillas, **JavaScript** para establecer la lógica de programación del código fuente y las **Dependencias de Express JS** usadas para complementar y completar el prototipo del proyecto. También **Otras tecnologías** de soporte al proyecto en sí.

2.2. Node JS

Node JS[22] es un entorno de tiempo de ejecución de **JavaScript** en tiempo real, que fue creado por los desarrolladores originales de dicho lenguaje de programación. Pasó de ser un entorno donde sólo podía ejecutarse en el navegador, a poder ejecutarse en los computadores como

si de aplicaciones independientes se tratara. A partir de ese momento, *Node JS* hizo que se fuera más lejos de la simple programación con *JavaScript*, no sólo creando sitios web interactivos, sino también, teniendo la capacidad de desarrollar lo que otros lenguajes de programación podían desarrollar, como *Python*. Para ser más precisos, es importante destacar que tanto *JavaScript* como *Node JS* se ejecutan en el motor de tiempo de ejecución de código abierto, **JavaScript V8**, quien alimenta *Google Chrome* y, por eso, se decidió realizar este proyecto en esta tecnología, por su compatibilidad con potentes navegadores y por su capacidad de adaptación en el mercado. Para ello, la ventaja con respecto a otros, es que cuando se introduce un código *JavaScript* en *Node JS*, su motor de tiempo de ejecución lo convierte en un código de máquina de nivel más bajo que la computadora pueda ejecutar, sin necesidad de interpretarlo primero, ignorando la compilación y, por tanto, aumentando su velocidad.

Figura 2.6: Logo de *Node JS*

Su uso utiliza un modelo de solicitud y respuesta sin bloqueo, controlado por eventos que lo hace ligero y eficiente. Es decir, utiliza operaciones, desde leer o escribir archivos de cualquier tipo, a hacer solicitudes *HTTP*. Pero, la idea principal de dicho modelo, es seguir siendo liviano y eficiente frente a las aplicaciones en tiempo real de uso de datos que se ejecutan en los dispositivos. Esto hace que la finalidad de esta tecnología sea, no usarlo para programación de más peso donde sus ventajas se verían muy reducidas, sino usarlo para la creación de aplicaciones de red rápidas debido a su capacidad de manejar muchas conexiones simultáneas de gran rendimiento. Lo que significa que su escalabilidad es alta.

Internamente, su funcionamiento tiene una gran ventaja con respecto al resto de sus competidores, porque mientras que en otros se genera un nuevo subproceso por cada conexión, ocupando espacio en la memoria RAM y soliendo maximizar su cantidad disponible, en *Node JS* opera en un solo subproceso.

Pero principalmente, el por qué se ha elegido esta tecnología o entorno para desarrollar este proyecto, es debido a su alta escalabilidad, su poca sobrecarga de subprocesos, el tener un código fuente sencillo de aprender, el desarrollarse en entornos de tiempo de ejecución de fuente

libre para crear proyectos únicos, la posibilidad de incorporar estrategias de codificación que desembocan en la creación de abundantes aplicaciones de internet de gran competencia para el mercado, su manejo simultáneo de peticiones, el crecimiento exponencial de su comunidad de desarrolladores, el ser la plataforma de software más utilizada y, la opción de poder enviar archivos de gran peso.

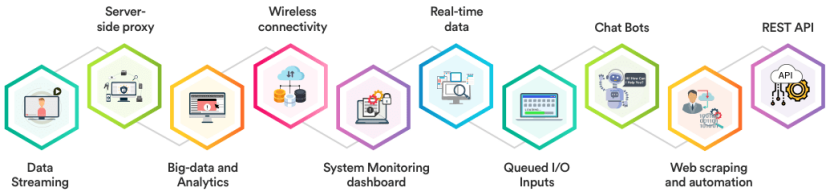


Figura 2.7: Imagen de las características del entorno de *Node JS*, sacado de [23]

2.3. Express JS

Express JS[24], o también conocido como **Express**, es una tecnología web o *framework* del entorno *Node JS*, mencionado anteriormente y usado para generar un proyecto de una aplicación web con su API (Interfaz de Programación de la Aplicación) o servidor web y su interfaz de usuario, a través de la instrucción del *Administrador de Paquetes de Node JS* de *npm install express-generator -g*.

Para ello, es importante entender que un *framework* es una herramienta o tecnología web basada en un conjunto de módulos que permiten el desarrollo ágil de aplicaciones mediante librerías y/o funcionalidades ya creadas, destacando que fue desarrollado para que los desarrolladores no estuvieran reinventado la rueda, se centraran en el problema a resolver y no en implementar funcionalidades de uso común que ya han sido resueltas por otros. Por otro lado, la ventaja de usar un *framework* frente a no usarlo, es porque tienen orden y control tanto en el código como en la organización de los archivos, porque están basados en la Arquitectura MVC (Modelo, Vista y Controlador), donde el Modelo es quien se encarga del código fuente que establece co-

Figura 2.8: Logo de *Express JS*

nexiones con las bases de datos, la Vista es quien se encarga del contenido de la interfaz de usuario y el Controlador es quien se encarga de conectar el código fuente del Modelo y el de la Vista a través de las solicitudes *HTTP*. También porque cuentan con protección y seguridad, porque tienen una comunidad de apoyo y porque fomentan el trabajo en equipo.

Hay muchos tipos[25] de *framework*, como por ejemplo: *Angular*, *Laravel*, *Symfony 2*, *Django*, *Phalcon*, *Spring Framework*, *Express JS*, *Struts* o *Zend*. Pero se ha elegido *Express JS* sobre el resto, debido a que es una tecnología minimalista, flexible, rápida, similar a la tecnología web o *framework* de *Sinatra*, robusta para desarrollar aplicaciones escalables y fiables. Además, ofrece un sistema de enrutamiento y características simplificadas para ampliarlo con componentes y partes más potentes en función de su uso, proporciona solicitudes y respuestas HTTP, construye y despliega aplicaciones a gran escala gracias a dicho enrutamiento y a sus *middlewares*[1] y cuenta con una herramienta CLI (Interfaz de línea de comandos) llamada NPM (Administrador de paquetes de Node JS) para obtener dependencias, módulos, librerías y/o funcionalidades. Además, se puede integrar fácilmente en el conjunto de tecnologías *Stack MEAN*[26] necesarias para el desarrollo de todas las capas o pilas de una aplicación web, puede conectarse con sistemas de gestión de bases de datos robustas como *MySQL* y es compatible con el motor *V8 de Google* cuya ventaja hace que sea muy potente para construir y desplegar aplicaciones en tiempo real, colaborativas y basadas en la red a nivel empresarial, recibiendo un enorme impulso de rendimiento y escalabilidad.

Como ya se ha comentado, **Express JS** está basado en la **Arquitectura MVC**.

2.3.1. Modelo

El **Modelo** de la **Arquitectura MVC** cuenta con las siguientes tecnologías para poder establecer conexiones con las bases de datos:

[1]Los *middlewares* son sistemas que ofrecen funciones y servicios de nube comunes para las aplicaciones, con el fin de implementarlas con la mayor eficiencia posible

MySQL

MySQL[27], que significa Mi Lenguaje de Consulta Estructurado por sus siglas, es un sistema de código abierto de administración de bases de datos, que puede ser usado y modificado a gusto del desarrollador. Desarrollado en 1995, pasando por varios administradores o propietarios, antes de acabar en *Oracle* en 2010. Para ello, una base de datos es una colección estructurada de datos, organizada para ser usada y extraída de forma sencilla.

Figura 2.9: Logo de *MySQL*

Por lo que, el funcionamiento de *MySQL* se entiende por dos conceptos:

- Las bases de datos relacionales, basadas en la fragmentación de los datos en varias áreas de almacenamiento separadas, conocidas como *tablas*, por una cuestión de organización. Cada tabla puede tener el nombre que se quiera, pero sin repetirse para evitar duplicidades y, cada una de ellas, está compuesta por columnas, que son los campos o variables que componen una tabla. A su vez, cada columna contiene filas, compuestas de información que un usuario va introduciendo por la Interfaz de Programación del Servidor Web. La parte relacional de una base de datos sería cuando se quiere consultar una tabla en concreto, tanto entera como por filas específicas, por medio de una llave primaria o columna única e irrepetible.

- El Modelo de Cliente-Servidor, basado en dos partes. El Servidor es el lugar donde residen los datos y el Cliente es quien pide acceder a ellos a través de peticiones *Query*, que son cuatro: para consultar de la base de datos, para insertar en base de datos, para borrar de la base de datos y para actualizar en base de datos.

Para concluir, la explicación de por qué se ha escogido esta tecnología de establecimiento de conexiones con las bases de datos, es porque *MySQL* es de fácil instalación a partir de pasos que se describirán en el *Capítulo 3*, tiene muy buen rendimiento debido a su velocidad y bajo consumo de recursos, es muy flexible y altamente escalable, es muy popular y forma parte de la muy conocida pila de código abierto *LAMP* (*Linux*, *Apache*, *MySQL*, *PHP*/*Perl*/*Python*).

phpMyAdmin

phpMyAdmin[28] es una herramienta de software libre escrito en *PHP*, destinado a ser la interfaz de administración de bases de datos de *MySQL* a través de la web. Para poder usarlo, es necesario un servidor web tipo *Apache*, una cuenta de alojamiento *MySQL* para acceder a la base de datos y la instalación del lenguaje *PHP*, base de *phpMyAdmin*. Para ello, la primera razón por la que se ha elegido *phpMyAdmin* es, porque es mucho más visual e intuitivo a la hora de mostrar la interfaz de usuario de las bases de datos. La segunda, porque la organización y manipulación de las bases de datos es mucho más sencillo e intuitivo con la capacidad de ejecutar directamente cualquier sentencia *SQL*. La tercera, porque es una tecnología web con múltiples funcionalidades como exportar o importar bases de datos, gestionar cuentas de usuario. Todo ello de forma sencilla. Y la cuarta, porque cuenta con un sistema de seguridad robusto, seguro y auditado por la empresa *NCC Group*, por lo que aumenta la imposibilidad de ser atacado informáticamente.

Figura 2.10: Logo de *phpMyAdmin*

Para conectarse y acceder a ello, sólo se necesita activar las tecnologías de *Apache* y *MySQL* bajo el lenguaje base de *PHP* e introducir el usuario y la contraseña.

Apache

Apache[29] es un servidor web de código abierto, multiplataforma, gratuito y uno de los más utilizados en el mundo, usado por el 43 porciento de los sitios webs. Desarrollado por *Apache Software Foundation* desde 1995.

Figura 2.11: Logo de *Apache*

Es un software necesario si se quiere utilizar y conectarse a *phpMyAdmin*, donde podrás usar *.htaccess* para trabajar de forma más sencilla con los principales *Sistemas de Gestión de Contenidos* (CMS). Su nombre hace referencia a la tribu de los nativos americanos, conocidos por su gran resistencia en el combate y por sus estrategias de guerra.

Su funcionamiento se basa en servir las webs alojadas en ello a los diversos navegadores, consiguiendo que la comunicación sea fluida y constante. Por otro lado, se aplica una alta personalización a través de su sistema modular, de forma que podemos activar o desactivar diversas funcionalidades a través de los módulos de *Apache*, pero usándolos con cautela debido a que puede afectar a la seguridad y funcionalidades del servidor web.

PHP

PHP[30], cuyo acrónimo es Preprocesador de Hipertexto, es un lenguaje de código abierto muy popular y especialmente adecuado para el desarrollo web, que puede ser incrustado en *HTML*.

Para aclarar conceptos, el desarrollo de *PHP* está centrado en la programación de scripts del lado del servidor. Lo que hace que sea muy usado en la configuración e instalación de *phpMyAdmin* como interfaz de usuario para administrar bases de datos.

Figura 2.12: Logo de *PHP*

2.3.2. Vista

La **Vista** de la **Arquitectura MVC** cuenta con las siguientes tecnologías para poder mostrar el contenido de la interfaz de usuario, conocidos como motores de plantillas o vistas estáticas que funcionan con el *framework* de *Express JS*, entre los que destacan: *EJS*, *Handlebars (HBS)*, *Jade*, *Twig* (este en concreto también funciona con el *framework* de *Symfony 2*) y *PUG*, que es el que ha sido el elegido para desarrollar la interfaz de usuario del prototipo de este proyecto:

PUG

PUG[31] es un motor de plantillas del *framework* de *Express JS*, cuya sintaxis es la más sencilla, clara y directa, tanto a la hora de escribir como de leer y modificar, de entre todas las

vistas estáticas o motores de plantillas que hay. También, su código fuente es limpio y conciso a la hora de trabajar y corregir errores. Funciona de acuerdo a la sintaxis de *HTML*, pero simplificando las etiquetas.

Por eso *PUG* ha sido el motor de plantillas o vista estática escogida para desarrollar la interfaz del prototipo del proyecto.

Figura 2.13: Logo de *PUG*

Además, la **Vista** cuenta con tecnologías para dar diseño y apariencia a la interfaz de usuario, donde se ha elegido la tecnología de *CSS*[32], que es un lenguaje de programación que permite crear, realizar y manejar el diseño, los cambios, las mejoras, la presentación y la personalización de las páginas de una interfaz de usuario de una aplicación web, de forma rápida y sencilla a través de hojas de estilo en cascada. Es decir, dicha tecnología determina el aspecto de las páginas de la interfaz de usuario, como el tipo de fuente, los colores, los fondos, los bordes, las líneas, entre otros elementos y, permite cambiar la apariencia de un sitio sin tener que modificar manualmente el contenido.

Figura 2.14: Logo de *CSS*

Sirve tanto para crear un solo archivo *CSS*, como varios de ellos, donde unos tengan propiedades heredadas de otros, lo que se le conoce como *diseño de estilos en cascada*.

Surgió en 1994 gracias al desarrollador web noruego *Håkon Wium*, quien presentó y desarrolló la idea de hojas en cascada para hacer el trabajo más sencillo a los programadores y desarrolladores.

Pero la explicación de por qué se ha elegido esta tecnología con respecto a otras de su competencia, es porque puede realizar cambios a la apariencia de una web sin alterar su contenido, tiene el control en el diseño de documentos HTML, puede hacer modificaciones a elementos

similares en forma de cascada, puede organizar por preferencias de estilo, ahorra tiempo en el diseño y personalización de una web, puede adaptarse a cualquier tamaño de pantalla, simplifica la creación de un sitio web, optimiza el rendimiento de las páginas web al mejorar los tiempos de carga y tráfico del servidor, hace posible la personalización de una página en su totalidad, es compatible con todos los navegadores y plataformas, permite separar el contenido del diseño, mantiene la misma apariencia en todas las secciones de un sitio web, es muy preciso con los tipos, tamaños de letras y colores y, hace mejorar la escalabilidad de la aplicación web.

Para finalizar, es importante conocer que las hojas de estilo *CSS* se componen de dos elementos: *declaraciones* y *selectores*. Las primeras son elementos que indican lo qué se debe hacer, compuestas por una propiedad. Las segundas son fases que indican elementos a quienes se les aplica una determinada *declaración*, sus tipos son: el selector universal, el selector etiqueta, el selector descendente, el selector de clase y el identificador. El primero afecta a todos los elementos de una página web. El segundo afecta a los elementos que tienen una etiqueta *HTML* del mismo valor. El tercero hace que el segundo sea más preciso, eligiendo a todos los elementos dentro de otros elementos que comparten esa etiqueta. El cuarto afecta a todo aquel que tenga el atributo de un punto y un valor. Y el quinto afecta a todo aquel que tenga el atributo de una almohadilla y un valor.

2.3.3. Controlador

El **Controlador** de la **Arquitectura MVC** es el intermediario entre el Modelo y la Vista, que depende del servidor web o API (Interfaz de Programación de la Aplicación). Usa la tecnología de *JavaScript* para llevar a cabo la lógica de prototipo del proyecto, pero sin ser el único que la usa, ya que todo el servidor web funciona con ese lenguaje de programación, al estar tanto *Node JS* como *Express JS* basados en *JavaScript*:

JavaScript

JavaScript[33], o también conocido como **JS**, es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, en vez de hacer páginas estáticas que sólo muestren información para que las veas. Entre su funcionalidad destaca: la programación de mapas interactivos, la animación de gráficos 2D/3D (dos dimen-

siones/tres dimensiones), o el desplazamiento de máquinas reproductoras de vídeo. Es decir, *JS* permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes, o encaminar las diferentes rutas (GET, POST, PATCH o DELETE) de una aplicación web por medio de un controlador del proyecto a desarrollar a partir de la sentencia de código *var variable = servidor.Router();* sacado de la tecnología o dependencia del *framework* de *Express JS* con la siguiente sentencia: *var servidor = require('express');*.

Figura 2.15: Logo de *JavaScript*

2.4. Dependencias de Express JS

Express JS es un *framework* generado en el entorno de *Node JS* para agilizar el desarrollo del servidor mediante **Dependencias** o módulos a instalar que complementan y completan el desarrollo de la aplicación web. En este caso se han necesitado las siguientes:

2.4.1. Axios

Axios[34] es una tecnología para poder hacer peticiones o llamadas al contenido de un enlace HTTP. Funciona como la tecnología o librería ya obsoleta de *request*, es decir, *Axios* viene a ser su nueva versión.

Figura 2.16: Representación gráfica del funcionamiento de la tecnología de *Axios*, sacado de [35] y editado por el autor en *Adobe Photoshop*

2.4.2. Bcrypt

Bcrypt[36] es una tecnología de cifrado de contraseñas, diseñado por Niels Provos y David Maxieres, basado en el cifrado de *Blowfish*[37], un codificador de bloques simétricos del algoritmo **SHA1** de 64 bits y claves que van desde los 32 bits hasta 448 bits. *Bcrypt* lleva incorporado un atributo conocido como *SAL*, fragmento aleatorio usado para generar el cifrado asociado a la contraseña y almacenado junto a ella en la base de datos, con el objetivo de evitar dos contraseñas iguales generadas por el mismo cifrado. Su funcionamiento práctico es el siguiente:

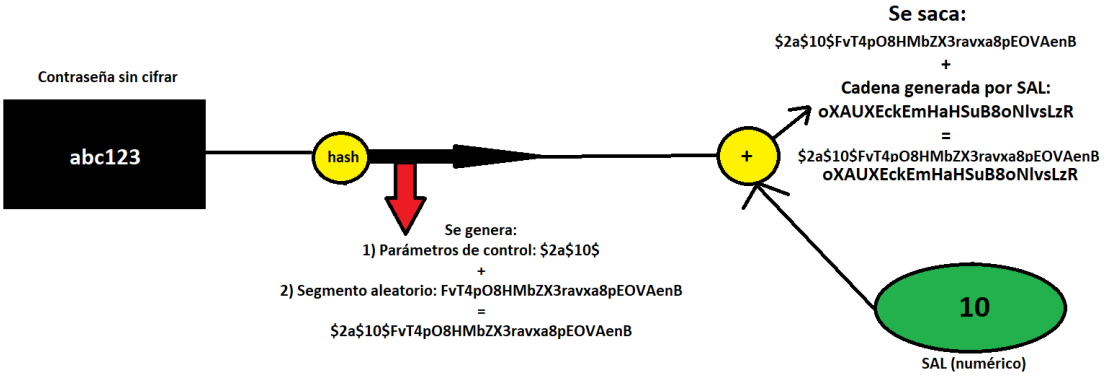


Figura 2.17: Representación gráfica del funcionamiento del cifrado con la tecnología de *Bcrypt*, editado por el autor en *Adobe Photoshop*

Aparte de cifrar, *Bcrypt* también es una tecnología de autenticado, donde se compara la contraseña sin cifrar introducida con la contraseña cifrada de la base de datos. Es evidente que la comparación entre ambas es inviable debido al tedioso trabajo de primero, descifrar el *SAL* introducido y, después, descifrar su cifrado general aleatorio más los parámetros de control. Esto hace que compense más cifrar de la misma forma la contraseña introducida y entonces comparar ambas. Su funcionamiento práctico es el siguiente:

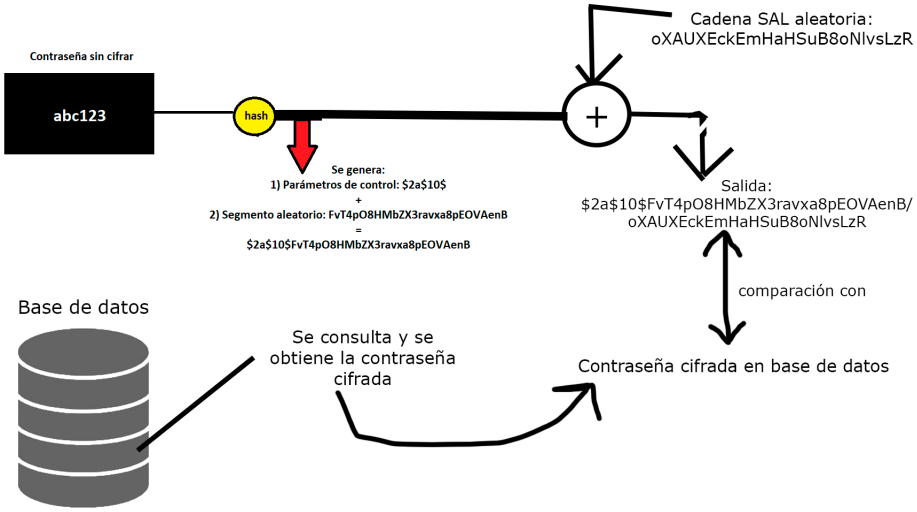


Figura 2.18: Representación gráfica del funcionamiento del autenticado con la tecnología de *Bcrypt*, editado por el autor en *Adobe Photoshop*

2.4.3. Body-parser

Body-parser[38] es una tecnología para analizar y procesar los datos de solicitudes *HTTP*, como JSON o datos de formulario, permitiendo acceder a los datos del cuerpo de la solicitud en un formato muy utilizable y sencillo.

```
JS index.js > ...
1   const express = require('express');
2   const bodyParser = require('body-parser');
3
4   const app = express();
5   app.use(bodyParser.json());
```

Figura 2.19: Representación gráfica del funcionamiento de la tecnología de *Body-parser*, sacado de [39]

2.4.4. Card-validator

Card-validator[40] es una tecnología para validar o verificar de forma oficial el número de tarjeta bancaria y el código CVV introducido por un usuario.

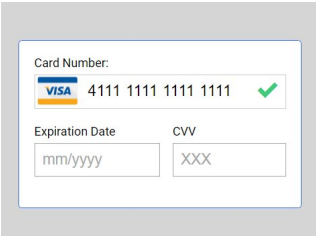


Figura 2.20: Representación gráfica del funcionamiento de la tecnología de *Card-validator*, sacado de [41]

2.4.5. Cookie-parser

Cookie-parser[42] es una tecnología para analizar el encabezado *Cookie* y rellenar *req.cookies* con objeto marcado con los nombres de las *Cookies*. Cuando estableces una *Cookie*, se configura su tiempo de expiración y se puede comprobar su disponibilidad a través de *document.cookie*. No obstante, una *Cookie* es una pequeña porción de datos enviados por un sitio web y son almacenados en el navegador del usuario mientras se navega por él, para cuando se vuelva a iniciar sesión, el navegador las use para enviárselas al servidor web y, así, las reconozca.

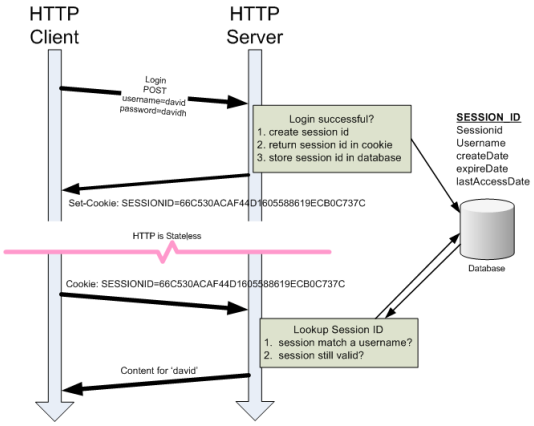


Figura 2.21: Representación gráfica del funcionamiento de la tecnología de *Cookie-parser*, sacado de [43]

2.4.6. CORS

CORS[44] es una tecnología para controlar el acceso HTTP intercambiando recursos de origen cruzado (Crossed Origin Resources - CORS), con el objetivo de utilizar cabeceras HTTP

adicionales y permitir que un agente de usuario obtenga permiso para acceder a recursos seleccionados desde un servidor web en un origen distinto al que pertenece.

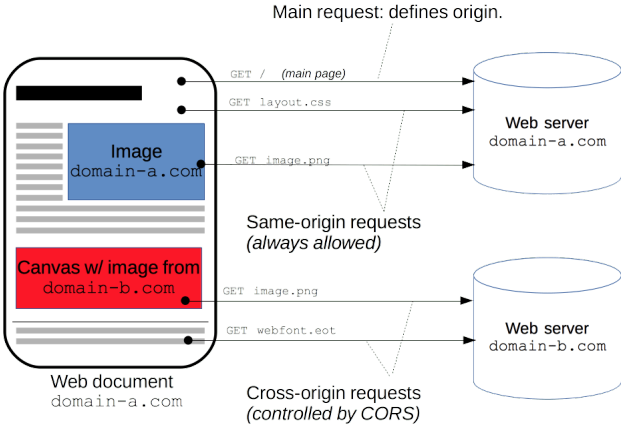


Figura 2.22: Representación gráfica del funcionamiento de la tecnología de *CORS*, sacado de [44]

2.4.7. Country-list y Country-list-spanish

Country-list[45] es una tecnología para listar los países existentes en el mundo en inglés con la función *getNames()* y para verificarlos con el país introducido por medio de la función *includes()*. También puede obtener los códigos de los países y verificarlos.

Country-list-spanish[46] es lo mismo, pero en español.

Figura 2.23: Representación gráfica del funcionamiento del listado de países con las tecnologías de *Country-list* y *Country-list-spanish*, sacado de [47]

2.4.8. Dotenv

Dotenv[48] es una tecnología para configurar y administrar la configuración de las aplicaciones por separado del código base, utilizando variables de entorno externas, almacenadas en un archivo o entorno de desarrollo, para poder utilizar su contenido y evitar que se muestre en el código base.

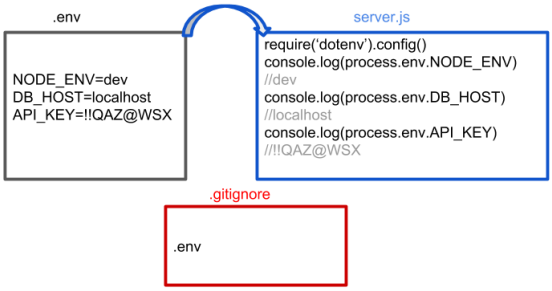


Figura 2.24: Representación gráfica del funcionamiento de la tecnología de *Dotenv*, sacado de [49]

2.4.9. Express-session

Express-session[50] es una tecnología para establecer sesiones o autenticar usuarios en una aplicación web de *Node JS* y *Express JS*. También para destruir sesiones o cerrar sesiones autenticadas previamente.

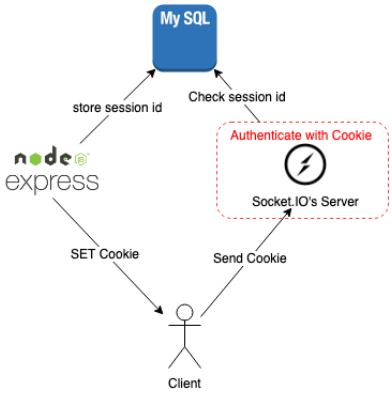


Figura 2.25: Representación gráfica del funcionamiento de la tecnología de *Express-session*, sacado de [51]

2.4.10. Fs

Fs[52] es una tecnología para interactuar con el sistema de archivos, leyendo y escribiendo un directorio específico y/o ficheros.

```
/**
 * Leer un archivo con Node.JS línea
 * por línea usando fs y readline
 *
 * @author parzibyte
 */
const readline = require("readline"),
    fs = require("fs"),
    NOMBRE_ARCHIVO = "archivo.txt";
```

Figura 2.26: Representación gráfica del funcionamiento de la tecnología de *Fs*, sacado de [53]

2.4.11. Helmet

Helmet[54] es una tecnología para darle seguridad a las cabeceras *HTTP* de la aplicación.

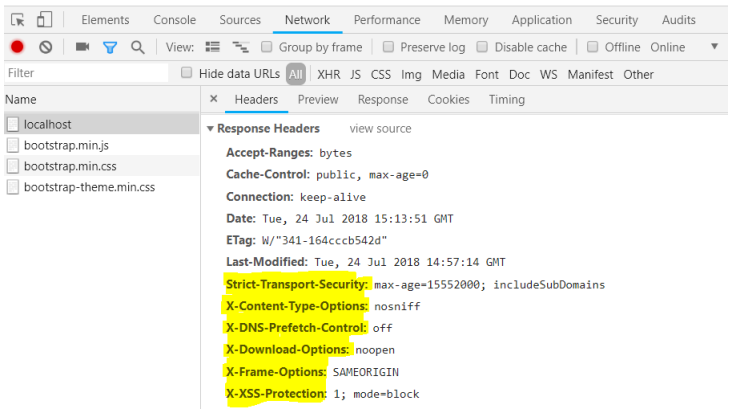


Figura 2.27: Representación gráfica del funcionamiento de la tecnología de *Helmet*, sacado de [55]

2.4.12. Http-errors

Http-errors[56] es una tecnología para indicar errores internos que la aplicación tiene en el código y, así, el servidor no pueda generar código *HTML* para devolver al usuario.

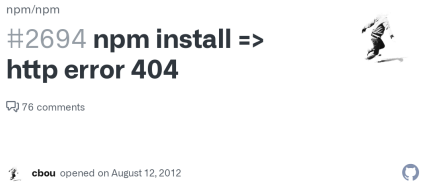


Figura 2.28: Representación gráfica de la configuración de un tipo de error con la tecnología de *Http-errors*, sacado de [57]

2.4.13. Luxon

Luxon[58] es una tecnología para obtener una fecha y hora específica, de acuerdo a un tipo de huso horario.

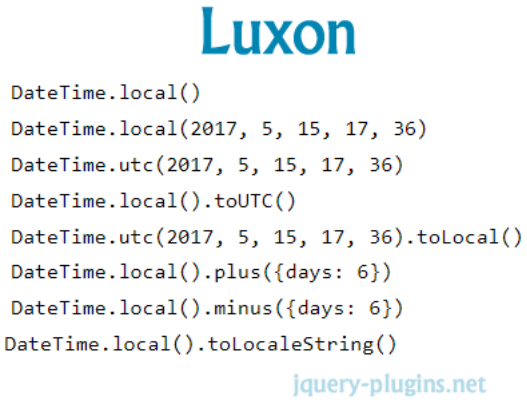


Figura 2.29: Representación gráfica del funcionamiento de la tecnología de *Luxon*, sacado de [59]

2.4.14. Method-override

Method-override[60] es una tecnología para poder utilizar en una aplicación web de *Node JS Express* más métodos como *PATCH* y *DELETE* aparte de los principales *GET* y *POST*. Esto se debe a que los dos primeros métodos mencionados no funcionan ni se reconocen en ese tipo de aplicaciones, porque para *Express JS* son métodos *POST* que manejan datos, ya sean para enviar, actualizar o borrar. Por tanto, para que puedan funcionar, es necesario instalar la tecnología e incluir explícitamente todos aquellos métodos que se vayan a usar en la aplicación.

```
const patchdeletemethods = require('method-override');
var madservices = servidor();
madservices.use(patchdeletemethods('_method', { methods: ['POST', 'GET', 'PATCH', 'DELETE'] }));
form(method='POST' action=`/sesion-cliente/${id}/perfil/borrar-tarjeta`)
input(type='hidden' name='_method' value='DELETE')
form(method='POST' action=`/sesion-cliente/${id}/perfil/actualizar-localizacion`)
input(type='hidden' name='_method' value='PATCH')
```

Figura 2.30: Representación gráfica del funcionamiento de la tecnología de *Method-override*, sacado del código del prototipo del proyecto

2.4.15. Multer

Multer[61] es una tecnología para el manejo de *multipart/form-data*, es decir, para el manejo y subida de archivos de imagen y vídeo.

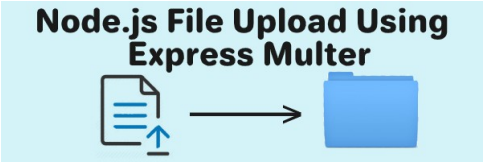


Figura 2.31: Representación gráfica del funcionamiento de la tecnología de *Multer*, sacado de [62]

2.4.16. Nif-dni-nie-cif-validation

Nif-dni-nie-cif-validation[63] es una tecnología para verificar o validar de forma oficial el código CIF o NIF del usuario. Sólo válido para España.

```
const cifvalidacion = require('nif-dni-nie-cif-validation');
}else if((cifvalidacion.isValidCif(nif) === false) && (cifvalidacion.isValidNif(nif) === false)) {
    res.status(401).render('paginas/empresas/registrarse', {mensaje: 'El CIF/NIF no es oficial'});
    return res.end();
```

Figura 2.32: Representación gráfica del funcionamiento de la tecnología de *Nif-dni-nie-cif-validation*, sacado del código del prototipo del proyecto

2.4.17. Node-notifier

Node-notifier[64] es una tecnología para sacar por pantalla notificaciones de avisos o alertas emergentes, dadas las circunstancias de la aplicación. Con posibilidad de que sean con sonido

y de diseñar un logo incorporado según se requiera.

Figura 2.33: Representación gráfica del funcionamiento de la tecnología de *Node-notifier*, sacado de [65]

2.4.18. Path

Path[66] es una tecnología para trabajar con rutas de fichero, en cuyo caso cuenta con submódulos como, *path.normalize(p)*, que normaliza la cadena de texto de una ruta, *path.join([path1], [path2], [...])*, que junta todos los argumentos y normaliza la ruta resultante, o *path.dirname(p)*, que devuelve el nombre del directorio de una ruta.

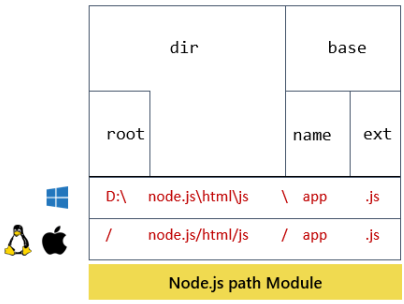


Figura 2.34: Representación gráfica del funcionamiento de la tecnología de *Path*, sacado de [67]

2.4.19. Postcode-validator

Postcode-validator[68] es una tecnología para validar o verificar de forma oficial los códigos postales de cada país del mundo.

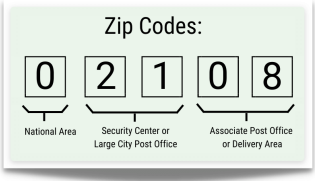


Figura 2.35: Representación gráfica del funcionamiento de la tecnología de *Postcode-validator*, sacado de [69]

2.4.20. Serve-favicon

Serve-favicon[70] es una tecnología para servir un *favicon* en la pestaña del navegador. No obstante, dicho concepto se define como una señal visual que los navegadores usan para identificar un sitio web.

Principalmente se ha decidido usar esta subtecnología porque almacena en caché el icono en la memoria para mejorar el rendimiento sin acceder al disco duro, proporciona un *ETag* basado en el contenido del icono y se sirve con los más compatibles *Content-Type*.

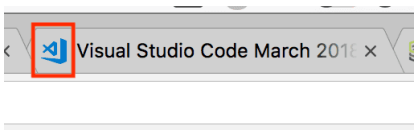


Figura 2.36: Representación gráfica del funcionamiento de la tecnología de *Serve-favicon*, sacado de [71]

2.4.21. Sharp

Sharp[72] es una tecnología para procesar apresuradamente una imagen, por ejemplo, rediseñando o cambiando su tamaño, admitiendo múltiples formatos de imagen.

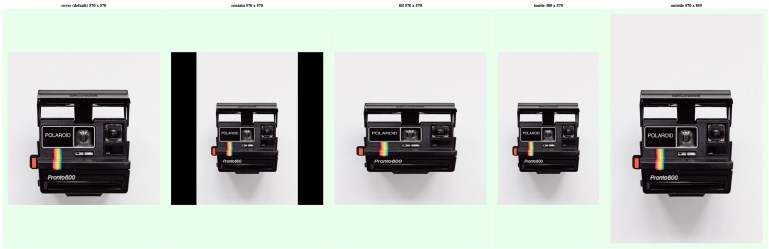


Figura 2.37: Representación gráfica de los tipos de tamaños al procesar una imagen con la tecnología de *Sharp*, sacado de [73]

2.4.22. Util

Util[74] es una tecnología para acompañar en la lectura y escritura del sistema de archivos, directorios y/o ficheros a la subtecnología **Fs**.

```
const readdir = util.promisify(fs.readdir);
//-- Fichero asíncrono leer fichero.
const readFile = util.promisify(fs.readFile);
//-- Fichero asíncrono borrar fichero.
const unlink = util.promisify(fs.unlink);
```

Figura 2.38: Representación gráfica del funcionamiento de la tecnología de *Util*, sacado del código del prototipo del proyecto

2.4.23. Validator

Validator[75] es una tecnología para validar las entradas de textos, números, emails y contraseñas que se introducen por el *Body* en una aplicación web.

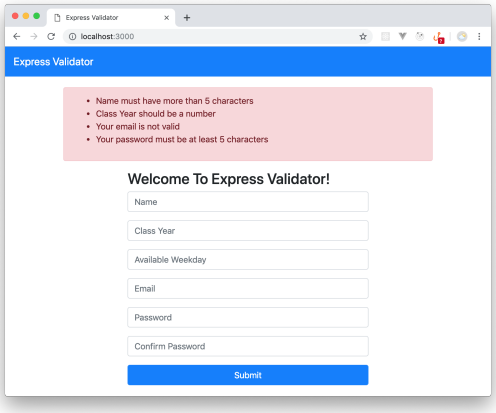


Figura 2.39: Representación gráfica del funcionamiento de la tecnología de *Validator*, sacado de [76]

2.4.24. Nodemon

Nodemon[77] es una tecnología de desarrollo para monitorear cualquier aplicación web en proceso de desarrollo y reiniciarla automáticamente cuando se detectan cambios en los archivos.

Se ha elegido principalmente porque reinicia el servidor de forma automática, puede ignorar archivos específicos, es multiplataforma y fácil de ejecutar.

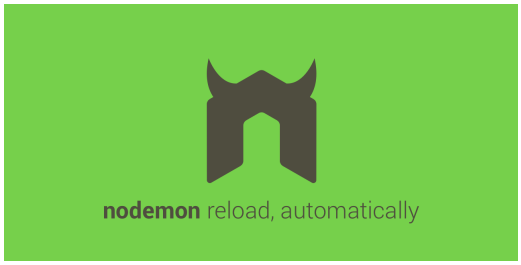


Figura 2.40: Representación gráfica del logo de *Nodemon*, sacado de [78]

2.4.25. Debug

Debug[79] es una tecnología para depurar o mostrar en consola el valor de variables o errores, ya que permite habilitar o deshabilitar mensajes en consola, enviando un valor indicando si debe habilitar o no.

Figura 2.41: Logo de *Debug*, sacado de [80]

2.5. Otras tecnologías

Por último, destacan otras tecnologías, utilizadas como ayuda externa al desarrollo del prototipo del proyecto. Son: **Overleaf**, para la redacción de la memoria del proyecto, **Visual Studio Code**, para la realización del código fuente del prototipo, **Thunder Client**, para simular llamadas a la API (Interfaz de Programación de la Aplicación), **Git**, para tener un servicio de control de versiones con mantenimiento activo y poder conectar con servicios en la nube, **GitHub**, para poder almacenar el código del prototipo en la nube, **Adobe Photoshop**, para poder editar todas y cada una de las imágenes que haya en el proyecto y **Adobe Premiere/Wondershare Filmora 9**, para poder editar todos y cada uno de los GIFs o vídeos que haya en ello.

2.5.1. Overleaf

Overleaf es un servicio o herramienta online del software **LaTeX**[81], utilizado para dar formato, redactar, editar y publicar de forma colaborativa, artículos, investigaciones, documentos técnicos, científicos o de Ingeniería, pósteres, diapositivas y/o cualquier tipo de documento en general[82]. Su creación vino por medio del lenguaje **TeX**[83], lenguaje base de *LaTeX* de bajo nivel que genera mucho código fuente para poder conseguir el resultado deseado, pero con un gran potencial, ya que genera bloques de código de instrucciones complejas a partir de otras más simples. Fue ideado en 1978 por el Dr. Donald Knuth, un reconocido experto en *Ciencias de la Computación* y matemático estadounidense, famoso por su investigación en el análisis de algoritmos y compiladores, que gracias a él y a su lenguaje se hizo posible la creación de *LaTeX*. En este caso, el autor que lo desarrolló fue el Dr. Leslie Lamport[84], un matemático y científico de la computación estadounidense, conocido por sus trabajos en *Sistemas Distribuidos*, que también desarrolló el *Sistema de Formateo de Textos* **BibTeX**, herramienta computacional para el manejo de referencias bibliográficas.

Figura 2.42: Logo de *Overleaf*

LaTeX, al contrario que *TeX*, es un lenguaje de alto nivel, mucho más sencillo de usar, se conecta con la nube, se puede utilizar en los Sistemas Operativos (S.O) de Linux, MAC OS y Windows, incluso online, a través de servicios web como *Papeeria*, *LaTeX base*, *CoCalc* y el ya mencionado *Overleaf*. Su versión más reciente es, **LaTeX 2E**, que data de 1994.

Volviendo a *Overleaf*, cabe destacar que su uso es sencillo y rápido, con posibilidad de descargar en archivo PDF y donde, mientras se edita, se compila automáticamente en tiempo real. Los requisitos mínimos para acceder a *Overleaf* son: tener una cuenta de email personal o de una institución y, tener el navegador actualizado.

Características principales de *Overleaf*[85]:

- No es necesaria su instalación ni actualización.

- Funciona en todos los entornos y con cualquier dispositivo.
- Es una herramienta colaborativa.
- Facilidad para compartir documentos gracias a su acceso a la nube.
- Su compilador está integrado y, por tanto, la visualización del resultado al compilar es inmediata.
- Cuenta con plantillas ya diseñadas para Trabajos de Fin de Grado, de Fin de Máster y Tesis Doctorales.
- Puedes desarrollar proyectos ilimitados en tu cuenta.
- Hay un historial completo de documentos.
- Puede hacer posible la restauración de versiones anteriores del documento.

Para finalizar, los motivos por el que se ha elegido esta tecnología son debido a que cuenta, sobre todo, con plantillas para Trabajos de Fin de Grado de la Escuela Técnica Superior de Ingeniería de Telecomunicaciones de la Universidad Rey Juan Carlos y, porque el tutor a cargo de este proyecto la recomienda por su rápida compilación instantánea, por la variedad de dichas plantillas mencionadas, por ser un lenguaje intuitivo y sencillo de aprender y, porque cuenta con un manual de sintaxis de ayuda para usuario[86] bastante explicativo y de cómodo entendimiento.

2.5.2. Visual Studio Code

Visual Studio Code[87] es un editor de código fuente optimizado y desarrollado por Microsoft, cuya función es escribir, probar y depurar código. Es bastante popular y con bastante aceptación entre los desarrolladores e informáticos. Según una encuesta realizada en 2021 por *Stack Overflow*, *Visual Studio Code* fue usado por el 71.06 porciento de los 82277 desarrolladores encuestados.

Figura 2.43: Logo de *Visual Studio Code*

Sus características consisten en que es un software libre y multiplataforma, disponible para Windows, GNU/Linux y macOS, cuenta con una buena integración con *Git*, por lo que puedes organizar archivos, realizar *commits* desde el editor y hacer

push y *pull* desde cualquier servicio de gestión de código fuente (SMC) a plataformas de servicio basado en la nube, como *GitHub* o *GitLab* y cuenta con múltiples extensiones, dando la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación, lo que lo hace un editor de código potente, ya que las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada, consiguiendo mejores experiencias tipo, conectar con otros servicios o agregar nuevos temas al editor y, lo más importante, sin afectar al rendimiento del mismo por su ejecución en procesos independientes. Respecto a su edición de código, autocompletado y resaltado de sintaxis, está basado en *IntelliSense*, función que permite al desarrollador ser más ágil a la hora de escribir código, proporcionar sugerencias de código y terminaciones inteligentes en base a los tipos de variables y/o funciones y, gracias a las extensiones, podría personalizarse y conseguirse un *IntelliSense* más completo. Además, cuenta con una función de depuración al ejecutar el código, para detectar errores en ello y, así, evitar tener que revisar línea por línea el error o errores, cabiendo la posibilidad de detectar errores de forma automática antes de ejecutar el código.

Para finalizar, los motivos por el que se ha elegido esta Tecnología son debido a sus características y grandes ventajas respecto al resto de sus competidores y, porque al autor le resulta bastante familiar e intuitivo.

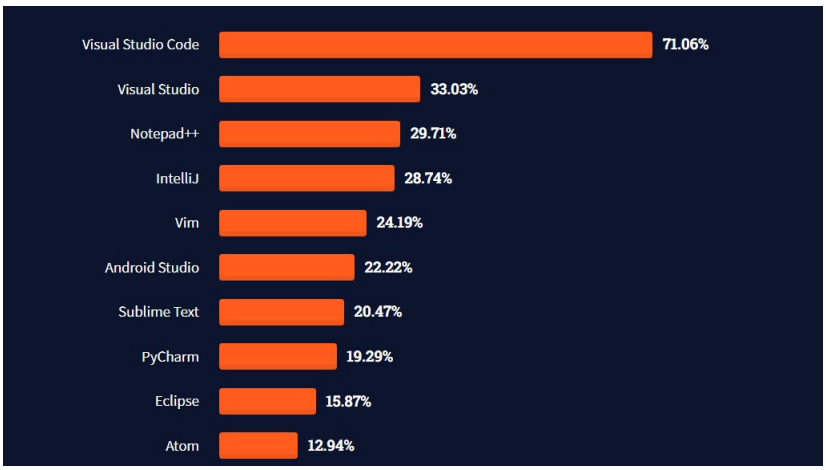


Figura 2.44: Encuesta de Stack Overflow de 2021 sobre la popularidad de los diferentes editores de código fuente que hay, sacado de [88]

2.5.3. Thunder Client

Thunder Client[89] es una extensión liviana y muy sencilla de *Visual Studio Code* para simular llamadas a la API (Interfaz de Programación de Aplicaciones). Fue elegido frente a otros de su competencia, como *Postman*, porque está integrado en *Visual Studio Code*, es ligero, consume poca memoria RAM, no necesita instalarse por separado, tiene una interfaz simple, limpia y fácil de usar, cuenta con colecciones y variables de entorno, puede hacer pruebas sin guiones, soporta código *Visual Studio Code* y maneja grandes respuestas en vista de pantalla completa.

Figura 2.45: Logo de *Thunder Client*

Su funcionamiento consiste en poder introducir contenido de entrada por el *Body* en distintos formatos, como *JSON*, *XML*, *Text*, *Formulario*, *Formulario codificado*, *Solicitud Query* y *Archivo de imagen (en binario)* y en tener la capacidad de sacar por salida: una respuesta, cabeceras, cookies, resultados y documentos.

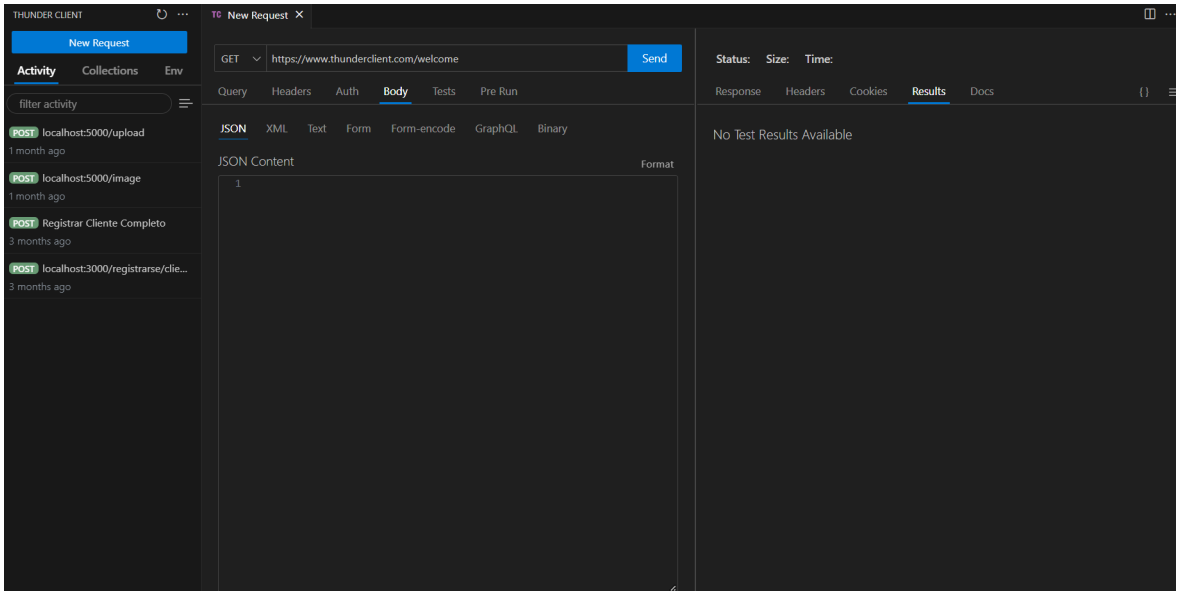


Figura 2.46: Interfaz de *Thunder Client* como simulador de llamadas a la API, sacado de la aplicación de escritorio de *Visual Studio Code* que tiene instalado el autor

2.5.4. Git y GitHub

Git[90] es un sistema de control de versiones moderno, de código abierto maduro y con un mantenimiento activo, que ayuda a rastrear, analizar, gestionar y revertir los cambios realizados en archivos, sin repercusiones si se cometen errores. Desarrollado por Linus Torvalds, el famoso creador del kernel del Sistema Operativo (S.O) Linux, en 2005. Funciona en una amplia variedad de IDEs (Entornos de Desarrollo Integrado) y de Sistemas Operativos. Su Arquitectura es distribuida, conocida como DVCS (Sistema de Control de Versiones Distribuido), donde en vez de tener un único espacio para todo el historial de versiones del software, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.

Figura 2.47: Logo de *Git*

Está diseñado teniendo en cuenta su rendimiento, su seguridad y su flexibilidad, con el objetivo de mejorarlos. En el caso del primero, quitando importancia a los nombres de los archivos a la hora de determinar su almacenamiento y el historial de versiones del árbol de los mismos, centrándose en el contenido del propio archivo. En el caso del segundo, usando como protección un algoritmo de HASH criptográficamente seguro llamado **SHA1**, para proteger y guardar el código, frente a las modificaciones accidentales y maliciosas y garantizando un historial trazable. Y en el último caso, ofreciendo compatibilidad con numerosos sistemas y protocolos para ganar eficiencia en cualquier tipo de proyecto y la capacidad de realizar varios tipos de flujos de trabajo no lineales.

Principales razones por las que se usa *Git* en este proyecto. La primera, debido a su amplia comunidad de soporte y ayuda. La segunda, por su rapidez en el ciclo de publicación. Y la tercera, por tener conexión con servicios en la nube que alojan los sistemas de control de versiones *Git* como *GitHub* y *GitLab*.

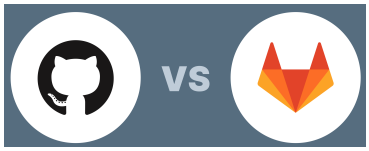


Figura 2.48: Tipos de servicios en la nube que alojan los sistemas de control de versiones *Git*, sacado de [91]

El servicio de *GitHub*[92] consiste en un servicio de alojamiento en la nube que aloja sistemas de control de versiones *Git*. Es el que ha sido elegido y usado para almacenar el código fuente del prototipo del proyecto debido a que a su autor le resulta más familiar, más cercano, más intuitivo y más sencillo de usar.

En conclusión, este servicio mencionado permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos. Incluso, mantener un seguimiento detallado de su progreso.

2.5.5. Adobe Photoshop

Adobe Photoshop[93] es un programa informático de edición de imagen desarrollado por la empresa *Adobe Systems* para dispositivos Windows y Mac Os. Es considerada la herramienta más completa de retoque fotográfico y la favorita de los profesionales del sector. Se puede aplicar para diseño gráfico, diseño web, 3D y fotografía. Destaca por su amplia gama de herramientas y funciones para editar y retocar imágenes con selecciones precisas, máscaras, filtros, efectos y transformaciones, por la posibilidad de personalizar su interfaz de usuario para adaptarla a las necesidades y preferencias del mismo, por pertenecer al grupo de la *Suite Creative Cloud* de *Adobe*, por tener compatibilidad con diferentes formatos de imagen, por su amplia comunidad de usuarios y utilización en el mercado laboral.

Es la que se ha escogido para editar todas las imágenes incluidas en el prototipo y en la memoria del Proyecto de Fin de Grado.

Figura 2.49: Logo de la tecnología software de edición de imágenes *Adobe Photoshop*

2.5.6. Adobe Premiere y Wondershare Filmora 9

Tanto **Adobe Premiere**[94] como **Wondershare Filmora 9**[95] son softwares o programas informáticos de edición de vídeo o GIFs disponibles para Mac y PC. El primero es el líder y más popular del sector para películas, TV y sitios web, donde puedes editar secuencias en cualquier formato, desde 8K hasta en RV (Realidad Virtual), hay compatibilidad con otros productos del grupo *Adobe* y la compatibilidad de formatos nativa, los sencillos flujos de trabajo de Proxy y las secuencias rápidas de *ProRes HDR* te permiten trabajar como quieras con tus medios, incluso en estaciones de trabajo móviles. Por el otro lado, el segundo es menos conocido pero no necesariamente peor, ya que es una herramienta muy potente en cuanto a formatos se refiere, permitiendo trabajar con videos HD y 4K. Además, cuenta con herramientas que lo vuelven un programa amigable y funcional, incluyendo 100 pistas de video (capas), superposiciones, filtros y transiciones.

Ambos han sido usados, porque son sencillos en su manejo, son intuitivos, cuentan con mucha documentación de tutoriales en Internet, cuyos resultados tienen calidad demostrable. Los GIFs del prototipo han sido editados en ambos programas, donde para la edición en sí se ha usado *Adobe Premiere* y para unir tomas, *Wondershare Filmora 9*.

Figura 2.50: Logos de las tecnologías de software de edición de vídeos y GIFs de *Adobe Premiere* y *Wondershare Filmora 9*

Capítulo 3

Proceso de desarrollo

El prototipo del proyecto se ha desarrollado de acuerdo al siguiente esquema:

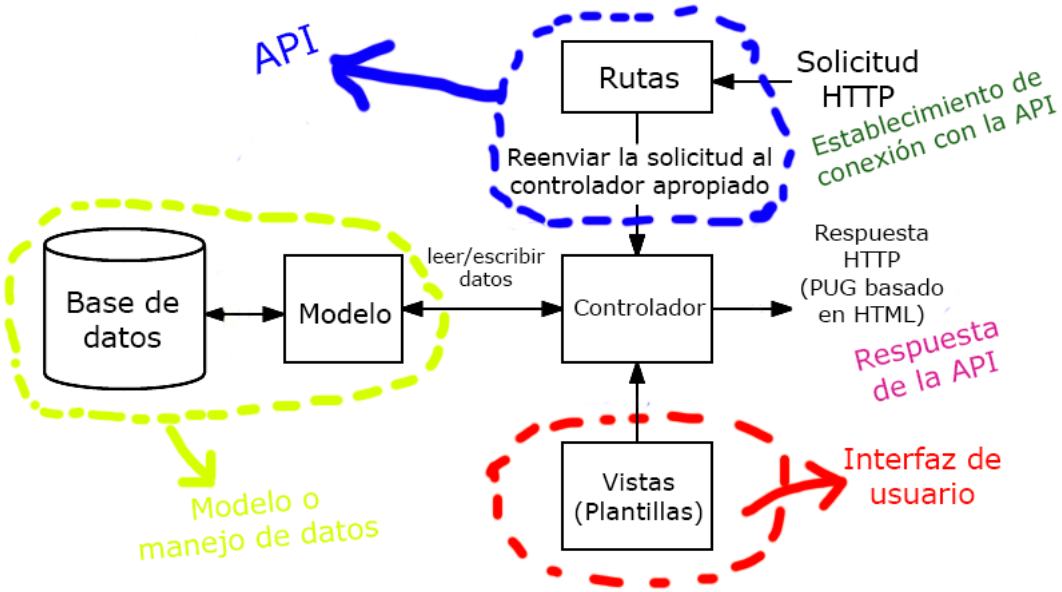


Figura 3.1: Esquema del prototipo del proyecto, anotado y editado en *Adobe Photoshop* a partir del original en [96]

Donde destacan tres capas: La *API*[97], el *Modelo o manejo de datos*[98] y la *Interfaz de usuario*[99]. La primera, conocida como *Interfaz de Programación de la Aplicación* por sus siglas, o también conocida como servidor web, es el medio encargado de manejar la aplicación web por medio de *rutas*[100] y de un *controlador*[101] y, desde donde se hace posible el establecimiento de comunicaciones entre un cliente y el propio servidor web a través de las

peticiones de solicitud y respuesta, bajo el protocolo HTTP. El controlador mencionado es un intermediario entre la capa de la API, de la que depende y, las capas del modelo o manejo de datos y de la interfaz de usuario. La segunda es el medio desde donde se maneja el conjunto de datos o información, organizados y almacenados de forma electrónica e informática en una base de datos, que la API decide manejar[1]. Y la tercera es el medio gráficamente visible de la aplicación por el cual se muestra en el navegador, donde la API es quien decide qué mostrar.

El desarrollo de cada capa se ha realizado a partir de ocho fases:

3.1. Fase 0: El servidor web o API

Es la primera fase de todas, donde se desarrolla la parte principal de la aplicación web: el servidor web o API. Su desarrollo se lleva a cabo en el entorno de tiempo de ejecución en tiempo real de *Node JS*, entorno necesario para poder generar[102] el *framework* de *Express JS* y, así, poder desarrollar ágilmente la API mediante un conjunto de módulos, librerías y/o funcionalidades ya creadas, siguiendo un orden basado en la Arquitectura MVC (Modelo, Vista y Controlador), explicada anteriormente y en consonancia con el esquema al principio de este capítulo y, a través del lenguaje de programación de *JavaScript*.

A su vez, el servidor web se divide en tres partes:

1. La raíz o *index.js*, encargada de la configuración y normalización del puerto de escucha (por defecto suele ser el 3000) donde se va a ejecutar la aplicación, bajo el protocolo HTTP y, de la depuración del código fuente y de los errores que surjan. Esta parte se enlaza con el servidor creado en la siguiente.

2. La aplicación o *app.js*, encargada de crear el propio servidor web con la tecnología de *Express*, de agrupar todos los módulos o dependencias de *Express JS* instaladas previamente en la aplicación a través del *Administrador de Paquetes de Node JS* para poder utilizarlas, de esta-

[1]Cuando se habla del manejo de datos a través de una base de datos, quiere decir que es el medio desde donde se obtiene y/o almacena un conjunto de datos o información

blecer la sesión y autenticación con la tecnología de *Express-session* para poder iniciar y cerrar sesión en el prototipo, de establecer todas y cada una de las rutas que se encargan del mapeo al objeto de aplicación, de configurar la ubicación de los archivos de las vistas estáticas o motores de plantillas de la tecnología *PUG*, de archivos externos y del archivo del diseño y apariencia de la interfaz de usuario hacia la propia API para poder ejecutarlos en esa interfaz y, de configurar las páginas de error cuando se acceda a recursos o páginas de la aplicación inexistentes. Esta parte, a su vez, se enlaza con la siguiente.

3. Las rutas o *routes*, encargadas de manejar la aplicación junto a un controlador y de establecer las comunicaciones entre un cliente y el propio servidor web o la API a través de las peticiones de solicitud HTTP y respuesta HTTP. Las primeras peticiones son aquéllas que establecen la conexión con la API para que la aplicación web empiece a funcionar. Y las segundas peticiones son aquéllas que sacan por la interfaz de usuario la respuesta del servidor web.

3.2. Fase 1: Las rutas del servidor web o API

Una ruta es un patrón de dirección *URL*[2][103] compuesto de una parte de la misma llamado *recurso*[3][104] y del método de petición HTTP que maneja la acción para acceder a este último. Cada ruta se asigna a un controlador y puede ser de dos tipos: según el mapeo al objeto de aplicación y según el mapeo a un objeto de rutas. El primero se establece por el propio servidor web a través de la parte ya mencionada en la fase 0 de este capítulo y, es quien agrupa, unifica y dirige hacia los tipos de agrupaciones creadas. El segundo son esas agrupaciones que crea el primero, establecidas por el módulo de *Router()* de *Express JS* y dividiéndose en cuatro tipos de rutas hacia recursos específicos:

Rutas con el método GET: son rutas creadas para operar bajo el método indicado, cuya función consiste en obtener o acceder a recursos o páginas de la aplicación.

[2]Una URL es un tipo de identificador uniforme de recursos que accede a información y está compuesto por diferentes partes: el protocolo, las doble barras, el servidor, el puerto, el recurso y los parámetros o valores literales de una página

[3]Un recurso es un localizador uniforme por donde se accede a la información requerida para identificar y unificar cada página de una interfaz de usuario a la que se quiere acceder

Rutas con el método POST: son rutas creadas para operar bajo el método indicado, cuya función consiste en manejar datos de recursos o páginas de la aplicación.

Rutas con el método PATCH: son rutas creadas para operar bajo el método indicado, cuya función consiste en actualizar datos de recursos o páginas de la aplicación.

Rutas con el método DELETE: son rutas creadas para operar bajo el método indicado, cuya función consiste en borrar datos de recursos o páginas de la aplicación.

```
//--##########################################################################################--//
//--################################## CONEXIÓN CON LAS RUTAS ################################--//
//--##########################################################################################--//
var rutasGet = require('./routes/GET.routes.js');
var rutasPost = require('./routes/POST.routes.js');
var rutasPatch = require('./routes/PATCH.routes.js');
var rutasDelete = require('./routes/DELETE.routes.js');

madservices.use(rutasGet, rutasPost, rutasPatch, rutasDelete);
//##########################################################################################--//
```

Figura 3.2: Ejemplo práctico de mapeo a un objeto de aplicación agrupando en grupos, sacado del código fuente del prototipo del proyecto y editado en *Adobe Photoshop*

Figura 3.3: Ejemplo práctico de mapeo a un objeto de rutas hacia recursos específicos, sacado del código fuente del prototipo del proyecto y editado en *Adobe Photoshop*

Aunque realmente, en el *framework* de *Express JS* sólo funcionan los tipos de métodos de GET y POST y para que funcionen los cuatro, hay que instalar y utilizar la tecnología mencionada de *method-override*.

3.3. Fase 2: El controlador de la aplicación web

El controlador[105] es la capa o intermediario del prototipo del proyecto, asignado a una ruta y dependiente de la API, que trabaja en el lenguaje de programación de *JavaScript* y donde se realiza el procesamiento, la lógica y la salida de la respuesta HTTP con su código de respuesta, renderización a su interfaz de usuario y, en ciertas ocasiones, con una notificación o alerta emergente sonora, definida por la tecnología mencionada de *node-notifier* y destacando cuatro tipos: alerta de proceso exitoso, alerta de proceso incorrecto, alerta de proceso no modificado y alerta de proceso de búsqueda. Los datos dinámicos y la información de la respuesta HTTP, se obtienen del modelo o manejo de datos y de las vistas estáticas que crean y manejan la interfaz de usuario, respectivamente.

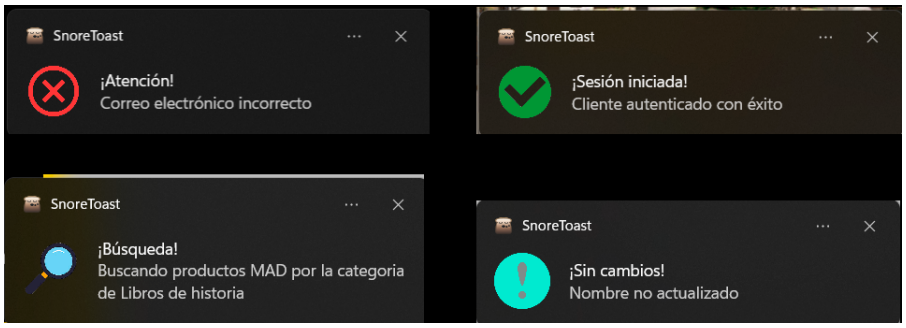


Figura 3.4: Tipos de notificaciones o alertas emergentes sonoras en el prototipo, sacado del código fuente del mismo y editado en *Adobe Photoshop*

Su división se lleva a cabo por los modos en que el prototipo del proyecto trabaja, donde se establece la lógica de programación, la llamada para obtener y/o almacenar datos y la verificación o validación de los mismos por medio de las siguientes tecnologías mencionadas: *Axios*, *Bcrypt*, *Card-validator*, *Country-list*, *Country-list-spanish*, *Postcode-validator* y *Validator*. El caso de *Axios* en concreto sirve para conectar con la aplicación externa de *GeoNames*[106] y verificar la localización del cliente a través de su código postal y de su código de país.

En este caso, los modos del controlador se dividen en cuatro:

1. El controlador del modo clientes, encargado de la programación verificada de su tipo de registro e inicio de sesión, de la búsqueda de productos propios de la aplicación con posibi-

lidad de expandirlos y verlos de forma completa, de la búsqueda de las empresas registradas con acceso a los productos y/o servicios ofertados en la interfaz diseñada por las mismas, del ingreso de esos productos al carrito de la compra para comprarlos, del ingreso, actualización y/o eliminación de la tarjeta bancaria necesaria para pagar la compra, de la actualización de sus datos y su localización, de su baja como cliente de la aplicación, de la eliminación de productos del carrito de la compra sin interés a comprarlos y del establecimiento del sistema de pago de la compra de productos y/o servicios del carrito.

2. El controlador del modo empresas, encargado de la programación verificada de su tipo de registro e inicio de sesión, de la búsqueda de su propia empresa u otras de su competencia para comprobar los productos y/o servicios ofertados en la interfaz diseñada por ellas mismas, de la actualización de sus datos, de su baja como empresa de la aplicación y de la actualización y/o eliminación de su descripción, redes sociales, logo, contacto y página web como *marketing* empresarial visible en su interfaz diseñada.

3. El controlador del modo miembros, encargado de la programación verificada de su tipo de registro e inicio de sesión autorizado previamente, de la búsqueda de productos propios de la aplicación con posibilidad de expandirlos y verlos de forma completa, de la búsqueda de las empresas registradas con acceso a los productos y/o servicios ofertados en la interfaz diseñada por las mismas, del ingreso, actualización y/o eliminación de los productos propios de la aplicación y de hasta un máximo de diez imágenes en cada uno de ellos, de la actualización de sus datos, de su baja como miembro de la aplicación y de la baja autorizada de clientes y empresas registradas.

4. El controlador del modo general para usuarios sin registro, encargado de la programación verificada de la búsqueda de productos propios de la aplicación con posibilidad de expandirlos y verlos de forma completa y de la búsqueda de las empresas registradas con acceso a los productos y/o servicios ofertados en la interfaz diseñada por las mismas.

Para concluir, el controlador también lleva a cabo el proceso y programación verificada de la subida de imágenes a la base de datos. Para ello, se realiza a través de la tecnología de *Mul-*

ter, donde recibe la imagen de una en una introducida por la interfaz de usuario y la almacena localmente con un determinado nombre por medio de las tecnologías de *Util*, *Path* y *Fs*, para después, editarla con la tecnología de *Sharp* y almacenarla en la base de datos con la tecnología de *MySQL*, para finalmente, borrarla localmente con *Util*, *Path* y *Fs* nuevamente. Todas esas tecnologías son mencionadas en el *Estado del arte, de la técnica y sistemas relacionados*.

```
const storage = multer.diskStorage({
    destination: (req, file, callback) => {
        callback(null, './archivos');
    },
    filename: (req, file, callback) => {
        const extension = file.originalname.split('.').pop();
        callback(null, `${Date.now()}.${extension}`);
    }
});
const upload = multer({ storage: storage });
const cargas = multer({ storage: storage });
```

Figura 3.5: Funcionamiento de la tecnología de *Multer*, sacado del código fuente del prototipo del proyecto y editado en *Adobe Photoshop*

3.4. Fase 3: El modelo o manejo de datos de la aplicación web

El modelo[107] es la capa donde se trabaja con los datos de forma dinámica, obteniendo y almacenando información en la base de datos, a través de la tecnología de *MySQL* y con la posibilidad de ver la base de datos en una interfaz gráfica gracias a la tecnología de *phpMyAdmin*.

Su división también se lleva a cabo por los mismos modos del controlador de la aplicación web, pero en vez de establecer la lógica de programación y la validación o verificación de los datos, se encarga del manejo de ellos, obteniéndolos y almacenándolos en la base de datos. Además, ese manejo se realiza a partir de diferentes usuarios de acceso a la base de datos:

1. El usuario cliente para acceder por el modelo del modo clientes y del modo general para usuarios sin registro, en cuyo caso, sólo cuenta con privilegios o permisos de manejo de datos.

2. El usuario empresa para acceder por el modelo del modo empresas, en cuyo caso, también sólo cuenta con privilegios o permisos de manejo de datos.

3. El usuario administrador, miembro o *root* para acceder por el modelo del modo miembros, en cuyo caso, cuenta con todos los privilegios o permisos de acceso.

Por otro lado, para llevar a cabo ese manejo de datos se utilizan cuatro tipos de instrucciones o funciones del lenguaje *SQL*: *SELECT*, *INSERT*, *UPDATE* y *DELETE*, donde la primera es para consultar la información requerida de la base de datos, la segunda es para insertar nueva información en la base de datos, la tercera es para actualizar con otra información en la base de datos y la cuarta es para borrar información de la base de datos.

Para concluir, hay que saber que para que funcione el modelo o manejo de datos de la aplicación web, es necesario instalar *MySQL*. Y si aparte se quiere mostrar la base de datos en una interfaz gráfica, es necesario instalar también *Apache* y *PHP* para poder mostrarla en esa interfaz de *phpMyAdmin*. Su procedimiento es el siguiente:

[108]En primer lugar, hay que instalar el servidor de *Apache* para poder conectarnos a través del puerto de aplicación a *phpMyAdmin*. Para ello, hay que dirigirse a *The Apache HTTP Server Project*[109], ahí dentro buscar el apartado de *Essentials*, después ir a *Download!*[110], luego hay que buscar *Apache HTTP Server 2.4.56 (httpd): 2.4.56 is the latest available version* e ir a *Files for Microsoft Windows*[111] porque la demo se ha trabajado en el Sistema Operativo (S.O) de Windows, a continuación hay que buscar *Downloading Apache for Windows* e ir a *ApacheHaus*[112] y entonces hay que buscar *Apache 2.4.x OpenSSL 1.1.1 VS17* para descargar la versión más reciente, donde en este caso fue *Apache 2.4.55 x64*[113] porque el sistema es de 64 bits con un procesador basado en x64, por lo que al pinchar en su logo, se obtendrá el archivo ejecutable.

Es importante saber, que para poder seguir con la instalación y evitar errores, hay que instalar los Paquetes Redistribuibles[114] para Visual Studio.

Una vez hecho todo, le sigue: descomprimir el archivo ZIP de *Apache 2.4.55 x64* en el directorio de descargas del computador, entrar en la carpeta descomprimida, cortar la carpeta de *Apache24*, pegarla en el directorio o disco local C, abrir la consola de comandos como administrador en la carpeta de *bin* dentro de *Apache24*, introducir *httpd -k install* para instalar httpd, permitir el acceso del *Firewall*, comprobar que *Apache24* está en ejecución en los *Servicios de Windows* y acceder al navegador para comprobar que *Apache* funciona a través de la dirección

web local de *localhost*.

[115]En segundo lugar, hay que instalar *PHP* en *Apache*, lenguaje base de *phpMyAdmin*. Para ello, hay que buscar *PHP* en *Google* y dirigirse a donde dice: *¿Qué es PHP? - Manual*[116], después ir a la parte superior donde dice: *Downloads*[117] y descargar la última versión para *Windows*[118] y luego descargar el archivo ZIP de *VS16 x64 Thread Safe (fecha y hora actualizada)*[119].

Una vez hecho todo, le sigue: crear una carpeta llamada *php8* en la carpeta descargas del computador, introducir el archivo ZIP de *php-8.2.7-Win32-vs16-x64* dentro, descomprimir dicho archivo, eliminar el ZIP y quedarse con el contenido de dentro de la carpeta descomprimida, mover la carpeta de *php8* al directorio o disco local C, ir a *Editar las variables de entorno del sistema/variables de entorno/variables del sistema/Path/Editar.../Nuevo/introducir la ruta donde se encuentra el directorio de php8/Aceptar* para poder tener la versión de *PHP* en el computador, pausar *Apache24* en los *Servicios de Windows* para poder integrar *PHP* en *Apache*, abrir el archivo de *httpd.conf* en *C:/Apache24/conf*, buscar dentro del archivo donde ponga *LoadModule foo module modules mod foo so*, debajo de ello hay que poner:

```
ServerName localhost
LoadModule php_module C:/Php8/php8apache2_4.dll
AddHandler application/x-httpd-php .php
PHPIniDir C:/Php8
```

Figura 3.6: Instrucciones 1 a poner en la parte de LoadModule, sacado del archivo de *httpd.conf* en *C:/Apache24/conf*

En ese mismo archivo, buscar *index.html* y añadir a continuación suya *index.php*:

```
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>
```

Figura 3.7: Instrucciones 2 a poner en la parte de LoadModule, sacado del archivo de *httpd.conf* en *C:/Apache24/conf*

Al acabar, hay que guardar el archivo de *httpd.conf* en *C:/Apache24/conf*, ir a *Servicios de Windows*, iniciar de nuevo *Apache24* y cambiar el archivo de *index.html* a *index.php* en *C/Apache24/htdocs*.

[120]En tercer lugar, hay que instalar *MySQL*. Para ello, hay que buscar *mysql* en *Google* y dirigirse a donde dice *MySQL* que es el enlace oficial[121], después ir a *Descargas*[122], acceder a *MySQL Community (GPL) Downloads*[123], acceder a *MySQL Installer for Windows*[124], descargar el archivo más pesado (que es el completo)[125] y ejecutarlo como completo en el directorio de descargas del computador.

Es importante saber, que para poder seguir con la instalación y que *MySQL* funcione correctamente, hay que instalar el conector de *Python*[126] y *MySQL para Visual Studio*[127]

Una vez hecho todo, le sigue: ejecutar todos y cada uno de los productos de *MySQL*, introducir una contraseña fuerte, continuar aceptando, configurar el *MySQL Router* y el *Samples and Examples*, escribir en la consola de comandos *mysql -u root -p contraseña introducida previamente* y testear la conexión con el servidor a través de dicho usuario y la contraseña.

[128]Por último, hay que instalar *phpMyAdmin*, integrándolo en *Apache*. Para ello, hay que buscar *phpMyAdmin* en *Google*, dirigirse a donde dice *phpMyAdmin* que es el enlace oficial[129] y después descargar el ejecutable que dice *Descargar y la versión última*[130].

Una vez hecho todo, le sigue: descomprimir el archivo ZIP en el directorio descargas del computador, renombrar la carpeta descomprimida a *phpMyAdmin*, eliminar el ZIP, mover dicha carpeta a *C/Apache24/htdocs*, ir a *C/Php8*, renombrar el archivo *php.ini-development* a *php.ini*, entrar en este último archivo, descomentar las siguientes instrucciones: *extension=curl*, *extension=ftp*, *extension=gettext*, *extension=gmp*, *extension=intl*, *extension=mysqli* y *extension=openssl*, añadir al inicio del contenido del archivo *php.ini* las líneas siguientes: *extension=php mbstring.dll*, *extension=php mysqli.dll*, *session.save path = Ruta a carpeta temp en Php8"* y *extension dir = Ruta a carpeta ext en Php8"*, ir a *Servicios de Windows*, detener *Apache24*, esperar unos segundos, volver a iniciar *Apache24* y ya se podría acceder a *phpMyAdmin* en el navegador a través de *localhost*.

La base de datos creada se llama *madservicesdb* y cuenta con ocho tablas: para almacenar

los clientes, para almacenar las empresas, para almacenar los miembros, para almacenar los productos y/o servicios comprados, para almacenar los productos y/o servicios en el carrito de la compra, para almacenar los productos propios de la plataforma que hay, para almacenar las tarjetas bancarias que hay y para almacenar las imágenes de cada producto que hay.

Figura 3.8: Interfaz gráfica de la base de datos del prototipo del proyecto en *phpMyAdmin*

3.5. Fase 4: Configuración de la interfaz de usuario hacia el servidor web o API

La configuración de la interfaz de usuario hacia el servidor web o API, como ya se ha mencionado al principio de este capítulo, se lleva a cabo configurando la ubicación de los archivos de las vistas estáticas o motores de plantillas de la tecnología *PUG*, de los archivos externos y del archivo del diseño y apariencia de la interfaz de usuario hacia la propia API, para así poder ejecutarlos en esa interfaz. Véase la imagen siguiente de su configuración y funcionamiento:

```
//-- Conectando con la interfaz de usuario o front-end.
madservices.use(servidor.static(path.join(__dirname, 'public')));
madservices.set('views', path.join(__dirname, 'views'));
madservices.set('view engine', 'pug');
//#################################################################
```

Figura 3.9: Código fuente de la configuración de la interfaz de usuario hacia el servidor web o API, sacado del código fuente del prototipo del proyecto

La imagen significa que las vistas estáticas están preparadas para la tecnología de *PUG* únicamente. En ella se establece la dirección desde su ruta hacia la API o servidor web para poder renderizar el contenido de las vistas estáticas. Además, se establece la dirección desde la ruta de *public*, donde están los archivos externos y el archivo del diseño y apariencia de la interfaz de usuario, hacia la API o servidor web también, para poder establecer y mejorar la apariencia de la propia interfaz.

3.6. Fase 5: Vistas Estáticas y su diseño y apariencia

```
html(lang="es")
  //-- Pestaña del Navegador del Comercio Electrónico.
  head
    //-- Codificación del Comercio Electrónico.
    meta(charset="utf-8")
    //-- Configuración del Favicon genérico del Comercio Electrónico.
    link(rel="icon" sizes='16x16 32x32 48x48 64x64' href="/favicon.ico")
    //-- Configuración y ajustes generales del Comercio Electrónico.
    meta(name="viewport" content="width=device-width, user-scalable=no, initial-scale=1, minimum-scale=1.0")
    //-- Configuración de la descripción del Comercio Electrónico.
    meta(name="description" content="El E-Commerce de los E-Commerces")
    //-- Otras configuraciones.
    meta(http-equiv="X-UA-Compatible" content="IE=edge")
    //-- Configuración del cambio de imágenes.
    script(src="/javascripts/cambioImagenes.js" defer)
    //-- Configuración de los Canvas 2D.
    script(src="/javascripts/marcaMAD.js" defer)
    script(src="/javascripts/multiMarca.js" defer)
    //-- Configuración de los iconos.
    script(src="https://kit.fontawesome.com/d232ed2b98.js" crossorigin="anonymous")
    //-- Configuración de las fuentes de escritura del Comercio Electrónico.
    link(rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" integrity="
    link(rel='stylesheet' href='/stylesheets/style.css')
    //-- Título de la Pestaña del Navegador del Comercio Electrónico.
    block title
  body
    block body
```

Figura 3.10: Base de todas las vistas estáticas del prototipo del proyecto, sacado del código fuente del mismo

Las vistas estáticas pertenecen a la capa de la interfaz de usuario, que trabaja en la tecnología de *PUG* con su estructura, sintaxis y código, para generar la propia interfaz de la aplicación.

Su división se lleva a cabo por medio de una base general, donde se establece su lenguaje *PUG* basado en *HTML* simplificado, la cabecera del código y las variables del título y del cuerpo reutilizables en las diferentes páginas de las vistas estáticas. Esas páginas se llevan a cabo por los mismos modos del controlador y del modelo de la aplicación web, pero en vez de establecer la lógica de programación, la validación o verificación de los datos y el manejo de ellos, se encargan de estructurar y desarrollar la sintaxis y el código de la interfaz de usuario a ejecutar en el navegador.

Por otro lado, el diseño y apariencia de la propia interfaz generada por las vistas estáticas trabaja en la tecnología de *CSS* con su estructura, sintaxis y código, para dar estilos, diseño y apariencia a esa interfaz.

Su uso consiste en un archivo *CSS* común dividido en secciones, que representan el diseño, estilo y apariencia de todas y cada una de las páginas de las vistas estáticas de *PUG* de la apli-

cación web.

Por último, las vistas estáticas cuentan con un archivo cuyo código consiste en mostrar una página de error cuando se acceda a un recurso de la interfaz de usuario inexistente (error 404), o cuando haya un error de servidor (error 500). Y también cuentan con iconos de la aplicación web externa *Font Awesome*[131], repartidos por toda la plataforma y cuyo fin es utilizarse como botones para pasar de una página a otra de la aplicación, sin necesidad de utilizar los botones de atrás/adelante del navegador web.

3.7. Fase 6: Archivos externos

Los archivos externos de la aplicación web se dividen en tres tipos:

1. Archivos externos que trabajan en el controlador de la API: el generador de ID o identificadores aleatorios y la consulta de esos IDs. El primero es una función que genera un ID aleatoriamente en el momento que el cliente, la empresa y el miembro se registran en la plataforma, ya que todos los usuarios de la aplicación deben tener un ID aleatorio, único e irrepetible. Los IDs también se almacenan en la base de datos, al igual que el resto de campos de cada tipo de registro independiente entre sí. La última es una función que consulta en la base de datos de los clientes, las empresas y los miembros el ID generado por la primera función, con el objetivo de no repetirlo.

```
function generarIDrandom() {

    //-- Variable que almacena el ID aleatorio.
    const id = parseInt(Math.random() * 1000);

    //-- Sacarlo por salida.
    return id;
}
//##############################################

//########################################### P
module.exports = generarIDrandom;
```

```
consultaID
(
    idCliente,
    (idExiste) => {
        while(idExiste) {
            idCliente = generarIDrandom() * 2;
            consultaID(idCliente, (idExiste) => {
                idExiste = idExiste;
            });
        }
    }
);
```

Figura 3.11: A la izquierda, la función generar ID aleatoriamente y, a la derecha, la función consultar ID aleatorio en clientes, sacado del código fuente del prototipo del proyecto

2. Archivos externos pertenecientes a la carpeta mencionada de *public* del prototipo de la aplicación web: el *Favicon*, los *GIFs*, las imágenes, los *PDFs* y los *JavaScript* externos. El

primero es un logo o icono que se muestra en la pestaña del navegador donde se ejecuta la interfaz de usuario. Los segundos son tres animaciones en bucle: una de ellas es la animación del logo de los clientes de la aplicación web establecida en el registro de los mismos, la otra es la misma animación pero en otro color y para el registro de las empresas y la última es la animación más larga y establecida en el inicio de la aplicación o plataforma, cuyo significado es *un lugar o plataforma donde agrupar las diferentes marcas*, realizado por medio de las tecnologías mencionadas de *Adobe Premiere* y *Wondershare Filmora 9*. Las terceras son representaciones visuales repartidas y utilizadas por todo el *Marketplace* y editadas por medio de la tecnología mencionada de *Adobe Photoshop*. Los cuartos son archivos o documentos oficiales y burocráticos de la sección de los términos y condiciones que toda aplicación web debe tener. Y los quintos son tres archivos, donde dos de ellos son utilizados para crear los *Canvas*[132] del movimiento de los logos para acceder a las dos secciones de productos y/o servicios de la plataforma y, el otro archivo es utilizado para crear la acción de pasar de imagen[133] en cada expansión de los productos propios de la plataforma.

3. Archivos externos generales del prototipo de la aplicación web: la *Licencia MIT* de la plataforma para que sea oficial y con posibilidad de reutilizar el código fuente, pero bajo la firma del propietario. El *gitignore* para ignorar archivos que no se quieran guardar en la nube por protección o exceso de tamaño de memoria de los mismos. El entorno de variables *ENV* para declarar variables con el fin de que en el código fuente del prototipo se muestren ellas en vez de su contenido, por protección de datos y seguridad. Las dependencias de los módulos del *Administrador de Paquetes de Node JS*: *node modules*, que hacen funcionar a cada una de las dependencias de *Express JS* mencionadas en el *Estado del arte, de la técnica y sistemas relacionados*. Y un directorio de archivos, donde almacenar las imágenes que se cargan al ingresar y/o actualizar nuevos productos y/o productos ya existentes, respectivamente.

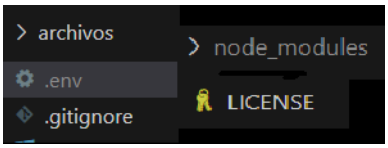


Figura 3.12: Archivos externos generales del prototipo de la aplicación web, sacados del código fuente del mismo

3.8. Fase 7: Despliegue del prototipo en la nube

El *despliegue*[134] de la aplicación web es la parte final del desarrollo del prototipo, que consiste en hacerla pública para que cualquier potencial usuario pueda acceder a ella, independientemente de donde se encuentre y mientras tenga acceso a Internet. Se puede llevar a cabo a través de dos tipos: creando un servidor físico con conexión a Internet que esté presente en la propia sede del negocio, o buscando soluciones en la nube con un servidor *Cloud*, como *Heroku*, *AWS* o *Azure*. La primera opción es muy tediosa y require de mucho tiempo, ya que el servidor *Cloud* se tiene que crear desde cero. La segunda opción parece más asequible, pero también requiere de un orden de configuración bastante arduo, incluso en muchos casos, suelen ser servidores *Cloud* de pago. En esta última opción hay que tener en cuenta previamente el entorno de tiempo de ejecución en tiempo real donde se ha desarrollado la aplicación web.

Figura 3.13: Logo de un Servidor *Cloud*

Para el caso del prototipo de este proyecto, sería muy interesante la primera opción de crear el servidor *Cloud* desde cero, pero llevaría mucho tiempo debido a su alta complejidad y falta de experiencia profesional en el despliegue de aplicaciones web por parte del autor. Por tanto, la única opción viable para desplegar el prototipo en la nube era la segunda. El problema estuvo en que muchos de los servidores *Cloud* eran de pago o tenían opción a pago y, los que había gratuitos, eran muy poco intuitivos y muy poco convincentes para el autor, tenían una alta probabilidad de errores en el proceso y requerían de mucho tiempo para conocer su tecnología. Para ello, en vez de llevar a cabo el despliegue de la aplicación, el objetivo era presentarlo en local, pero con una característica más favorable. Ésta *última*[135] consiste en crear un archivo *.bat* y un archivo *.vbs*, ambos con el nombre del puerto de escucha de la aplicación, para que cuando el computador del autor se encienda, el código fuente del prototipo del proyecto se ejecute automáticamente en él y, así, que el propio autor no tenga que abrir el lugar donde se encuentra ese código y ejecutarlo manualmente, sino que sólo necesite escribir la URL de local en el navegador y pulsar la tecla *enter* para acceder al prototipo que está corriendo.

Para ejecutarlo automáticamente al encender el computador, hay que ir a *Ejecutar* de *Windows* y escribir: *shell:startup*, donde se abrirá una carpeta o directorio donde copiar el archivo *.vbs*, ya que el computador al encenderse siempre ejecuta esa carpeta o directorio automáticamente.

En conclusión, la gran ventaja de la característica explicada es que ahorra mucho tiempo a la hora de ejecutar la aplicación en el navegador.

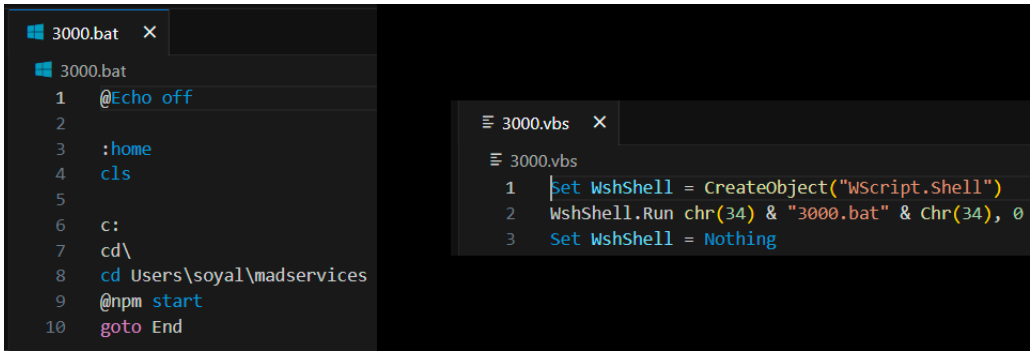


Figura 3.14: A la izquierda, el archivo BAT que compila las instrucciones a ejecutarse en la aplicación al encender el computador y, a la derecha, el archivo VBS que lo ejecuta, sacados ambos del código del prototipo del proyecto

Capítulo 4

Resultados

Los *resultados*[136] del prototipo del proyecto son aquéllos que definen la materialización que se ha llevado a cabo en él. Es decir, son las consecuencias de su desarrollo teórico y práctico, obteniendo el funcionamiento de su interfaz final.

4.1. Funcionamiento de la interfaz final

El funcionamiento de la interfaz final consiste en navegar por el prototipo a través de cuatro modos: el modo usuario sin registro, el modo usuario cliente, el modo usuario empresa y el modo usuario miembro.

El modo usuario sin registro consiste en navegar por el prototipo para ver los productos propios de la plataforma o para ver los productos y/o servicios del resto de empresas accediendo a la interfaz diseñada por ellas mismas, pero sin poder añadir al carrito de la compra ni comprar ni pagar el producto o productos elegidos, ya que para eso se debe estar registrado como cliente. Para el caso de los productos propios de la plataforma, se pueden ver todos los que hay, con la posibilidad de expandir cada uno de los productos para verlos completos con su descripción, con más imágenes aparte de su portada y con más características suyas en cuestión. Y para el otro caso, se pueden ver todos los tipos de empresas que hay por colores, con la posibilidad de acceder a ellas para ver su interfaz diseñada, donde se encuentran los productos y/o servicios ofertados por ellas mismas. Ambos tipos de productos cuentan con una búsqueda de productos o empresas a medida, que consisten en filtrar el objetivo a buscar a partir de una serie de carac-

terísticas preestablecidas. Éstas son: el filtrado de productos propios de la plataforma según el nombre del producto a buscar, el tipo al que pertenece y/o su precio de venta y, el filtrado de las empresas registradas en la plataforma según el nombre de la empresa a buscar y/o el tipo al que pertenece diferenciado por colores.

Otra característica del modo usuario sin registro es la posibilidad de acceder a registrarse o iniciar sesión en la plataforma como cliente, empresa o miembro, donde para acceder a éste último es requisito indispensable estar autorizado. En cualquier tipo de registro, se generará aleatoriamente y de forma única, un ID (identificador) para clientes, empresas y miembros sin mostrar el proceso al usuario. Para registrarse como cliente, es necesario introducir un nombre, dos apellidos, la elección de un género de tres establecidos, un correo electrónico válido, una contraseña válida a repetir por segunda vez y cuyos requisitos mínimos son la introducción de mayúsculas, minúsculas, diez caracteres como mínimo, números y caracteres especiales y, la introducción de su localización para poder llevar a cabo el envío de la compra. Ésta última consiste en introducir el país válido donde viva, con un código postal asociado a él, la región y la población asociadas a ese código postal y la dirección de su domicilio en cuestión. Para registrarse como empresa, es necesario introducir el nombre de la empresa en cuestión, su código CIF/NIF[1][138] válido, el tipo al que pertenece, donde de momento sólo hay cuatro tipos: comercio, peluquería, restaurante y academia y, el correo electrónico y contraseña repetida válidos necesarios en todo registro. Para registrarse como miembro, como ya se ha mencionado, es necesario estar autorizado y se debe introducir el nombre junto a sus apellidos del miembro en cuestión, el tipo de departamento al que pertenece dentro de la plataforma, la elección de un género de tres establecidos, un correo electrónico válido y una contraseña válida a repetir por segunda vez y cuyos requisitos mínimos son los mismos que los ya descritos anteriormente. La validez del correo electrónico a introducir en los tres tipos de registro se rige por medio del estándar RFC 5322[137], cuyo formato debe seguir la estructura de: primero, el nombre de usuario sin puntos ni guiones al principio y al final, segundo, el símbolo @ y tercero, un dominio con su extensión. Y para iniciar sesión tanto como cliente, como empresa o como miembro, es necesario introducir un correo electrónico registrado en su respectiva base de datos y la con-

[1]El código CIF/NIF es un código de identificación fiscal empleado para identificar a las empresas españolas o personas jurídicas que están registradas oficialmente

traseña asociada a ese correo electrónico.

Para concluir este modo, es posible acceder a una serie de secciones extra: donde ver el contacto de la plataforma, donde acceder a la información sobre ella y a la intrahistoria de su surgimiento, donde enviar el Currículum Vitae del propio usuario sin registro en cuestión, donde conocer y ver los términos y condiciones de la plataforma, con la posibilidad de descargar sus documentos burocráticos y, la opción de acceder a sus redes sociales en una pestaña del navegador web externa.

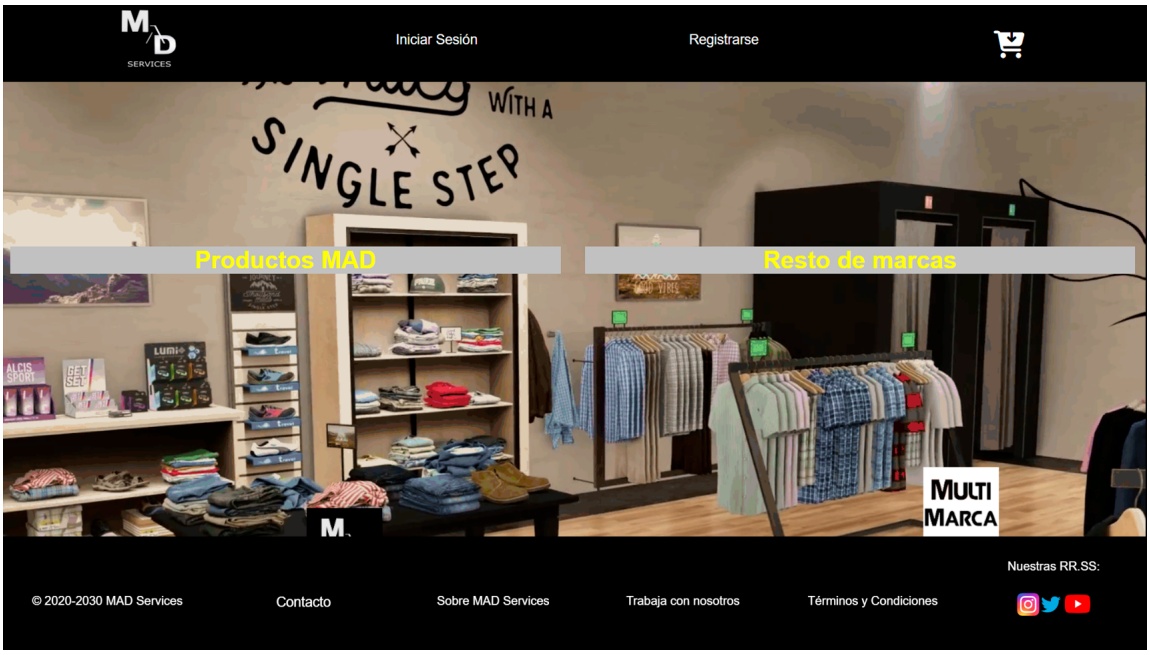


Figura 4.1: Pantalla inicial del modo usuario sin registro, sacado del prototipo de la interfaz final

El modo usuario cliente consiste en navegar por el prototipo una vez se ha iniciado sesión en su respectivo modo, cuyo fin es comprar de forma online productos y/o servicios que hay en la plataforma, son de su interés, elegidos previamente y añadidos al carrito de la compra. Su navegación da la posibilidad de ver los productos propios de la plataforma, con la opción de añadir el producto deseado al carrito las veces que quiera y de acuerdo a su cantidad y, con la opción de expandir cada uno de los productos para verlos completos con su descripción, con más imágenes aparte de su portada y con más características suyas en cuestión. También da la posibilidad de ver los productos y/o servicios del resto de empresas accediendo a la interfaz

diseñada por ellas mismas. Ambas posibilidades cuentan con una búsqueda de productos o empresas a medida, tal y como se describe anteriormente.

Otra característica del modo usuario cliente es el proceso de compra de productos y/o servicios por medio del acceso al carrito de la compra, donde puede tanto añadir como eliminar los productos que quiera, pasando a comprar todo lo que haya en ese momento en él. Para proceder a pagar, es necesario verificar la localización, la cantidad y el precio final a pagar e introducir el método de pago, con la opción de guardar la tarjeta bancaria en la base de datos o no hacerlo. Además, el cliente puede acceder a su perfil para ver sus datos personales y su localización utilizada en el proceso de envío de la compra, para actualizar ambos, para ingresar, actualizar, eliminar y ver su tarjeta bancaria utilizada en las compras, para darse de baja de la plataforma y para ver todas sus comprar realizadas.

Para concluir este modo, es posible acceder a una serie de secciones extra, que coinciden con las del usuario sin registro en cuestión, pero con la diferencia de que este modo puede cerrar su sesión cuando quiera.

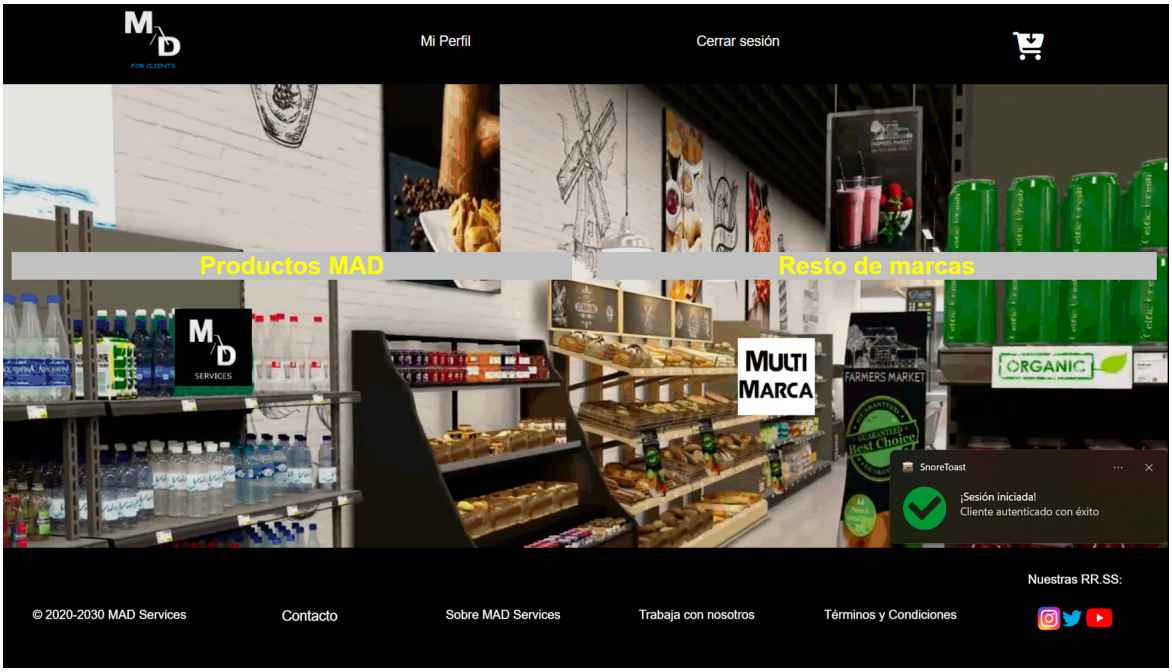


Figura 4.2: Pantalla inicial del modo usuario cliente, sacado del prototipo de la interfaz final

El modo usuario empresa consiste en navegar por el prototipo una vez se ha iniciado sesión

en su respectivo modo, cuyo fin es desarrollar su propia interfaz donde ofertar los productos y/o servicios que quiera para que los clientes puedan acceder a ella. Su navegación da la posibilidad de ver todos los tipos de empresa que hay con acceso a la interfaz diseñada por ellas mismas para ver sus productos y/o servicios. Esta posibilidad cuenta con una búsqueda de empresas a medida, tal y como se describe en el modo usuario sin registro.

Otra característica del modo usuario empresa es su opción de acceso a su perfil o interfaz para ver y actualizar los datos de su empresa, para darse de baja de la plataforma y para ingresar, ver, actualizar y eliminar la descripción de la empresa, su contacto, su logo y sus accesos a las redes sociales y a su página web que se muestran en su interfaz diseñada y dirigida a los clientes.

Para concluir este modo, es posible acceder a una serie de secciones extra, que coinciden con las del usuario sin registro y las del usuario cliente en cuestión, pero con la diferencia de que en este modo no hay ninguna sección para enviar el Currículum Vitae. Lo sí puede llevar a cabo es su cierre de sesión cuando quiera, al igual que el usuario cliente.

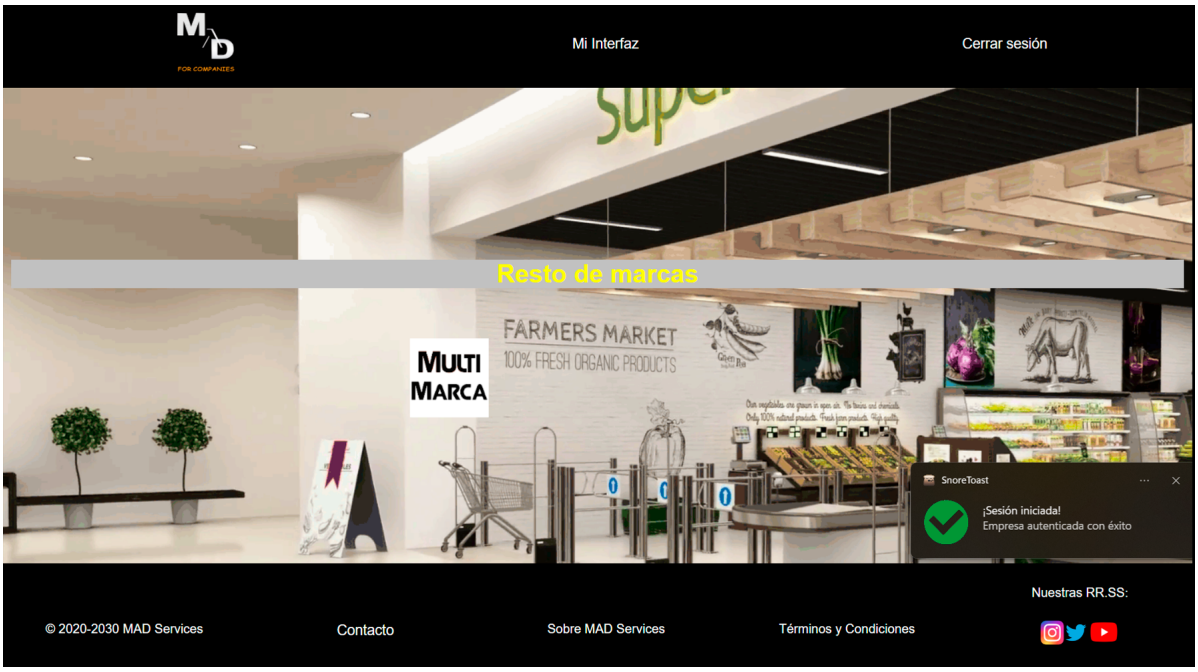


Figura 4.3: Pantalla inicial del modo usuario empresa, sacado del prototipo de la interfaz final

El modo usuario miembro consiste en navegar por el prototipo una vez se ha iniciado sesión en su respectivo modo, cuyo fin es controlar que todo funcione y se vea correctamente y con-

trolar el número de usuarios que hay en la plataforma. Su navegación da la posibilidad de ver los productos propios de la plataforma, con la opción de expandir cada uno de ellos para verlos completos con su descripción, con más imágenes aparte de su portada y con más características suyas en cuestión. También da la posibilidad de ver los productos y/o servicios del resto de empresas accediendo a la interfaz diseñada por ellas mismas. Ambas posibilidades cuentan con una búsqueda de productos o empresas a medida, tal y como se describe en el modo usuario sin registro.

Otra característica del modo usuario miembro es la posibilidad de ingresar, editar y eliminar productos propios de la plataforma, la acción primera a través del perfil o interfaz del propio miembro y las dos últimas a través de la búsqueda de productos propios. Su ingreso y edición consiste en introducir una imagen de portada reajustada a sus condiciones, la cantidad del producto a vender, el tipo de producto al que pertenece, un título, un precio, un tipo de peso entre varias opciones para llevar a cabo el proceso de envío y una descripción del producto. También da la posibilidad de ingresar, editar y eliminar más imágenes aparte de la de portada en los productos propios mencionados. El procedimiento de las tres acciones es llevarlo a cabo de una imagen en una imagen hasta un máximo de nueve, admitiendo únicamente los formatos de *PNG* y *JPG*/*JPEG* (es evidente que no puedes subir las imágenes muy seguidas porque al trabajar en asíncrono, tarda en procesar, por eso hay que esperar un tiempo entre medias). Y por último, da la posibilidad de acceder a su perfil o interfaz para ver y actualizar sus datos personales, para darse de baja de la plataforma y poder dar de baja a cualquier cliente y empresa con su autorización explícita, para ver los productos que han añadido todos y cada uno de los clientes en el carrito de la compra, para ver los productos que han comprado todos y cada uno de los clientes, para ver las ganancias totales generadas por la venta de los productos propios de la plataforma, para saber los productos que quedan a la venta, para saber cuántos clientes hay, de qué género son y quienes de ellos guardan su tarjeta bancaria, para saber cuántas empresas hay registradas según los cuatro tipos mencionados anteriormente, para saber el número de trabajadores que hay en la plataforma según el departamento donde trabajen y para saber el correo electrónico y el ID de cada uno de los usuarios registrados en la plataforma.

Para concluir este modo, es posible acceder a una serie de secciones extra, que coinciden con las

del usuario empresa en cuestión, tanto por parte de que no hay ninguna sección para enviar el Currículum Vitae, como que puede llevar a cabo su cierre de sesión cuando quiera. Ésta última opción también la tiene el usuario cliente, como ya se ha mencionado anteriormente.

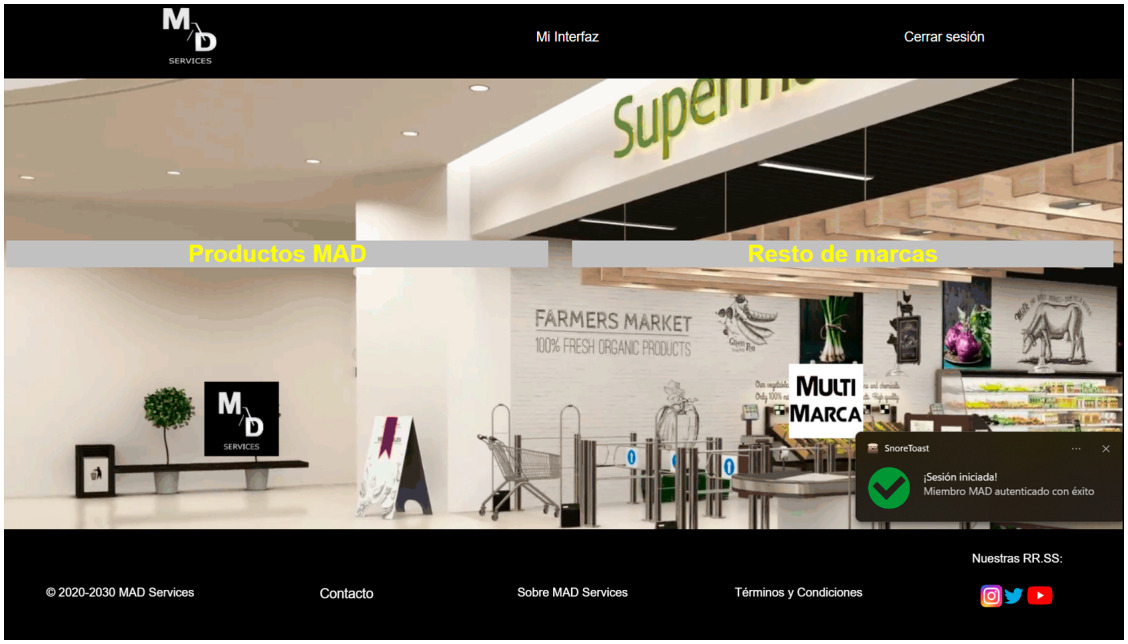


Figura 4.4: Pantalla inicial del modo usuario miembro, sacado del prototipo de la interfaz final

Capítulo 5

Conclusiones

Finalmente, habiendo realizado y explicado el proyecto, concluimos destacando la forma en que se ha desarrollado en el tiempo, los objetivos o metas que se han conseguido, el aprendizaje que se ha obtenido, las asignaturas que han servido de base, los proyectos futuros que se desarrollarán en futuras versiones y la intrahistoria del surgimiento de la idea.

5.1. Desarrollo temporal

El proyecto se ha desarrollado en cinco etapas:

Etapa 1. Durante el mes de septiembre de 2022. Es el inicio y puesta en marcha del Trabajo de Fin de Grado, con el objetivo de presentarlo al final del curso. A partir de ese mes, el autor estuvo trabajando en la empresa de *RPG Ingeniería Audiovisual S.A* como montador e instalador de cabinets (pantallas LED de Ultra Alta Definición - UHD) y equipos audiovisuales. También se programó la primera reunión entre el tutor y dicho autor de este proyecto para decidir sobre el tema a desarrollar. El segundo fue quien propuso la idea de hacer una aplicación web de comercio electrónico para vender productos online en el entorno de tiempo de ejecución en tiempo real de *Node JS*, con el *framework* para servidor web de *Express JS* y el *framework* para la interfaz de usuario de *React*. Aunque el primero fue quien recomendó hacerlo en RV (Realidad Virtual) a través de *A-Frame*, en vez del *framework* para la interfaz de usuario que propuso el autor.

Etapa 2. Durante los meses de octubre y noviembre de 2022. Con la documentación sobre la tecnología recomendada por el tutor, los apuntes de asignaturas relacionadas con la realización de aplicaciones en el lado del servidor y del cliente, como *Construcción de Servicios y Aplicaciones Audiovisuales en Internet* y *Laboratorio de Tecnologías Audiovisuales en la Web*, la documentación del entorno de *Node JS* y el *framework* de *Express JS*, el autor empezó la investigación y estructuración del prototipo del proyecto. Como trabajaba en la empresa mencionada anteriormente de 07:00 a 15:00, sólo tenía las tardes para dedicarle al Proyecto de Carrera, donde en muchas ocasiones acababa con mucha saturación y cansancio. Durante esos meses se le dedicó, aproximadamente, 39 horas durante los días de diario y 36 horas durante los fines de semana. Para que finalmente se llegara a la conclusión de que a futuro sí que se podría desarrollar con *A-Frame*, pero llevaría más de un año, debido a su complejidad para adaptar la interfaz de usuario en dicha tecnología a un comercio electrónico por parte de un desarrollador con los conocimientos básicos sobre *A-Frame*. Aparte de que el tutor no podía ayudar mucho, ya que tampoco la conocía a la perfección. Fue entonces cuando se volvió al formato que propuso el autor desde el principio, dejando el desarrollo en RV (Realidad Virtual) para una segunda versión futura de la aplicación. Es decir, se volvió a la aplicación web con el mismo entorno, mismo *framework* de servidor web y *React* como *framework* para la interfaz de usuario. La diferencia estaba en que pasó de ser una aplicación web de comercio electrónico a una *Plataforma de Comercio online*, *Centro Comercial online*, o *Marketplace* para empresas. Además, a finales de ese periodo, el autor cambio de trabajo, ya que otra empresa llamada MdTel (Maintenance Development Technology S.A), también de Madrid, le ofreció un mejor contrato, con mejores condiciones y se cambió.

Etapa 3. Durante los meses de diciembre de 2022 y enero y febrero de 2023. Ya en su nuevo trabajo, también se empezó con la investigación y puesta en práctica del prototipo de *Marketplace* bajo el entorno de *Node JS*, el *framework* de servidor web de *Express JS* y el *framework* de interfaz de usuario de *React*, cuya funcionalidad consistía, por un lado, en la venta de productos propios como un comercio electrónico más y, por otro lado, en el desarrollo de una plataforma común donde las empresas se pudieran registrar y vender sus productos y/o servicios, tipo Centro Comercial online. El objetivo de hacerlo con *React* fue para hacer la interfaz de usuario de forma dinámica, sin necesidad de crear cada página que necesitara la aplicación. No obstante,

el autor seguía trabajando a jornada completa, esta vez de 08:00 a 16:30, con media hora para comer, donde al igual que en el anterior trabajo, sólo podía dedicar las tardes al Trabajo de Fin de Grado y, encima, también en muchas ocasiones acababa con mucha saturación y cansancio. Durante esos meses se le dedicó, aproximadamente, 120 horas durante los días de diario y 78 horas durante los fines de semana. Fueron meses de mucho agobio, de prueba y error con aplicaciones web prácticas, viendo ejemplos y vídeos tutoriales de prueba. En verdad, así es como se aprende, con el método o técnica de *prueba-error*. Pero al final, se llegó a la conclusión de que la tecnología *React* resultaba ser bastante tediosa y compleja, debido a su desarrollo de páginas dinámicas con la consiguiente dificultad para desarrolladores que están iniciándose en dicha tecnología. Aparte, el tutor tampoco la conocía a la perfección. Aun así, se la dieron más oportunidades porque hay que ser perseverante con los objetivos a obtener. Se seguía programando y desarrollando con el método de *prueba-error*, cuyo fin era realizar la demo del *Marketplace*. Hasta que se llegó a un punto en que, por productividad y finalidad del proyecto, el tutor y el autor decidieron migrar del *framework* de *React* a los motores de plantillas o vistas estáticas de *Express JS* para la realización de la interfaz de usuario del proyecto en cuestión.

Etapa 4. Durante los meses de marzo a mayo de 2023. Fue cuando el autor decidió dejar el trabajo en MdTel y dedicarse en su totalidad al Trabajo de Fin de Grado, ya que veía que al final no iba a poder presentarlo por falta de tiempo. Este periodo fue el más clave de todos, donde se dio todo el desarrollo de la demo actual, con la API (Interfaz de Programación de Aplicaciones) bajo el entorno de *Node JS* con el *framework* de *Express JS*, la tecnología de establecimiento de conexión con bases de datos *MySQL* y la interfaz de usuario con la vista estática o motor de plantillas de *PUG*. Esta última tecnología fue la elegida frente a otras, debido a que es la que tiene la sintaxis más sencilla, clara y directa, tanto a la hora de escribir como de leer y modificar. Su código fuente es limpio y conciso a la hora de trabajar y corregir errores. Y funciona de acuerdo a la sintaxis de HTML, pero simplificando las etiquetas. No obstante, en mayo también estuve trabajando de profesor particular a través de la Plataforma de clases particulares y grupales, Didáctica S.L. Tenía tres alumnas a mi cargo, impartiendo matemáticas de 1º ESO a dos de ellas y matemáticas de 4º ESO a la otra restante. La verdad me gusta dar clases de apoyo escolar y encima, tengo más de tres años de experiencia en ello.

Etapa 5. Durante parte de mayo y el mes de junio de 2023. Es la última etapa del desarrollo temporal del proyecto, cuando se redactó la memoria del proyecto y se preparó la presentación para defenderlo. Lo primero se llevó a cabo por medio de la tecnología de *LaTex* con el servicio online de *Overleaf*, recomendado por el tutor. Lo segundo se llevó a cabo por medio de la tecnología de *Microsoft PowerPoint*, que es la que mejor conoce el autor.

5.2. Consecución de objetivos

Los objetivos de este proyecto se centran en la realización de la aplicación web descrita, a partir del entorno en tiempo de ejecución en tiempo real mencionado en capítulos anteriores de la memoria, que es clave para poder generar el *framework* también mencionado y donde funciona la API o servidor web del prototipo del proyecto.

También están los objetivos de tipo comercial, donde uno de ellos, es la venta de productos propios como un comercio electrónico más y, el otro, es el desarrollo de una plataforma común donde conectar y unificar los negocios locales, PYMES (Pequeñas y Medianas Empresas, de sus siglas en español) y *Start-Ups* o empresas emergentes para que se puedan dar a conocer y ampliar sus potenciales clientes. Aunque también podrán ser incluidas las grandes empresas o Multinacionales, pero en este caso, no es tanto para darse a conocer o ampliar sus potenciales clientes, debido a su gran influencia en el mercado, sino que sería, sobre todo, para aliviar su *stock* de productos y/o servicios. Es decir, la meta final es el desarrollo de un Centro Comercial online.

Por último, están los objetivos de tipo funcionamiento de la aplicación web, donde hay varios: el registro de los clientes para poder comprar, tanto productos propios de la Plataforma de Comercio online, como productos y/o servicios del resto de empresas. El registro de empresas oficiales con su código CIF/NIF para poder diseñar la interfaz de su marca mostrándosela al cliente y poder ofertarles los productos y/o servicios que quieran comprar. El registro de los miembros para asegurar el funcionamiento correcto de la aplicación, controlar que no haya fraude, verificar compras realizadas, verificar los clientes, empresas y miembros que haya y resolver todas las dudas posibles. Y la utilización de la aplicación web sin registro para poder ver

los productos y/o servicios del Centro Comercial online y los propios de la plataforma, pero sin poder comprar ni añadir al carrito.

5.3. Lecciones aprendidas

Este proyecto ha enseñado al autor sobre la realización y puesta en práctica de una aplicación web de *Marketplace*, ya que la única experiencia previa que tenía, era cuando estuvo seis meses de Erasmus+ en Novi Sad (Serbia), desde febrero de 2022 a julio de 2022 y, llevó a cabo junto a tres compañeros serbios, el desarrollo de una aplicación web de contactos y búsqueda de empleo con su *back-end* o API (desarrollado más por dos de los serbios) y su *front-end* o interfaz de usuario (desarrollado más por el otro serbio y el propio autor) en *Angular*. También ha aprendido sobre la tecnología web o *framework* de *Express JS*, cuyos conocimientos se fueron adquiriendo en función avanzaba en el Trabajo de Fin de Grado, a través de la técnica o método de *prueba-error*, ya que no tenía conocimientos previos sobre ello. Lo mismo con la tecnología de *MySQL*, *phpMyAdmin* y las vistas estáticas o motores de plantillas *PUG*. Aunque este último, al estar basado en *HTML*, pero simplificando sus etiquetas, no ha sido necesario tanto aprendizaje como con el resto, porque esta última tecnología mencionada sí que se ha dado y trabajado en algunas asignaturas del Grado en Ingeniería de Sistemas Audiovisuales y Multimedia.

Por otro lado, se ha aprendido sobre la gran variedad de tecnologías o dependencias que agilizan el desarrollo del servidor web por medio del *framework* de *Express JS*. Todas ellas mencionadas en el capítulo del *Estado del arte, de la técnica y sistemas relacionados*.

Y por último, sobre tecnologías ajenas al propio desarrollo de la aplicación, como *Overleaf/LaTeX* y *Thunder Client*. La primera, recomendada por el tutor de este proyecto, ya que el autor no la conocía, donde el aprendizaje se ha llevado a cabo por la técnica o método de *prueba-error* y gracias a su manual de usuario[86]. La segunda, conocida por el proyecto de la aplicación web que realicé en Serbia, junto a mis tres compañeros, ya que sirve para verificar que el funcionamiento de la API es correcto, antes de tener la propia interfaz de usuario. Por otra parte, *Adobe Photoshop*, *Adobe Premiere* y *Wondershare Filmora 9*, son tecnologías para la edición

de imágenes y vídeos, cuyo aprendizaje se lleva dando desde que el autor tenía doce años de edad y ajeno a la Universidad.

5.4. Asignaturas relacionadas

No obstante, hay tecnologías que el autor las ha aprendido gracias a asignaturas de la carrera para el posible y completo desarrollo de una aplicación web de *Marketplace*. Éstas han sido: *Arquitectura de Internet*, por su contenido y aprendizaje sobre la Arquitectura TCP/IP, base de una aplicación web; *Sistemas Telemáticos para Medios Audiovisuales*, por el conocimiento de HTTP/HTTPS, protocolo utilizado para el básico funcionamiento de una aplicación web; *Protocolos para la Transmisión de Audio y Vídeo en Internet*, en cuyo caso no se cursó en la propia Universidad, ya que el autor la cursó durante su estancia de Erasmus+ en Serbia; *Construcción de Servicios y Aplicaciones Audiovisuales en Internet*, para el desarrollo del *front-end* o interfaz de usuario con las tecnologías de *HTML*, *CSS* y *JavaScript*; y *Laboratorio de Tecnologías Audiovisuales en la Web*, para el desarrollo del *back-end* o API con la tecnología del entorno de tiempo de ejecución en tiempo real de *Node JS*, el desarrollo de documentos y datos en *XML* y *JSON*, la realización de formularios y cookies, la importancia de las peticiones AJAX, la administración de los módulos, dependencias o tecnologías de *Node JS*, el funcionamiento de los *Websockets* y la tecnología de *Electron*.

5.5. Trabajos futuros

Antes de acabar, es importante remarcar que como todo proyecto desarrollado, tendrá versiones futuras que mejorarán e incluso cambiarán el formato de la aplicación. Como ya se ha mencionado, el objetivo futuro será realizar el *Marketplace* en RV (Realidad Virtual), a través de la tecnología web o *framework* de *A-Frame*[139].

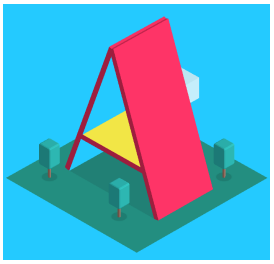


Figura 5.1: Logo de *A-Frame*

Dicha nueva tecnología en 3D (tres dimensiones) es un *framework* en *JavaScript* para generar web en un entorno virtual, usando la Ar-

quitectura ECS (Sistema de Componentes de Entidad, por sus siglas en inglés), sobre todo, para el desarrollo de juegos donde cada objeto es una entidad.

Su elección con respecto a otras tecnologías de RV (Realidad Virtual) ha sido porque está diseñado para que los objetos ya no estén fijos en una jerarquía, por lo que hace que ellos puedan tener un comportamiento sin límites.

Volviendo a la Arquitectura ECS, caben destacar tres campos importantes: los Sistemas, las Entidades y los Componentes. Los primeros son el entorno donde manejar y desarrollar los componentes, usados para separar la funcionalidad de la información y ocupados de la lógica del uso de ellos. Los segundos son objetos contenedores donde están los Componentes. Y los terceros son propiedades que hacen diferenciar una entidad de otra, haciendo que tengan su propio comportamiento, apariencia y funcionalidad.

Centrándonos en su desarrollo práctico, las Entidades se representan con la etiqueta *a-entity*, situadas dentro de *a-scene* que son los Sistemas. Los Componentes son las propiedades *HTML* escritas en *a-entity*.

Por tanto, es evidente que el *framework A-Frame* permite llevar a la Arquitectura ECS a otro nivel, por ser una tecnología declarativa, tener *HTML* y estar basado en el DOM, la interfaz de programación para los documentos *HTML* y *XML*. Esta última tecnología proporciona a dicha arquitectura: un potente sistema selector de consultas que permite seleccionar una entidad o entidades para que coincidan con una condición, a través de *ID*, *clases* o *atributos de datos*; la capacidad de escuchar y emitir eventos como comunicación entre entidades; la actualización de elementos HTML y del árbol; la solicitud de entidades con un determinado conjunto de componentes; y la realización de un código fuente limpio, declarativo, legible y extensible.

Ejemplo de Sistema:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, WebVR! - A-Frame</title>
    <meta name="description" content="Hello, WebVR! - A-Frame">
    <script src="https://aframe.io/releases/0.5.0/aframe.min.js">
    </script>
  </head>
  <body>
    <a-scene>
      <a-entity light="color: #ccccff; intensity: 1;
        type: ambient;"></a-entity>
      <a-entity geometry="primitive: plane; width: 10000;
        height: 10000;" rotation="-90 0 0"
        material="src: #grid; repeat: 10000 10000;
        transparent: true; metalness:0.6;
        roughness: 0.4; sphericalEnvMap: #sky;">
      </a-entity>

      <a-box position="-1 0.5 -3" rotation="0 45 0"
        color="#4CC3D9"></a-box>
      <a-sphere position="0 1.25 -5" radius="1.25"
        color="#EF2D5E"></a-sphere>
      <a-cylinder position="1 0.75 -3" radius="0.5"
        height="1.5" color="#FFC65D"></a-plane>
      <a-sky color="#ECECEC">
    </a-scene>
  </body>
</html>
```

SISTEMA

Figura 5.2: Tipos de Sistemas de *A-Frame*, sacado de [140]

Ejemplo de Entidad:

```
  type: ambient;"></a-entity>
<a-entity geometry="primitive: plane; width: 10000;
  height: 10000;" rotation="-90 0 0"
  material="src: #grid; repeat: 10000 10000;
  transparent: true; metalness:0.6;
  roughness: 0.4; sphericalEnvMap: #sky;">
</a-entity>

<a-box position="-1 0.5 -3" rotation="0 45 0
  color="#4CC3D9"></a-box>
```

Entidad

Figura 5.3: Tipos de Entidades de *A-Frame*, sacado de [140]

Ejemplo de Componente:

```
<a-entity geometry="primitive: plane; width: 10000;
  height: 10000;" rotation="-90 0 0"
  material="src: #grid; repeat: 10000 10000;
  transparent: true; metalness:0.6;
  roughness: 0.4; sphericalEnvMap: #sky;">
</a-entity>
```

Componente

Figura 5.4: Tipos de Componentes de *A-Frame*, sacado de [140]

A continuación, se mostrará la estructura de una aplicación con *A-Frame* y su prototipo de interfaz de usuario (no significa que vaya a ser la interfaz futura, sino un simple ejemplo):

```
index.js > madservices.listen() callback
1  const servidor = require('express');
2  const madservices = servidor();
3  require('./config/env.js');
4
5  madservices.use(servidor.static('public'));
6
7  madservices.listen(process.env.PUERTO, () => {
8      console.log(`MAD Services escuchando en: ${process.env.ENLACE}${process.env.PUERTO}`);
9  });
```

Figura 5.5: Código fuente del archivo principal del servidor web con *Express JS*, sacado del código fuente del prototipo del proyecto con RV (Realidad Virtual)

```
8      <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
9  </head>
10 <body>
11     <a-scene>
12         <a-assets>
13             <img src="textures/sky.jpeg" id="sky">
14             <img src="textures/madservices.png" id="madservices">
15         </a-assets>
16         <!-- Fondo de Marketplace -->
17         <a-sky src="#sky"></a-sky>
18         <!-- Objetos de MAD Services flotantes -->
19         <a-box position="0 -2 -3.8" rotation="0 0 0" material="src: #madservices"></a-box>
20         <a-box position='2 2 -4' rotation="0 0 0" material="src: #madservices"></a-box>
21         <a-box position="-1 3 -0.75" rotation="0 0 0" material="src: #madservices"></a-box>
22         <a-box position="3 -2 -3.8" rotation="0 0 0" material="src: #madservices"></a-box>
23         <a-box position='-5 2 -4' rotation="0 0 0" material="src: #madservices"></a-box>
24         <a-box position="6 3 -0.75" rotation="0 0 0" material="src: #madservices"></a-box>
25         <!-- Botón Iniciar sesión -->
26         <a-entity position="0 1.5 -3">
27             <a-box color="black" width="1" height="0.5" depth="0.1"></a-box>
28             <a-text value="Login" position="-0.3 0 0.05"></a-text>
29             <a-plane position="0 0 0" width="1" height="0.5" material="color: black;"></a-plane>
30             <a-box position="0 0 -0.1" width="1" height="0.5" depth="0.1" material="color: black;"></a-box>
31         </a-entity>
32         <!-- Botón Registrarse -->
33         <a-entity position="-1.5 2.5 -3">
34             <a-box color="black" width="1" height="0.5" depth="0.1"></a-box>
35             <a-text value="Sign up" position="-0.3 0 0.05"></a-text>
36             <a-plane position="0 0 0" width="1" height="0.5" material="color: black;"></a-plane>
37             <a-box position="0 0 -0.1" width="1" height="0.5" depth="0.1" material="color: black;"></a-box>
38         </a-entity>
39     </a-scene>
40 </body>
41 </html>
```

Figura 5.6: Código fuente de la interfaz de usuario, sacado del código fuente del prototipo del proyecto con RV (Realidad Virtual)

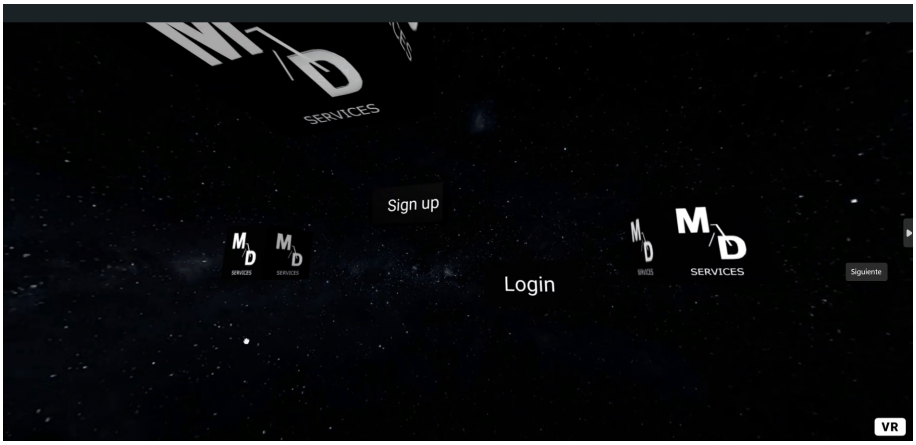


Figura 5.7: Imagen de la interfaz de usuario del prototipo del proyecto con RV (Realidad Virtual), editado en *Wondershare Filmora 9* y *Adobe Premiere* y sacado del canal de YouTube MAD Services[141]

Por otro lado, cuando se comercialice la aplicación web, se acordará con las empresas tipo comercios para conectar la plataforma MAD Services con su comercio electrónico (si tienen), realizando el proceso de autorización entre ambas partes, que consistirá en registrar automáticamente al cliente que haya pulsado en conectar y, por ende, enviar todos sus datos de la plataforma a la base de datos del comercio electrónico al que ha conectado, con el objetivo y ventaja de no tener que estar registrándose en todos y cada uno de ellos. Si el cliente ya está registrado en el comercio electrónico a conectar, se actualizarán sus datos. También tendrá la posibilidad de darse de baja de los que quiera. Y habrá en la interfaz de cada empresa registrada (aquí el tipo da igual) los productos y/o servicios que éstas quieran.

Además, se establecerá el proceso de autenticación con *Cookies* al iniciar sesión, para que a partir de la *Cookie* creada, la almacene para futuros inicios de sesión y así, facilitar el proceso de entrada de usuarios en la plataforma.

5.6. Intrahistoria de la idea

Y para concluir, es importante saber que todo proyecto tiene su historia detrás, la idea que lo hizo surgir con ambición, lucha diaria y perseverancia.

Realmente, el autor de este proyecto nunca hubiera pensado en hacer una aplicación web de comercio electrónico y/o *Marketplace* sin los siguientes acontecimientos que surgieron en su vida. En primer lugar, su curiosidad por las ventas empezó cuando estuvo trabajando en las tiendas de ropa, juguetes y ultramarinos de Kalahari Resorts and Conventions en Wisconsin Dells, Wisconsin, Estados Unidos de América. Fue durante cuatro meses, de junio 2019 a septiembre 2019, gracias al programa de trabajo *Work and Travel*. En segundo lugar, a raíz de vender libros y otros productos por medio de otro comercio electrónico existente durante la Pandemia de la Covid-19 (2020-2022). Y en tercer lugar, gracias a su experiencia durante el periodo de seis meses de Erasmus+ en Novi Sad (Serbia), desde febrero de 2022 a julio de 2022, que recordará para siempre, donde hizo amistad con personas asombrosas, conoció y descubrió la cultura balcánica y mejoró su inglés. Allí tuvo una asignatura sobre servicios web y aplicaciones web, asignatura que equivaldría a *Protocolos para la Transmisión de Audio y Vídeo en Internet* en

su Universidad de España, donde desarrolló junto a tres compañeros serbios (en inglés, obviamente) una aplicación web de contactos y búsqueda de empleo con su *back-end* y su *front-end*. Sinceramente, fue un proyecto muy apasionante para él y el inicio de su pasión por el desarrollo web, lo que encaminaría a la realización de un TFG (Trabajo de Fin de Grado) sobre ello.

Antes de acabar, cabe destacar que a través de sus experiencias Worldpackers, tanto en el voluntariado de organización de campings y medios rurales de Döbriach, al sur de Austria, como en el voluntariado sobre jardinería, alimentación y medios rurales en la ecogranja de Sammatz, cerca de Hamburgo (Alemania), descubrió la opción de ser *nómada digital*, es decir, de trabajar de forma online, mientras viajas por el mundo, visitas nuevos lugares y vives nuevas experiencias.

Bibliografía

[1] Información sobre qué es una aplicación web:
-https://www.strappinc.com/blog/strapp-datos/
que-es-una-aplicacion-web

[2] Información sobre los *Marketplace* y sus tipos:
-https://blog.hubspot.es/sales/que-es-marketplace#:~:
text=Un%20marketplace%20es%20un%20espacio,log%C3%ADstica%
20para%20soportar%20dichas%20transacciones.

[3] Información sobre los tipos de componentes que tiene la aplicación web del prototipo desarrollado:
-https://www.ibm.com/docs/es/db2-for-zos/11?topic=
environment-components-web-based-applications

[4] Información sobre los tipos de interfaces de usuario de una aplicación web según su programación y diseño:
-https://www.esic.edu/rethink/tecnologia/
que-son-las-aplicaciones-web-c
-https://emma.io/blog/tipos-aplicaciones-caracteristicas-ejemplos/
#2_App_hibrida

[5] Información sobre qué son las ventas cruzadas:
-https://www.salesforce.com/es/learning-centre/sales/
cross-selling/

[6] Esquema gráfico de un *Marketplace* de tipo *B2C*:
-https://axiadigital.com/solutions/marketplaces-e-commerce/marketplace-platform-b2c-b2b/

[7] Información sobre el navegador web de *Brave*:
-https://brave.com/es/

[8] Información sobre el navegador web de *Mozilla Firefox*:
-https://www.arimetrics.com/glosario-digital/firefox

[9] Información sobre qué es una aplicación web:
-https://www.strappinc.com/blog/strapp-datos/que-es-una-aplicacion-web

[10] Información sobre el protocolo HTTP:
-https://concepto.de/http/

[11] Información sobre las características de una aplicación web:
-https://www.esic.edu/rethink/tecnologia/que-son-las-aplicaciones-web-c
-https://blog.hubspot.es/website/que-es-aplicacion-web

[12] Información sobre el protocolo HTTPS:
-https://www.aepd.es/es/prensa-y-comunicacion/blog/https-cifrado-en-la-web

[13] Información sobre el sistema de cifrado TLS:
-https://www.webperformance.es/la-version-de-tls/

[14] Imagen del Modelo de comunicaciones Cliente-Servidor de una aplicación web:
-https://blog.infranetworking.com/modelo-cliente-servidor/

[15] Información sobre un Proxy:
-https://www.mcafee.com/blogs/es-es/privacy-identity-protection/que-es-un-proxy/

[16] Información sobre los tipos de interfaces de usuario de una aplicación web según su programación y diseño:
-https://www.esic.edu/rethink/tecnologia/que-son-las-aplicaciones-web-c
-https://emma.io/blog/tipos-aplicaciones-caracteristicas-ejemplos/#2_App_hibrida

[17] Imagen o esquema gráfico de los tipos de Aplicaciones Web según su programación y diseño:
-https://emma.io/blog/tipos-aplicaciones-caracteristicas-ejemplos/#2_App_hibrida

[18] Información sobre los tipos de una aplicación web según sus categorías desarrolladas:
-https://www.yeeply.com/blog/5-tipos-desarrollo-de-aplicaciones-web/#portal

[19] Información sobre los *Marketplace* y sus tipos:
-https://blog.hubspot.es/sales/que-es-marketplace#:~:text=Un%20marketplace%20es%20un%20espacio,log%C3%ADstica%20para%20soportar%20dichas%20transacciones.

[20] Información sobre la empresa mayorista Crambo S.A que vende y distribuye a otras empresas:
-https://www.retailactual.com/empresas/crambo-sa

[21] Imagen o representación gráfica de los tipos de *Marketplace*:
-https://es.vecteezy.com/arte-vectorial/3836190-modelos-de-negocio-incorporacion-aplicacion-movil-pagina-pan

[22] Información sobre el entorno de tiempo de ejecución Node JS:
-https://openwebinars.net/blog/que-es-nodejs/

[23] Imagen de las características del entorno de Node JS:
-https://www.section.io/engineering-education/history-of-nodejs/

[24] Información sobre el *framework* de *Express JS* del entorno *Node JS*:
-https://www.wearemarketing.com/es/blog/
frameworks-en-el-desarrollo-web-las-mejores-practicas-para-tu-negoci
html#:~:text=%C2%BFQu%C3%A9%20es%20un%20framework%20web,y%
2Fo%20funcionalidades%20ya%20creadas.
-https://kinsta.com/es/base-de-conocimiento/
que-es-express/#qu-es-expressjs

[25] Información sobre los tipos de *framework* del entorno de *Node JS*:
-https://blog.hubspot.es/website/framework-desarrollo-web

[26] Información sobre el *Stack MEAN* al que pertenece *Express JS*:
-https://platzi.com/blog/que-es-mean-full-stack-javascript/
#:~:text=MEAN%20es%20desarrollo%20full%2Dstack,js.

[27] Información sobre la tecnología *MySQL* de establecimiento de conexiones con las bases de datos:
-https://kinsta.com/es/base-de-conocimiento/que-es-mysql/
-https://blog.mdcloud.es/postgresql-vs-mysql-cual-usar-para-mi-proyec
#:~:text=Las%20principales%20razones%20para%20elegir,muy%
20flexible%20y%20altamente%20escalable.

[28] Información sobre la tecnología *phpMyAdmin* de visión de la interfaz de usuario para administrar las bases de datos:
-https://www.dongee.com/tutoriales/que-es-phpmyadmin/
-https://www.redeszone.net/2016/06/15/
phpmyadmin-supera-exito-una-auditoria-seguridad/

[29] Información sobre la tecnología del servidor web *Apache* para poder usar *phpMyAdmin*:
-https://www.webempresa.com/hosting/que-es-servidor-apache.
html

[30] Información sobre la tecnología del lenguaje *PHP* que usa *phpMyAdmin*:
-https://www.php.net/manual/es/intro-whatis.php

[31] Información sobre la tecnología *PUG*, utilizada como motor de plantillas o vista estática para el desarrollo de la interfaz de usuario de la demo del proyecto:
-https://www.silocreativo.com/introduccion-primeros-pasos-pug/

[32] Información sobre la tecnología *CSS*, utilizada como diseño y estilos de la apariencia de las páginas de la interfaz de usuario de la demo del proyecto:
-https://www.nextu.com/blog/que-es-css-rc22/

[33] Información sobre la tecnología *JS*, utilizada para poder conectar el código fuente del *Modelo* con el de la *Vista* a través de las solicitudes *HTTP*:
-https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[34] Información sobre la subtecnología o dependencia *Axios* de NPM:
-https://keepcoding.io/blog/libreria-axios-para-javascript/#Que_es_la_libreria_axios_para_JavaScript

[35] Imagen sacada de un ejemplo de petición a un contenido de un enlace HTTP con *axios* como función:
-https://www.knowledgehut.com/blog/web-development/http-requests-with-axios-npm

[36] Información sobre la subtecnología o dependencia *Bcrypt* de NPM:
-https://www.izertis.com/es/-/blog/encriptacion-de-password-en-nodejs~:text=Bcrypt%20es%20una%20funci%C3%B3n%20de,algunas%20distribuciones%20Linux%20y%20SUSE.

[37] Información sobre el tipo de cifrado que usa *Bcrypt*:
-https://www.wikiwand.com/es/Blowfish

[38] Información sobre la subtecnología o dependencia de *Body-parser* de NPM:
-https://apuntes.de/nodejs-desarrollo-web/body-parser/#gsc.tab=0

[39] Imagen de un ejemplo de funcionamiento de *Body-parser*:
-https://www.simplilearn.com/tutorials/nodejs-tutorial/body-parser-in-express-js

[40] Información sobre la subtecnología o dependencia de *Card-validator* de NPM:
-https://www.npmjs.com/package/card-validator

[41] Imagen del funcionamiento de *Card-validator*:
-https://www.cssscript.com/minimal-credit-card-input-validation-libra

[42] Información sobre la subtecnología o dependencia de *Cookie-parser* de NPM:
-https://ull-esit-dsi-1617.github.io/estudiar-cookies-y-sessions-en-e
CAPITULO8/CAPITULO8.html

[43] Imagen del funcionamiento de *Cookie-parser*:
-https://lazymankook.tistory.com/35

[44] Información sobre la subtecnología o dependencia de *Cors* de NPM y de donde se ha obtenido su imagen:
-https://developer.mozilla.org/es/docs/Web/HTTP/CORS

[45] Información sobre la subtecnología o dependencia de *Country-list* del *framework* de *Express JS*:
-https://www.npmjs.com/package/country-list

[46] Información sobre la subtecnología o dependencia de *Country-list-spanish* de NPM:
-https://www.npmjs.com/package/country-list-spanish/v/0.2.0

[47] Imagen del funcionamiento del listado de países de *Country-list* y *Country-list-spanish*:
-https://co.pinterest.com/pin/391039180144290671/?amp_client_id=CLIENT_ID%28_%29&mweb_unauth_id=%7B%7Bdefault.session%7D%7D&simplified=true

[48] Información sobre la subtecnología o dependencia de *Dotenv* de NPM:
-https://www.victorvr.com/tutorial/variables-de-entorno-con-nodejs

[49] Imagen del funcionamiento de *Dotenv*:
-https://sherryhsu.medium.com/managing-configurations-in-node-js-apps

[50] Información sobre la subtecnología o dependencia de *Express-session* de NPM:
-https://www.npmjs.com/package/express-session

[51] Imagen del funcionamiento de *Express-session*:
-https://npm.io/package/express-session-resolver

[52] Información sobre la subtecnología o dependencia de *Fs* de NPM:
-https://kinsta.com/es/base-de-conocimiento/nodejs-fs/

[53] Imagen del funcionamiento de *Fs*:
-https://parzibyte.me/blog/2018/12/27/
leer-archivo-node-js-fs-readline/

[54] Información sobre la subtecnología o dependencia de *Helmet* de NPM:
-https://platzi.com/clases/1649-passport-2019/
22007-seguridad-con-helmet-y-npm-audit/#:~:text=Helmet%
20es%20un%20middleware%20que,las%20headers%20de%20nuestra%
20aplicaci%C3%B3n.

[55] Imagen del funcionamiento de *Helmet*:
-https://www.veracode.com/blog/secure-development/
fasten-your-helmetjs-part-1-securing-your-express-http-headers

[56] Información sobre la subtecnología o dependencia de *Http-errors* de NPM:
-https://www.npmjs.com/package/http-errors

[57] Imagen del funcionamiento de *Http-errors*:
-https://github.com/npm/npm/issues/2694

[58] Información sobre la subtecnología o dependencia de *Luxon* de NPM:
-https://www.npmjs.com/package/luxon

[59] Imagen del funcionamiento de *Luxon*:
-https://jquery-plugins.net/luxon-library-for-working-with-dates-and-

[60] Información sobre la subtecnología o dependencia de *Method-override* de NPM:
-https://www.npmjs.com/package/method-override

[61] Información sobre la subtecnología o dependencia de *Multer* de NPM:
-https://github.com/expressjs/multer/blob/master/doc/
README-es.md

[62] Imagen del funcionamiento de *Multer*:
-https://codingstatus.com/node-js-file-upload/

[63] Información sobre la subtecnología o dependencia de *Nif-dni-nie-cif-validation* de NPM:
-https://www.npmjs.com/package/nif-dni-nie-cif-validation

[64] Información sobre la subtecnología o dependencia de *Node-notifier* de NPM:
-https://www.npmjs.com/package/node-notifier

[65] Imagen del funcionamiento de *Node-notifier*:
-https://davidwalsh.name/system-notifications-node

[66] Información sobre la subtecnología o dependencia de *Path* de NPM:
-https://nodejs-es.github.io/api/path.html

[67] Imagen del funcionamiento de *Path*:
-https://www.javascripttutorial.net/nodejs-tutorial/
nodejs-path-module/

[68] Información sobre la subtecnología o dependencia de *Postcode-validator* de NPM:
-https://www.npmjs.com/package/postcode-validator

[69] Imagen del funcionamiento de *Postcode-validator*:
-https://www.edq.com/blog/what-is-a-postal-code-a-zip-code-and-how-do

[70] Información sobre la subtecnología o dependencia de *Serve-favicon* de NPM:
-https://expressjs.com/en/resources/middleware/
serve-favicon.html

[71] Imagen del funcionamiento de *Serve-favicon*:
-https://btholt.github.io/intro-to-web-dev-v2/node

[72] Información sobre la subtecnología o dependencia de *Sharp* de NPM:
-https://products.fileformat.com/es/image/javascript/sharp/

[73] Imagen del funcionamiento de *Sharp*:
-https://www.npmjs.com/package/gatsby-plugin-sharp

[74] Información sobre la subtecnología o dependencia de *Util* de NPM:
-https://www.npmjs.com/package/util

[75] Información sobre la subtecnología o dependencia de *Validator* de NPM:
-https://www.npmjs.com/package/validator

[76] Imagen del funcionamiento de *Validator*:
-https://auth0.com/blog/express-validator-tutorial/

[77] Información sobre la subtecnología o dependencia de *Nodemon* como dependencia o subtecnología de desarrollo de NPM:
-https://nodemon.softonic.com/#:~:text=Nodemon%20es%20un%20desarrollo%20programa,que%20hacerlo%20manualmente%20nunca%20m%C3%A1s.

[78] Imagen del logo de *Nodemon* como dependencia o subtecnología de desarrollo:
-https://github.com/remy/nodemon

[79] Información sobre la subtecnología o dependencia de *Debug* de NPM:
-https://medium.com/@aarnlpezsosa/usando-el-m%C3%B3dulo-debug-en-node-js-d9b9ca39fee1

[80] Imagen del logo de *Debug* de NPM:
-https://logopond.com/swordfish625/showcase/detail/276173

[81] Información sobre el software *LaTex*, software base del servicio online *Overleaf*:
-https://www.uv.es/~ivorra/Latex/LaTeX.pdf

[82] Otra información sobre *Overleaf* y *LaTex*:
-https://yosigopublicando.ugr.es/courses/introduccion-al-lenguaje-latex-y-overleaf-para-edicion-de-textos-aca

#:~:text=Overleaf%20es%20un%20servicio%20de,todas%20las%
20herramientas%20de%20Overleaf.

[83] Información sobre el lenguaje *Tex*, lenguaje base del software *LaTex* y sobre su creador:
-https://www.xataka.com/magnet/la-historia-detras-de-donald-knuth-el-

[84] Información sobre el creador del software *LaTex*:
-https://www.microsiervos.com/archivo/ordenadores/
leslie-lamport-matematico-revoluciono-informatica.html

[85] Información sobre las características del servicio online de *Overleaf*:
-https://www.uc3m.es/sdic/servicios/overleaf#descripcion

[86] Manual de ayuda para aprender sobre el servicio online de *Overleaf*:
-https://es.overleaf.com/learn/how-to

[87] Información sobre el editor de código fuente *Visual Studio Code*:
-https://openwebinars.net/blog/que-es-visual-studio-code-y-que-venta

[88] Encuesta de 2021 sobre la popularidad de *Visual Studio Code* entre los desarrolladores:
-https://insights.stackoverflow.com/survey/2021#
most-popular-technologies-new-collab-tools

[89] Información sobre *Thunder Client*, que es una extensión de *Visual Studio Code* para simular llamadas a la API (Interfaz de Programación de Aplicaciones):
-https://www.latirus.com/blog/2021/04/23/
thunder-client-la-alternativa-a-postman-en-vscode/
-https://fullstackdeveloper.guru/2023/01/09/
postman-vs-thunder-client-for-vscode/
-https://rangav.medium.com/thunder-client-alternative-to-postman-68ee

[90] Información sobre el sistema de control de versiones de código abierto *Git*:
-https://www.atlassian.com/es/git/tutorials/install-git

[91] Imagen de los tipos de servicios en la nube que alojan los sistemas de control de versiones *Git*:
-https://www.genbeta.com/actualidad/microsoft-se-quedo-github-asi-que

[92] Información sobre el servicio en la nube *GitHub* que aloja sistemas de control de versiones *Git*:
-https://www.hostinger.es/tutoriales/que-es-github#%C2%BFQue_es_GitHub

[93] Información sobre el software de edición de imágenes *Adobe Photoshop*:
-https://www.arimetrics.com/glosario-digital/photoshop

[94] Información sobre la tecnología software de edición de vídeos y GIFs *Adobe Premiere*:
-https://pccadla.com/programas/adobe-premiere-pro/#:~:text=Premiere%20Pro%20es%20el%20software,y%20videos%20de%20gran%20calidad.

[95] Información sobre la tecnología software de edición de vídeos y GIFs *Wondershare Filmora 9*:
-https://www.lenguasmodernas.ucr.ac.cr/wp-content/uploads/2020/08/Manual-Filmora.pdf

[96] Esquema editado en *Adobe Photoshop* y sacado de:
-https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

[97] Explicación sobre qué es un servidor web o API (Interfaz de Programación de Aplicaciones):
-https://www.webempresa.com/hosting/que-es-servidor-web.html

[98] Explicación sobre qué es una *Base de datos*:
-https://www.oracle.com/es/database/what-is-database/

[99] Explicación sobre qué es una *Interfaz de usuario*:
-https://blog.hubspot.es/website/interfaz-usuario

[100] Explicación sobre qué es una *Ruta*:
-https://learn.microsoft.com/es-es/aspnet/web-forms/

overview/getting-started/getting-started-with-aspnet-45-web-forms/
url-routing

[101] Información sobre el funcionamiento de un Controlador en *Express JS*:
-https://www.adictosaltrabajo.com/2016/06/06/
primeros-pasos-con-express/#:~:text=Controlador%3A%
20donde%20se%20realiza%20el,Contiene%20la%20l%C3%B3gica%
20de%20negocio.

[102] Generación de la estructura de una aplicación por medio de un *framework* del entorno utilizado e iniciado:
-https://developer.mozilla.org/es/docs/Learn/Server-side/
Express_Nodejs/development_environment

[103] Explicación sobre las URLs:
-https://www.arimetrics.com/glosario-digital/url#:~:
text=Una%20URL%2C%20acr%C3%B3nimo%20del%20ingl%C3%A9s,
la%20recuperaci%C3%B3n%20de%20esta%20misma.
-https://www.hostinger.es/tutoriales/que-es-una-url#:~:
text=documento%20en%20Internet.-,%C2%BFQu%C3%A9%20es%20un%
20ejemplo%20de%20direcci%C3%B3n%20URL%3F,www.linux.org%2F%
20.

[104] Información sobre qué es un recurso como parte una *URL*:
-https://docs.moodle.org/all/es/Recurso_URL

[105] Explicación sobre qué es un *Controlador*:
-https://www.adictosaltrabajo.com/2016/06/06/
primeros-pasos-con-express/#:~:text=Controlador%3A%
20donde%20se%20realiza%20el,%C2%ABmasticado%C2%BB%20a%
20la%20vista.

[106] Información sobre la aplicación web externa de *GeoNames*:
-https://www.geonames.org/

[107] Explicación sobre qué es un *Modelo*:
-https://desarrolloweb.com/articulos/que-es-mvc.html

[108] Vídeo tutorial de YouTube de donde se ha sacado la instalación de *Apache*:
-https://www.youtube.com/watch?v=yBy16UXaJqQ

[109] Acceso al servidor de *Apache*:
-https://httpd.apache.org/

[110] Acceso a descargas del servidor de *Apache*:
-https://httpd.apache.org/download.cgi

[111] Acceder a descargar para Windows el servidor de *Apache*:
-https://httpd.apache.org/docs/current/platform/windows.html#down

[112] Acceder a la sección de *ApacheHaus*:
-https://www.apachehaus.com/cgi-bin/download.plx

[113] Archivo ejecutable de Apache 2.4.55 x64:
-https://www.apachehaus.com/cgi-bin/download.plx?dli=gYzwmbXNVT41EVNdnZqZVbhpWMUNlVSZETWx2Mhp3Z

[114] Acceso al ejecutable del Paquete Redistribuible para Visual Studio:
-https://aka.ms/vs/17/release/vc_redist.x64.exe

[115] Vídeo tutorial de YouTube de donde se ha sacado la instalación de *PHP*:
-https://www.youtube.com/watch?v=6BnaOpwGXY8

[116] Acceso a la página oficial de *PHP*:
-https://www.php.net/manual/es/intro-whatis.php

[117] Acceso a las descargas de *PHP*:
-https://www.php.net/downloads

[118] Acceso a la última versión de *PHP* para *Windows*:
-https://windows.php.net/download#php-8.2

[119] Acceso al ejecutable de *PHP* para *Windows*:
-https://windows.php.net/downloads/releases/php-8.2.7-Win32-vs16-x64.zip

[120] Vídeo tutorial de YouTube de donde se ha sacado la instalación de *MySQL*:
-https://www.youtube.com/watch?v=KfckVt-_LOk

[121] Acceso a la página oficial de *MySQL*:
-https://www.mysql.com/

[122] Acceso a la página de descargas de *MySQL*:
-https://www.mysql.com/downloads/

[123] Acceso a la página de *MySQL Community GPL for Windows*:
-https://dev.mysql.com/downloads/

[124] Acceso al instalador de *MySQL* para *Windows*:
-https://dev.mysql.com/downloads/installer/

[125] Acceso al ejecutable de *MySQL* para *Windows*:
-https://dev.mysql.com/downloads/file/?id=518835

[126] Acceso al ejecutable del conector de *Python*:
-https://www.python.org/ftp/python/3.11.2/python-3.11.2-amd64.exe

[127] Acceso al ejecutable de *Visual Studio Professional*:
-https://my.visualstudio.com/Downloads?q=visual%20studio%202019&wt.mc_id=o~msft~vscom~older-downloads

[128] Vídeo tutorial de YouTube de donde se ha sacado la instalación de *phpMyAdmin*:
-https://www.youtube.com/watch?v=OaMt7Y4rTQI

[129] Acceso a la página oficial de *phpMyAdmin*:
-https://www.phpmyadmin.net/

[130] Acceso a descargar el ejecutable de *phpMyAdmin*:
-https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-all-languages.zip

[131] Información sobre los iconos utilizados en la plataforma y sacados de la aplicación web externa de *Font Awesome*:
-https://fontawesome.com/

[132] Tutoriales sobre como realizar los Canvas en *HTML*, *CSS* y *JavaScript*:
-https://developer.mozilla.org/es/docs/Web/API/Canvas_API/Tutorial
-https://github.com/myTeachingURJC/2020-2021-CSAAI/wiki/S6:-Animaciones-2D#elemento-canvas-dibujando-gr%C3%A1ficos-2d

[133] Vídeo tutorial sobre como hacer el carrusel de imágenes para pasar de una imagen a otra con flechas:
-https://www.youtube.com/watch?v=4FWyJCEQBrs

[134] Información sobre qué es un despliegue de aplicaciones, tipos que hay y como funciona:
-https://www.unir.net/ingenieria/revista/despliegue-aplicaciones-web/#:~:text=El%20despliegue%20de%20aplicaciones%20implica,tener%20acceso%20desde%20cualquier%20lugar.

[135] Vídeo tutorial sobre la descarga de los archivos BAT y VBS y su funcionamiento para ejecutar la aplicación automáticamente al encender el computador del autor:
-https://www.youtube.com/watch?v=6qpPQpowqc8

[136] Información sobre qué son los resultados en un proyecto de investigación o proyecto de Fin de Carrera:
-https://tutfg.es/investigacion-cientifica/#:~:text=Los%20resultados%20de%20una%20investigaci%C3%B3n,o%20cuantitativo)%20que%20hayas%20elegido.

[137] Información sobre el estándar de validez de correos electrónicos de RFC 5322:
-https://docs.aws.amazon.com/es_es/ses/latest/dg/send-email-concepts-email-format.html

[138] Información sobre qué es el código CIF/NIF:
-https://economia3.com/que-es-cif-codigo-identificacion-fiscal/

[139] Información sobre la tecnología *A-Frame*:
-https://www.adictosaltrabajo.com/2017/06/28/aframe-bienvenido-a-webvr/

[140] Imágenes de los tipos de campos de la Arquitectura ECS:
-https://www.adictosaltrabajo.com/2017/06/28/aframe-bienvenido-a-webvr/

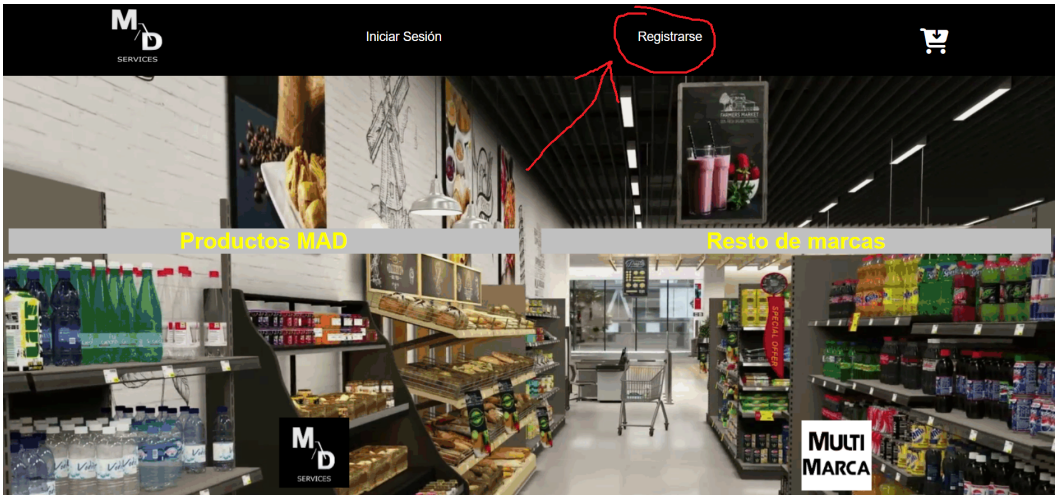
[141] Vídeo sobre el prototipo de la aplicación con RV (Realidad Virtual):
-https://www.youtube.com/watch?v=s3dD9j6fEBs

Apéndice A

Manual de usuario

Registrarse en la aplicación

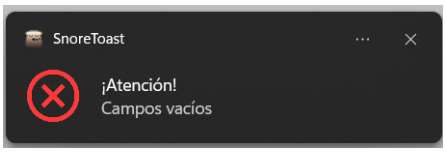
Partiendo del inicio del prototipo en el modo usuario sin registro, vamos a:



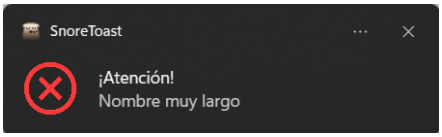
Y elegimos el tipo de registro que queremos llevar a cabo:

1. Como cliente:

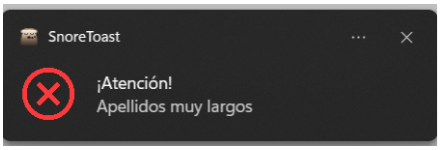
Si dejamos algún campo vacío, aparecerá esta notificación o alerta sonora (funciona igual en todo ámbito del prototipo):



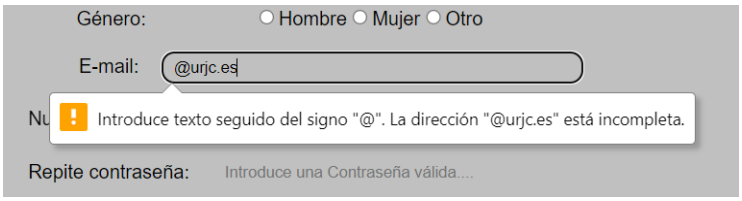
Si el nombre es muy largo, aparecerá esta notificación o alerta sonora:



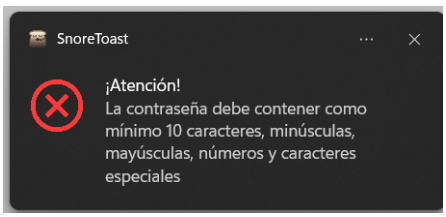
Si el apellido es muy largo, aparecerá esta notificación o alerta sonora:



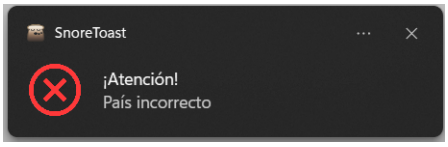
Si el correo electrónico no es válido, directamente no podrás darte de alta y sacará por pantalla lo siguiente (funciona igual en todo ámbito del prototipo):



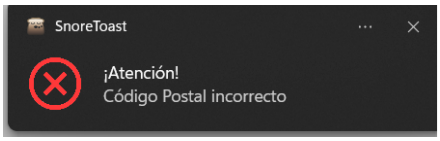
Si la contraseña no es válida, aparecerá esta notificación o alerta sonora (funciona igual en todo ámbito del prototipo):



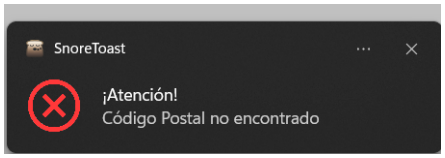
Si el país no existe, aparecerá esta notificación o alerta sonora:



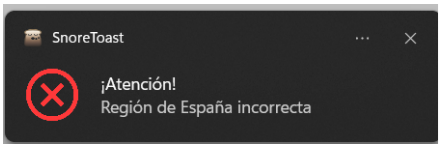
Si el código postal no existe en el país escrito, aparecerá esta notificación o alerta sonora:



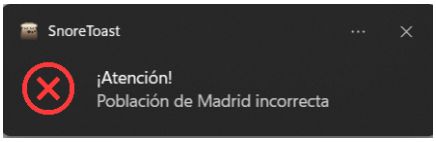
Si el código postal escrito no está asociado a ningún lugar de ese país, aparecerá esta notificación o alerta sonora:



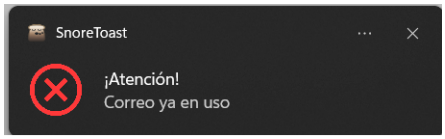
Si la región no está asociada al código postal escrito, aparecerá esta notificación o alerta sonora:



Si la población no está asociada al código postal escrito, aparecerá esta notificación o alerta sonora:



Si se escribe un correo electrónico ya existente, aparecerá esta notificación o alerta sonora (funciona igual en todo ámbito del prototipo):



Si todo es correcto, aparecerá esta notificación o alerta sonora y te redirigirá al inicio:

2. Como empresa:

Si todo es correcto, aparecerá esta notificación o alerta sonora y te redirigirá al inicio:

3. Como miembro:

Debes estar autorizado para poder registrarse como miembro MAD:

Si escribes bien la contraseña, podrás acceder al registro:

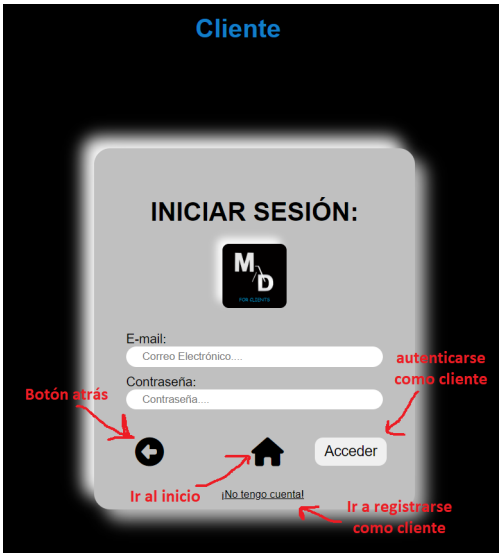
Si todo es correcto, aparecerá esta notificación o alerta sonora y te redirigirá al inicio:

Iniciar sesión en la aplicación

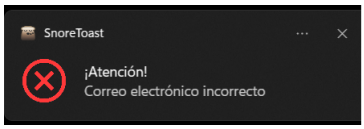
Partiendo del inicio del prototipo en el modo usuario sin registro, vamos a:

Y elegimos el tipo de inicio de sesión que queremos llevar a cabo:

1. Como cliente:

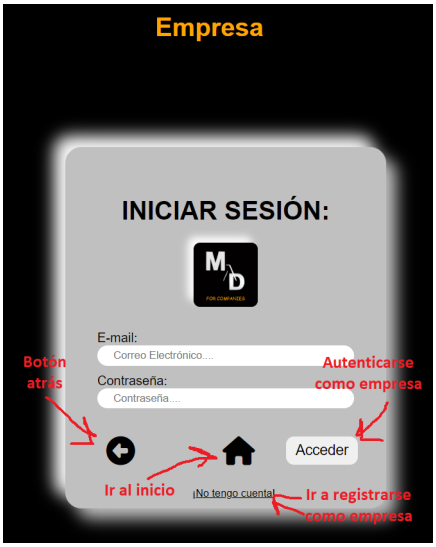


Si el correo electrónico escrito no existe en la plataforma, aparecerá esta notificación o alerta sonora (funciona igual en todos los inicios de sesión del prototipo):



Si todo es correcto, aparecerá esta notificación o alerta sonora y te redirigirá al inicio:

2. Como empresa:



Si todo es correcto, aparecerá esta notificación o alerta sonora y te redirigirá al inicio:

3. Como miembro:

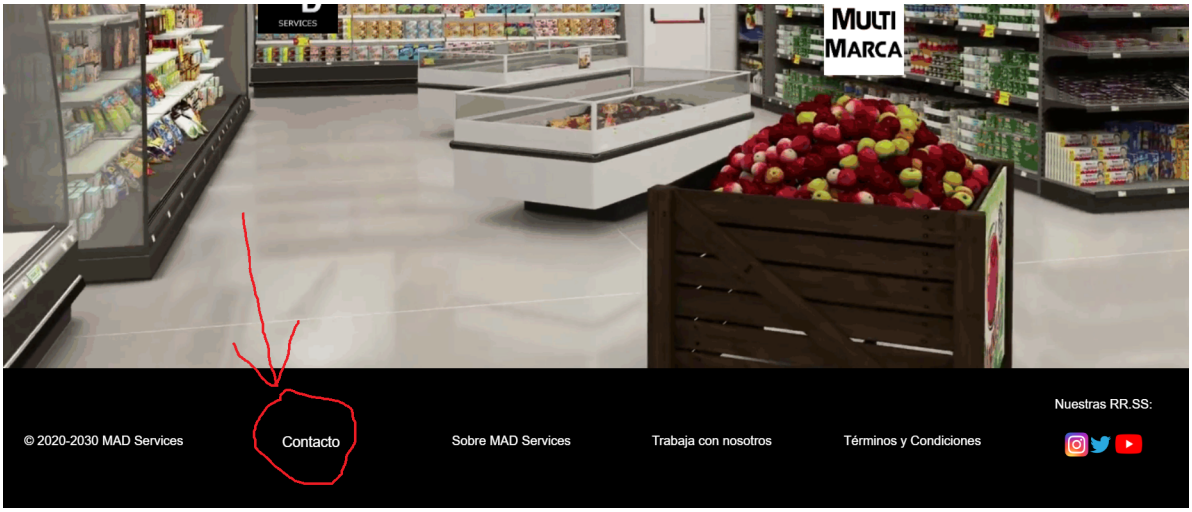
Debes estar autorizado para poder iniciar sesión como miembro MAD:

Si escribes bien la contraseña, podrás acceder al inicio de sesión:

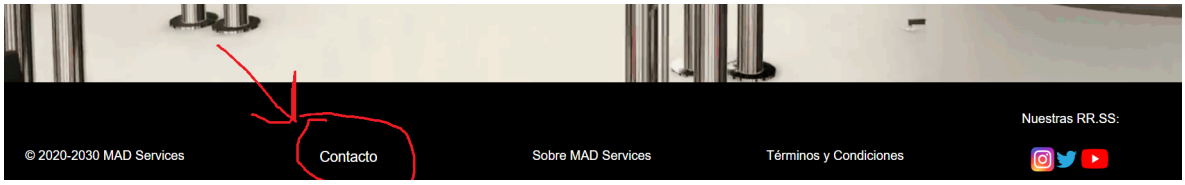
Si todo es correcto, aparecerá esta notificación o alerta sonora y te redirigirá al inicio:

Acceso a Contacto de la aplicación

1. Como cliente y como usuario sin registro:



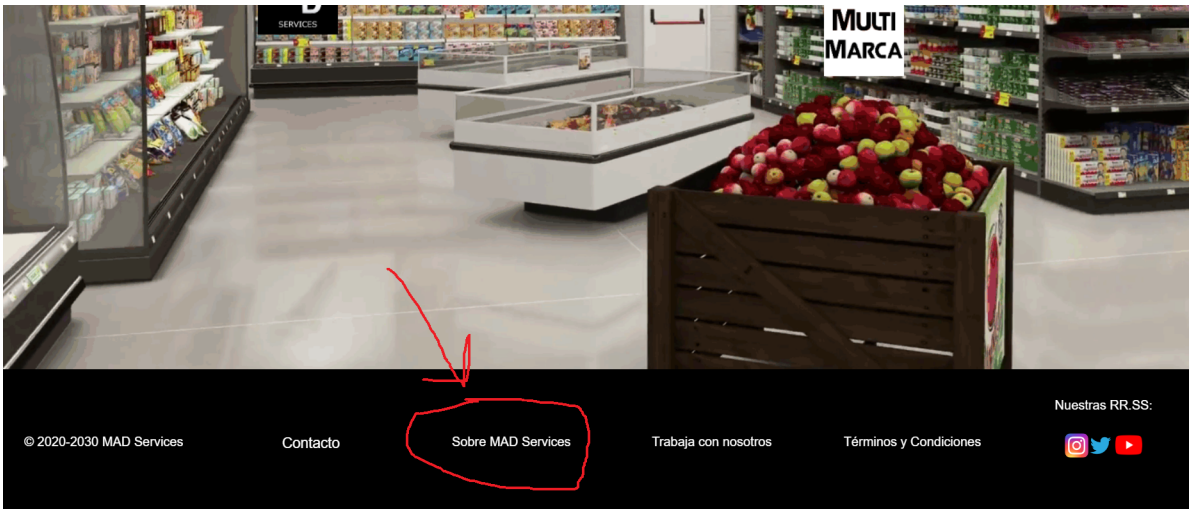
2. Como empresa y como miembro:



***) Su contenido es:**

Acceso a Sobre MAD Services de la aplicación

1. Como cliente y como usuario sin registro:

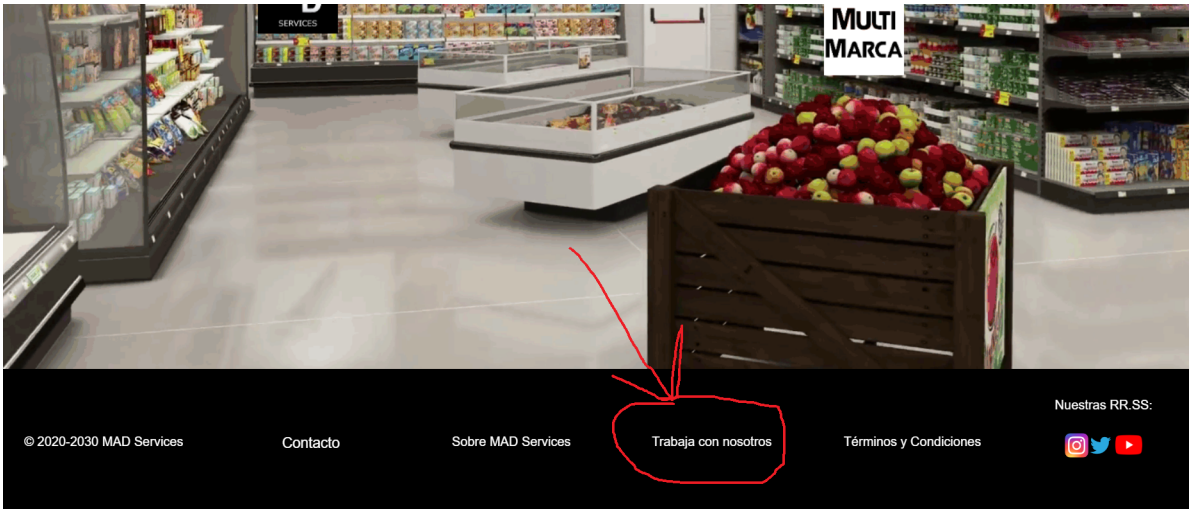


2. Como empresa y como miembro:

***) Su contenido es:**

Acceso a Trabaja con nosotros de la aplicación

1. Como cliente y como usuario sin registro:

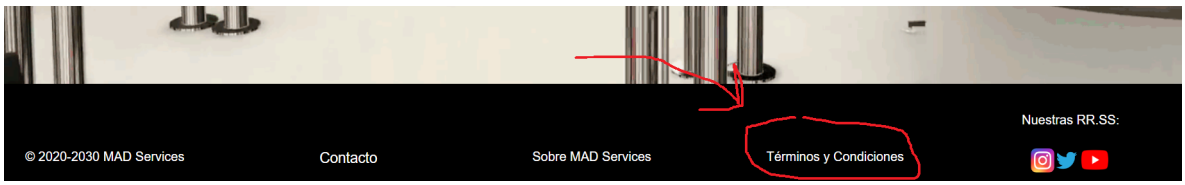


***) Su contenido es:**

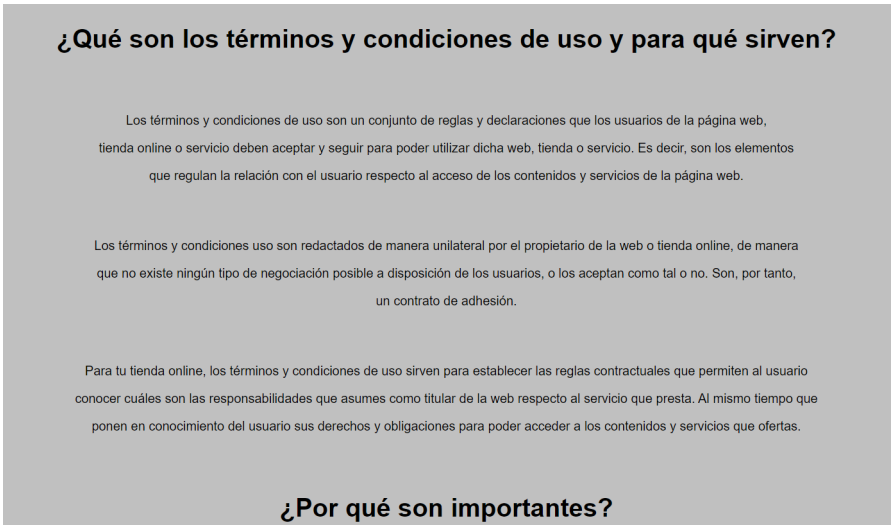
Acceso a Términos y Condiciones de la aplicación

1. Como cliente y como usuario sin registro:

2. Como empresa y como miembro:

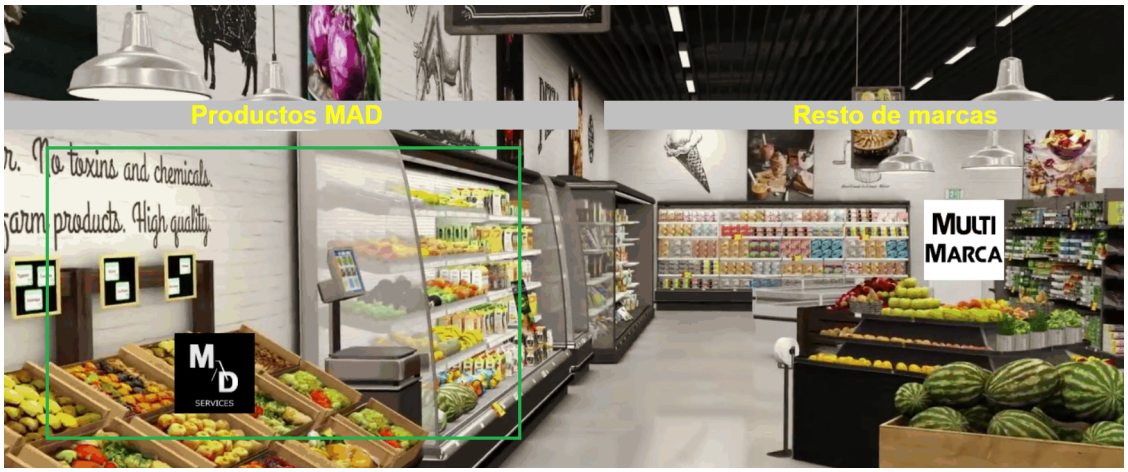


*) Su contenido es:



Acceso a los productos propios de la aplicación

Se parte del inicio que es igual en los tres:



(NOTA: puedes pulsar en cualquier punto dentro del rectángulo verde dibujado en la imagen, no tienes que pulsar únicamente en el logo, ya que éste se mueve rápido por el canvas).

1. Como cliente:

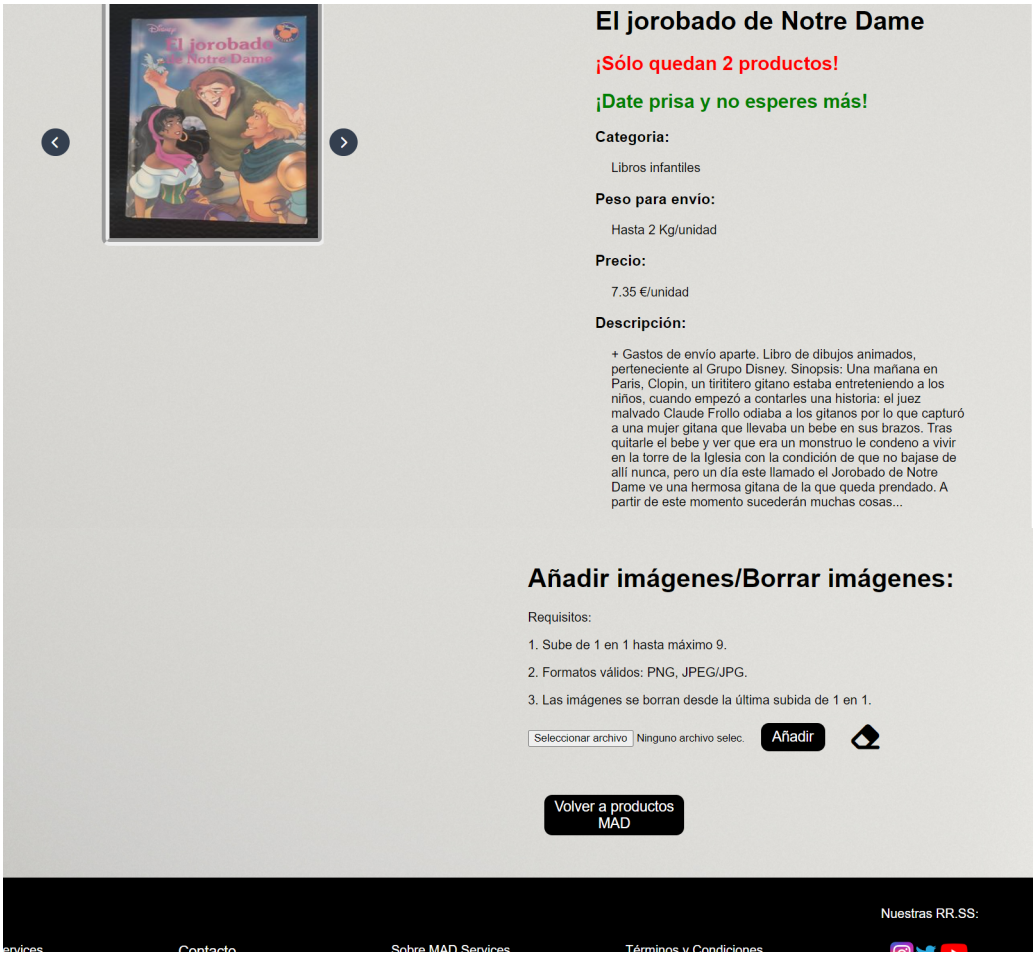
Si le damos a añadir al carrito, aparecerá una notificación o alerta sonora y se añadirá a su respectivo carrito de compra:

Si le damos a expandir, aparecerá la pantalla con el producto completo, con su descripción, más imágenes y otras características:

2. Como miembro:

Aquí en vez de dar a añadir al carrito, hay dos opciones: editar el producto en cuestión o borrarlo definitivamente. Su edición se lleva a cabo en:

Si le damos a expandir, aparecerá la pantalla con el producto completo, con su descripción, más imágenes, otras características y la opción de añadir más imágenes de una en una o borrar las que hay, también de una en una:



3. Como usuario sin registro:

Si le damos a añadir al carrito, no se añadirá a ninguno, ya que para eso hay que estar registrado y, por tanto, te llevará a la pantalla de registro como cliente:

Si le damos a expandir, aparecerá la pantalla con el producto completo, con su descripción, más imágenes y otras características que aparece cuando accedes como cliente.

Acceso a los productos y/o servicios de las empresas registradas en la aplicación

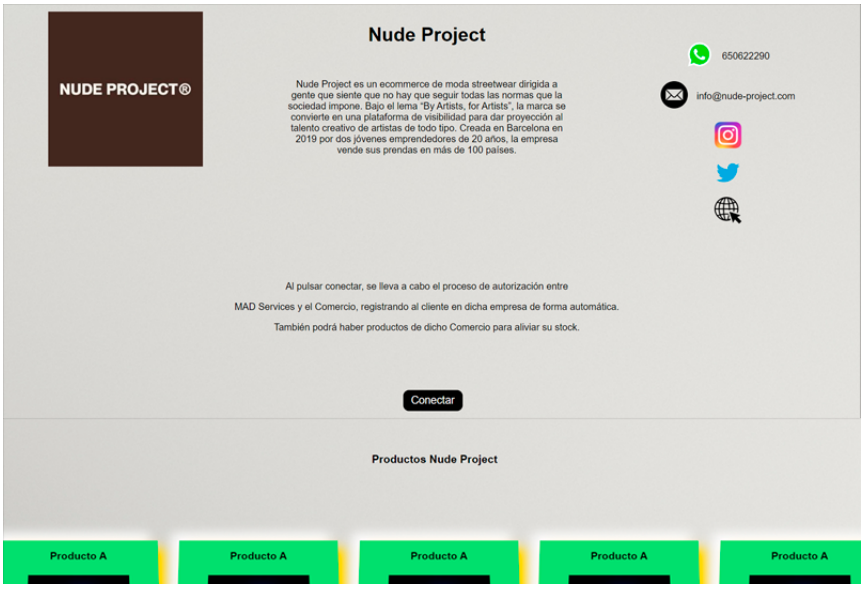
Tanto como **cliente**, **miembro** o **usuario sin registro**, se parte del inicio:

(NOTA: puedes pulsar en cualquier punto dentro del rectángulo verde dibujado en la imagen, no tienes que pulsar únicamente en el logo, ya que éste se mueve rápido por el canvas).

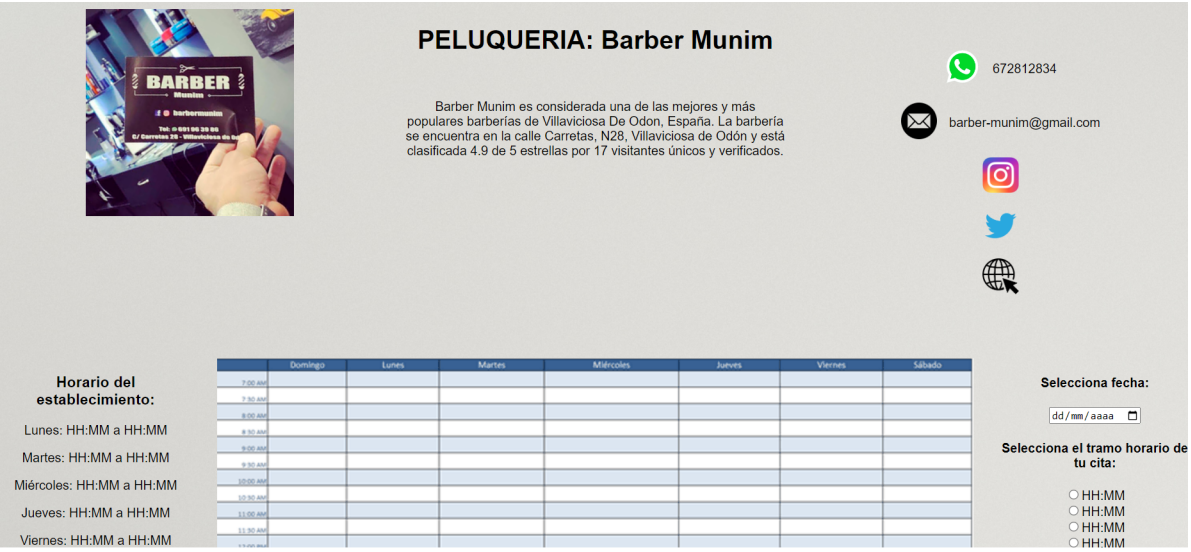
Y se accede a otra pantalla muy similar en los modos de autenticado de cliente y miembro y en el de usuario sin registro.

Se puede acceder a la interfaz diseñada de cuatro tipos de empresas:

Empresa de tipo comercio:



Empresa de tipo peluquería:



Empresa de tipo academia:

Empresa de tipo restaurante:

Con la diferencia de que el cliente es el único que puede contratar, añadir al carrito, pedir cita y/o reservar. Incluso conectar con un comercio electrónico.

Para volver a la pantalla de búsqueda de empresas, hay un botón al final de cada interfaz diseñada en todos los modos de autenticado y en el del usuario sin registro.

2. Como empresa:

En este caso se parte de un inicio diferente:

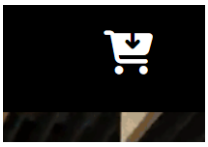
(NOTA: puedes pulsar en cualquier punto dentro del rectángulo verde dibujado en la imagen, no tienes que pulsar únicamente en el logo, ya que éste se mueve rápido por el canvas).

Lo que viene después es igual que en los otros modos de autenticación y en el del usuario sin registro.

Acceso al carrito de la compra de la aplicación

1. Como cliente:

Para acceder a su carrito de la compra, debemos pulsar en el símbolo de la imagen siguiente, que está en la parte superior derecha de la plataforma:



Y aparecerá la pantalla siguiente:

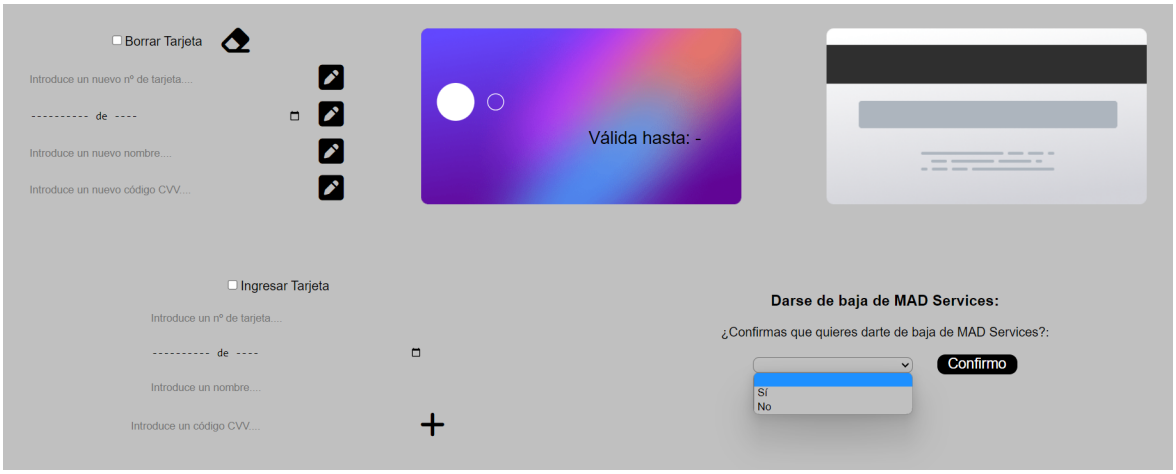
Acceso al perfil o interfaz de la aplicación y su interactividad

Cada modo de autenticación tiene su perfil, en caso del cliente y, su interfaz, en caso del miembro y de la empresa.

1. Como cliente:

Datos del cliente en el perfil y actualización de ellos:

Visualización de su tarjeta bancaria, con la posibilidad de borrarla, ingresar una nueva y actualizar campo a campo. También la opción de darse de baja de la plataforma:



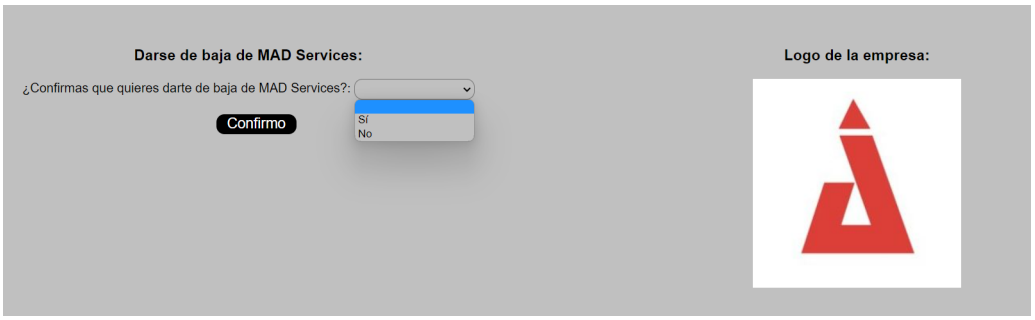
Y por último, la posibilidad de ver sus productos comprados:

2. Como empresa:

Datos de la empresa y campos como la descripción, el contacto, el acceso a las redes sociales y el acceso a la página web, que se visualizan en la interfaz que ven los clientes:

Actualización de los datos de la empresa, de la descripción, del contacto, del nombre de usuario de las redes sociales, de la URL de la página web y del logo de la empresa:

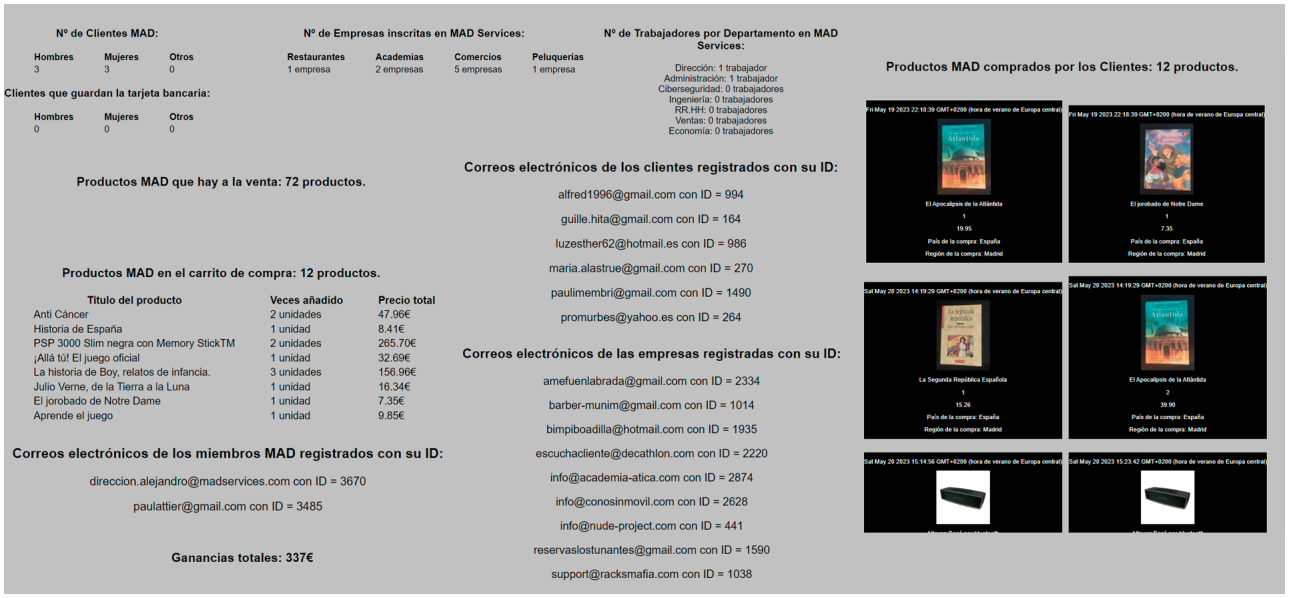
Y la visualización del logo que se muestra en la interfaz que ven los clientes, más la opción de darse de baja de la plataforma:



3. Como miembro:

Datos del miembro MAD y actualización de ellos:

Panel de control de todo miembro MAD:

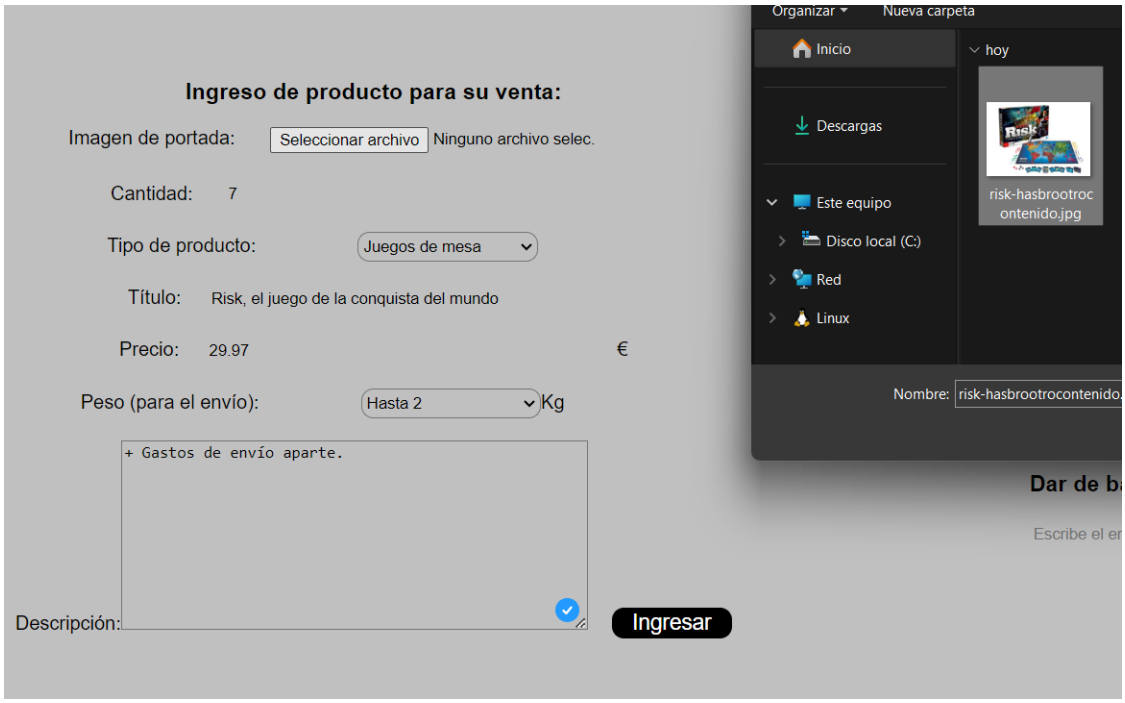


Donde se controla: el número de clientes, según su género, que hay; los clientes, según su género, que guardan su tarjeta bancaria; el número de tipos de empresas que hay; el número de empleados de la plataforma por departamento; los productos MAD que hay a la venta; los productos que han sido añadidos al carrito; la lista de IDs y correos electrónicos de clientes, empresas y miembros o empleados de la plataforma; las ganancias que hay por las ventas; y los

productos MAD que han sido comprados por los clientes.

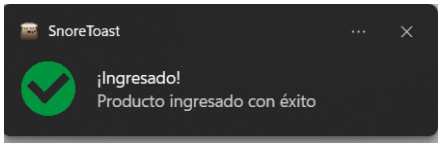
La opción de darse de baja de la plataforma y dar de baja a clientes y empresas, bajo el consentimiento de ellos.

La opción de ingresar nuevos productos MAD:



Donde es obligatorio completar todos los campos, si no te mostrará una notificación o alerta sonora de *Campos vacíos*. Esos campos son: una imagen en formato PNG o JPG/JPEG, la cantidad del producto a vender, el tipo de producto, un título llamativo, un precio, un peso de los que hay y una descripción. Todos los campos son verificados y, si todo es correcto, el

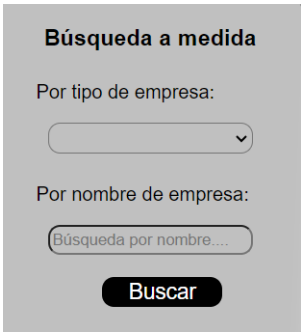
producto se ingresará en la base de datos oficialmente, mostrándose en la búsqueda de productos propios de la plataforma:



Búsqueda a medida de los productos propios de la plataforma

Puedes buscar el producto que quieras a través de tres filtros: el filtro de la búsqueda por el nombre del producto, por el tipo de producto que buscas y/o por el precio que cuesta.

Búsqueda a medida de las empresas registradas en la plataforma

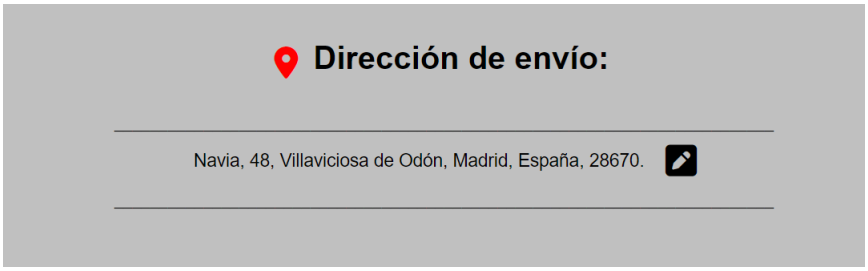


Puedes buscar la empresa a la que quieras acceder, para ver y/o comprar sus productos, o para ver y/o contratar sus servicios, a través de dos filtros: el filtro de la búsqueda por el nombre de la empresa en cuestión y/o por el tipo de empresa diferenciado por colores, donde el color amarillo es para restaurantes, el color rojo para academias, el verde para comercios y el azul para peluquerías.

Proceso de compra y pago por parte del cliente

En la pantalla del proceso de compra y pago destacan tres partes:

1. La primera es aquélla que muestra la localización del envío de la compra, con opción a editarla:

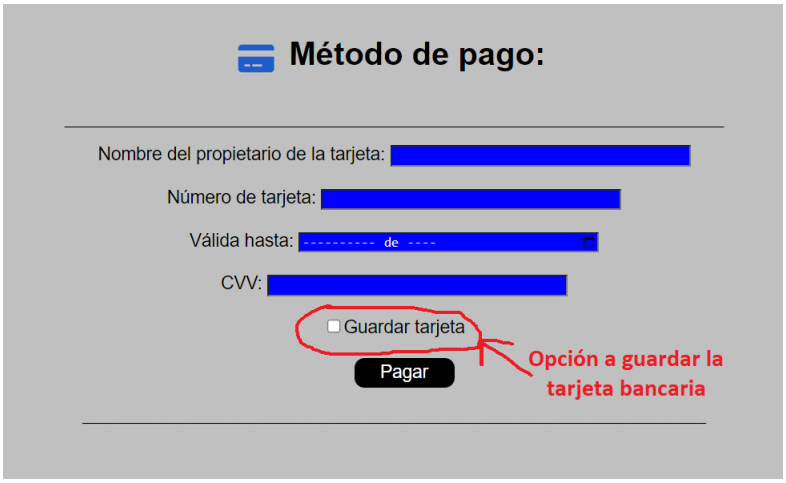


2. La segunda es aquélla que muestra la cantidad de productos y/o servicios a comprar y/o contratar y, el precio total a pagar:

3. Y la tercera es aquélla donde pueden darse dos casos:

Caso 1. Que muestre el método de pago con el que se va a llevar a cabo el proceso de compra:

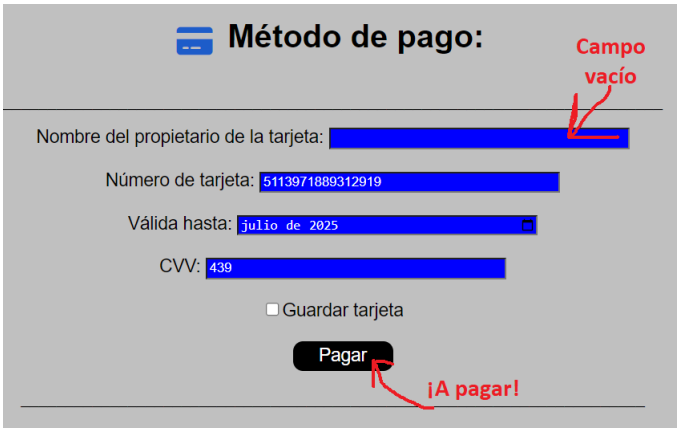
Caso 2. Que se tenga que ingresar el método de pago con el que se va a llevar a cabo el proceso de compra, con la posibilidad de guardarlo o no:



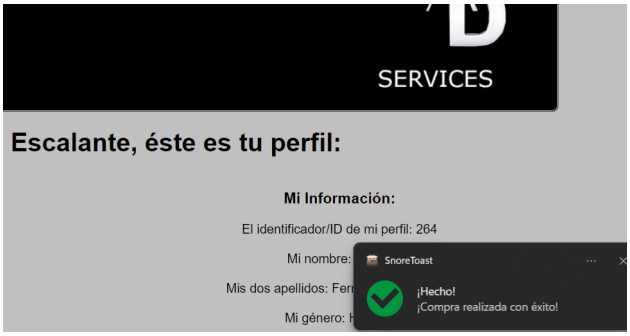
Si estando en el caso 2, se le da pagar sin introducir una tarjeta bancaria válida, o dejando

campos vacíos, aparecerá la siguiente notificación o alerta sonora:

Lo que sí podrías dejar vacío es el *nombre del propietario de la tarjeta*, ya que la propia aplicación usaría el nombre que el cliente usó para registrarse:

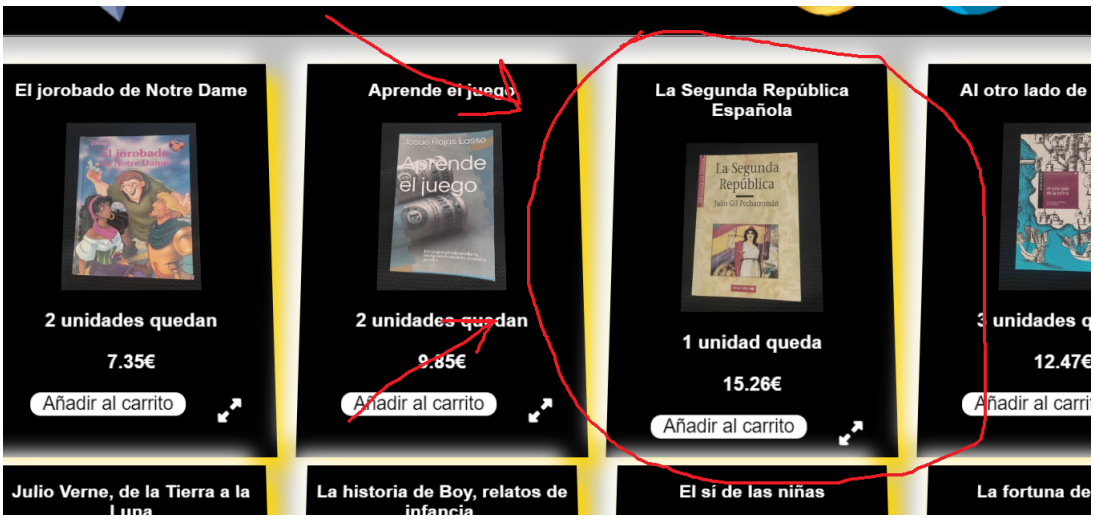


Al darle a pagar, sacará una alerta o notificación de que la compra se ha realizado y te redirigirá a tu perfil, para que veas todas tus compras realizadas, desde la primera a la última (las compras se muestran porque se obtienen de la base de datos):



Éste es el producto comprado de 15.26€:

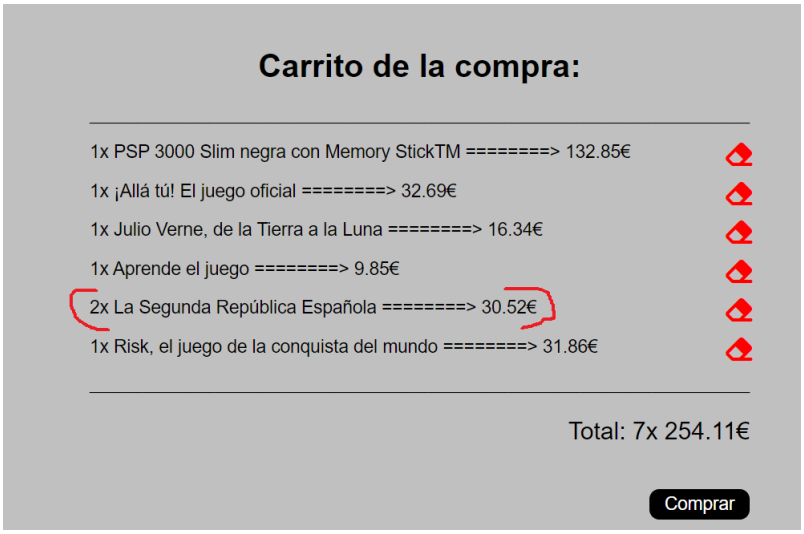
Si os fijáis bien, en la segunda imagen del *Acceso a los productos propios de la aplicación*, ponía que había dos productos a la venta del producto comprado aquí arriba y ahora queda un producto:



A.0.1. Dilema práctico

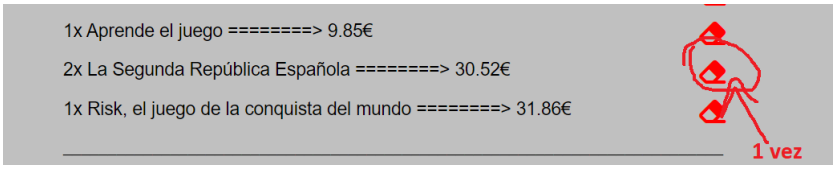
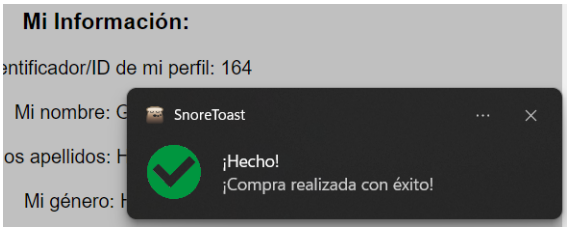
Para concluir, si ahora otro usuario quiere comprar y pagar los dos productos que había de este producto comprado y que tenía y sigue teniendo ambos en su carrito de la compra antes de que este usuario comprara y pagara uno de ellos, la plataforma no le dejaría comprarlos, ya que no puede comprar lo que no existe, al quedar sólo uno de los dos en la bolsa de productos. Para ello, tendría que eliminar uno y así actualizar su carrito de la compra de productos oficialmente disponibles.

Carrito de este otro usuario:



Aunque el resto de productos los pueda comprar, como uno de ellos está marcado con una cantidad de dos cuando sólo queda uno, si quiere llevar a cabo el pago de la compra, debe actualizar el carrito de la compra:

Por tanto, eliminaría uno de ellos, a través del icono de la *goma de borrar*:



Y finalmente, ya podría comprar todo sin ningún problema: