HELP! Aplicación web sobre información del pequeño comercio.

Nombre Estudiante
Rebeca Huerta Rubio

Máster Universitario en Desarrollo de sitios y aplicaciones web
Informática, Multimedia y Comunicación

Tutor/a de TF
Bruno Francisco Orcha García

Profesor/a responsable de la asignatura
César Pablo Córcoles Briongos

ENERO 2023

Universitat Oberta de Catalunya

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Ficha del Trabajo Final

Título del trabajo:	*HELP! Aplicación web sobre información del pequeño comercio.*
Nombre del autor/a:	*Rebeca Huerta Rubio*
Nombre del Tutor/a de TF:	*Bruno Francisco Orcha García*
Nombre del/de la PRA:	*César Pablo Córcoles Briongos*
Fecha de entrega:	*01/2023*
Titulación o programa:	*Máster Universitario en Desarrollo de sitios y aplicaciones web*
Área del Trabajo Final:	*Informática, Multimedia y Comunicación*
Idioma del trabajo:	*Castellano*
Palabras clave	aplicación web, búsqueda de productos, horarios, pequeño comercio.

Resumen del Trabajo

El objetivo de este proyecto será el desarrollo y documentación de una aplicación web que permita a los pequeños comercios disponer de una herramienta para tener actualizado sus horarios y días festivos abiertos, para que sus clientes puedan consultarlo en cualquier momento de forma rápida.
También ayudara a los clientes a encontrar un determinado producto en los comercios de proximidad.
La aplicación utilizara Laravel como Backend con la información almacenada en una base de datos MySQL y para la parte de Frontend usare Angular.

Abstract

The objective of this project will be the development and documentation of a web application that allows small businesses to have a tool to update their open hours and holidays, so that their customers can quickly consult it at any time.

It will also help customers find a certain product in local stores.

The application will use Laravel as Backend with the information stored in a MySQL database and for the Frontend part I will use Angular.

ÍNDICE

Lista de Figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Como todos sabemos el auge del comercio electrónico ha incrementado en los años de la pandemia, como consecuencia el pequeño comercio ha sido durante estos años el gran olvidado. Este factor ha provocado que una buena parte de los consumidores elijan cada vez menos este tipo de establecimientos para realizar sus compras.

También hay que tener en cuenta que las grandes superficies tienen más influencia, medios, comunicación sobre los clientes que los pequeños comercios.

Por ello el pequeño comercio prefiere centrar sus esfuerzos en la fidelización con su cliente para ver crecer su negocio.

Figura 1 – Imagen comercio de proximidad

Por esto, ahora es el momento de estar al lado del pequeño comercio y ayudarles a superar estos años tan difíciles. Para ello, tenemos que volver a salir a la calle y visitar estos pequeños comercios de nuestros barrios, por muchas razones:

- Los pequeños comercios suponen una gran parte del tejido empresarial de nuestro municipio. Ateniendo a los datos, las empresas que más trabajadores contratan son los pequeños negocios de barrio, sin embargo, son los negocios que más sufren debido a su desventaja con las grandes superficies con las que no pueden competir en igualdad de condiciones.

- La gran fortaleza del pequeño comercio es su calidez y cercanía en el trato con el cliente, confiar en los profesionales que saben.

- Otro factor muy importante es la vida de calle, y a ella colabora el hecho de que los pequeños comercios estén vivos. La falta de estos en una municipio o barrio se refleja en una peor calidad de vida e incluso inseguridad, y por eso es muy importante incentivarlo.

- Los pequeños comercios suelen contar con determinados productos más especializados que las grandes superficies.

- La mayoría de pequeños comercios utilizan productos en sus establecimientos adquiridos también a empresas locales.

- Cuidas el medio ambiente, no tener que mover tu coche o utilizar el transporte público, ahorras y no contaminas. Con el comercio electrónico ha incrementado el uso de plásticos y medios de transporte para repartirlo.

El pequeño comercio es vida para un municipio, y es una pieza importante de una actividad económica que da empleo a muchas personas. Su colaboración al desarrollo empresarial de un municipio es fundamental. Su supervivencia está ahora en nuestra mano. Nuestros hábitos de consumo, y sobre todo nuestra responsabilidad a la hora de contribuir a la generación de riqueza, es fundamental para que el sector sea capaz de salir adelante.

Quiero crear esta aplicación para ayudar al pequeño comercio a aumentar la fidelidad con sus clientes. Que el cliente pueda consultar de una manera rápida y sencilla, que horarios tiene su establecimiento de confianza, que días tiene de apertura, que producto puede comprar en un comercio cercano.

Que el pequeño comercio tenga la oportunidad de comunicarse con sus clientes de manera rápida y eficaz para avisar de cualquier cambio de horario, días de apertura, un nuevo producto a la venta, etc.

Información consultada en webs [1], [2]
Imagen obtenida de la web [6]

1.2. Objetivos del Trabajo

Con este proyecto lo que quiero conseguir es dar una solución a la problemática mencionada en el apartado anterior y crear un vínculo más estrecho entre el pequeño comercio y el cliente.

Help! ofrece al pequeño comercio un sistema sencillo y ágil de utilizar, para tener actualizado sus horarios, días festivos abiertos y mostrar al cliente un listado de los productos que ofrece. El cliente podrá comunicarse mediante mensajes con el comercio, para preguntarte dudas, si dispone de un producto determinado, etc.

A continuación, muestro un listado con los objetivos del proyecto, los principales son que tengo que conseguir realizar a tiempo para la presentación del proyecto. Los objetivos secundarios los dejo para el final y si tuviera tiempo poder realizarlos.

Objetivos principales:

- ✓ Desarrollar una aplicación para el pequeño comercio donde pueda crear una cuenta para tener sus datos actualizados diariamente sobre dirección, horarios, días de apertura, productos más demandados que ofrece.
- ✓ Ofrecer a los clientes una aplicación sencilla y rápida para consultar cualquier horario o buscar algún producto de algún establecimiento cercano.
- ✓ El cliente no tendrá la necesidad de darse de alta para poder consultar o hacer búsquedas.
- ✓ El cliente si tendrá que darse de alta en la aplicación para poder comunicarse con el comercio y guardar cuáles son sus comercios favoritos.
- ✓ El comercio podrá enviar mensajes a los clientes que tenga su comercio como favorito, con cualquier modificación de sus horarios, días de apertura, etc.
- ✓ La aplicación tendrá 3 perfiles, administrador, comercio y cliente.

Objetivos secundarios:

- ✓ Crear un sistema de envío de mensajes directos entre el comercio y el cliente.
- ✓ Crear un apartado de opiniones que puedan dejar los clientes sobre un establecimiento concreto.
- ✓ Aumentar la usabilidad de las interfaces de usuario.
- ✓ Comprobar la seguridad del producto y los datos que utiliza, como confidencialidad o privacidad.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Según el documento “Guía transversal sobre la Competencia Ética y Global”, la competencia de compromiso ético y global (CCEG) está definida a nivel de Master como:

> “Actuar de manera honesta, ética, sostenible, socialmente responsable y respetuosa con los derechos humanos y la diversidad, tanto en la práctica académica como en la profesional, y diseñar soluciones para mejorar estas prácticas.”

Hay que considerar, pues, que aborda tres grandes dimensiones:

I. Sostenibilidad

Creo que mi TFM tendrá un resultado positivo sobre aspectos de sostenibilidad medioambiental y/o huella ecológica (consumo/ahorro energético, residuos, contaminación, agotamiento de materias primas...). Con mi aplicación lo que quiero conseguir es que el cliente no realice compras online y salga a comprar al pequeño comercio.

Desplazarse a un centro comercial o una gran superficie lleva consigo en muchas ocasiones el desplazamiento en coche, lo que implica una mayor contaminación. Del mismo modo, cuando compras online, el producto puede llegar a tener que viajar cientos de kilómetros hasta el domicilio, con la consecuencia de todo el plástico y materiales que se utilizan para envolver esos paquetes.

II. Comportamiento ético y responsabilidad social (RS)

Este proyecto no tendrá ninguno impacto negativo en aspectos ético-sociales ni en ningún marco normativo/legislación, como puede ser datos, privacidad, laboral, propiedad intelectual, seguridad de las personas. No pone en riesgo ningún puesto de trabajo, yo creo que incluso puede mejorar en ese aspecto.

Puede contribuir al bien común de la sociedad, al colaborar que los pequeños comercios sigan abiertos eso puede aportar que un municipio o barrio tenga una mejor calidad de vida y seguridad en sus calles.

III. Diversidad (género, entre otros) y derechos humanos

El fin de este proyecto no tiene ningún impacto negativo en aspectos de género, diversidad (raza, religión, orientación sexual, funcional, etnia, ideología...) o derechos humanos.

El uso de la aplicación será accesible para distintos tipos de colectivos, ya que será una aplicación gratuita y solo se necesitará un móvil y conexión de datos para consultar la información.

No se pedirán datos personales sobre raza, religión, orientación sexual, funcional, etnia, ideología...

Se podrá utilizar sin registrarnos como clientes, sin necesidad de dar ningún dato personal. En caso que nos demos de alta como cliente, se respetará la legislación sobre datos, privacidad, laboral, propiedad intelectual y seguridad de las personas.

1.4. Enfoque y método seguido

El proyecto se basa en una aplicación web desarrollada desde cero, desconozco si actualmente hay algo parecido en apps.

Actualmente si un cliente quiere comprar un producto y desconoce donde hacerlo, seguramente se irá a Google para realizar una búsqueda de producto.

Cuando haces una búsqueda en Google, por ejemplo, “comprar paquete arroz alicante”, te devuelve un resultado no muy convincente, te muestra varios restaurantes donde hacen arroces, también te muestra varios comercios donde venden arroz, pero todas grandes superficies, tipo Al Campo, Mercadona, Carrefour, El Corte Ingles… El pequeño comercio de barrio que vende arroz al lado de tu casa, no aparece por ningún sitio.

Tendremos que definir un conjunto de estrategias para las diferentes etapas del desarrollo de un producto para introducirlo en el mercado. Tenemos que ver cuál es nuestro público objetivo, sus necesidades, si ya hay creados productos parecidos, las funcionalidades que va a tener nuestro producto.

1.5. Planificación del Trabajo

Según el documento Módulo 2 “El trabajo final como un proyecto”: Un proyecto es un conjunto de actividades llevado a cabo durante un tiempo por un conjunto de personas para crear un producto, servicio o resultado único. En este caso, en particular para el TFM, tendremos que distribuir las tareas y el tiempo según acordes a la disponibilidad de una única persona.

Según la clasificación del PMBOK, el proyecto se divide en cinco etapas o grupos de procesos.

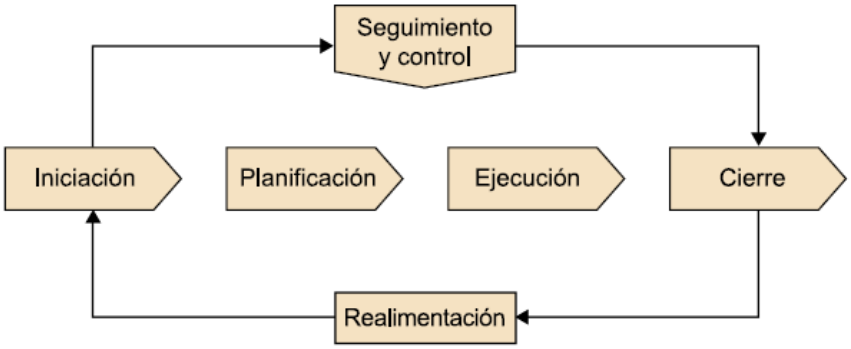


Figura 2 – Ciclo de vida de un proyecto

- Iniciación: elección del tema de trabajo
- Planificación: plan de trabajo
- Ejecución: desarrollo del trabajo: entregables parciales
- Seguimiento y control: desarrollo del trabajo: entregables parciales
- Cierre: entregables finales

Como se ha mencionado en el Módulo 1: “Introducción al trabajo final”, el trabajo final se puede dividir en las siguientes etapas:

- La elección del tema.
- La planificación del trabajo, que da lugar a la primera entrega: un plan de trabajo.
- El desarrollo del trabajo, durante el que se producen varias entregas parciales.
- La entrega final, que incluye tres elementos:
  - Un producto
  - Una memoria
  - Una presentación
- La defensa del trabajo ante un tribunal.

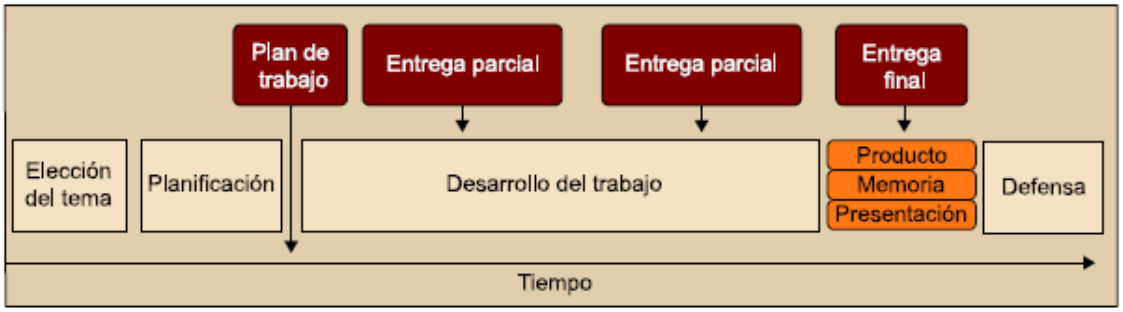


Figura 3 – Etapas de la elaboración del trabajo final

Las herramientas que voy a utilizar para elaborar el plan de trabajo, serán la tabla de hitos y el diagrama de Gantt.

En la tabla de hitos se indican las fechas de inicio y final de los entregables. La tabla de hitos es una buena herramienta de planificación, seguimiento y control de nuestro proyecto.

Este proyecto se divide en cuatro fases denominadas PEC, y a su vez en pequeñas tareas que se irán realizando durante todo el desarrollo del proyecto hasta su entrega final.

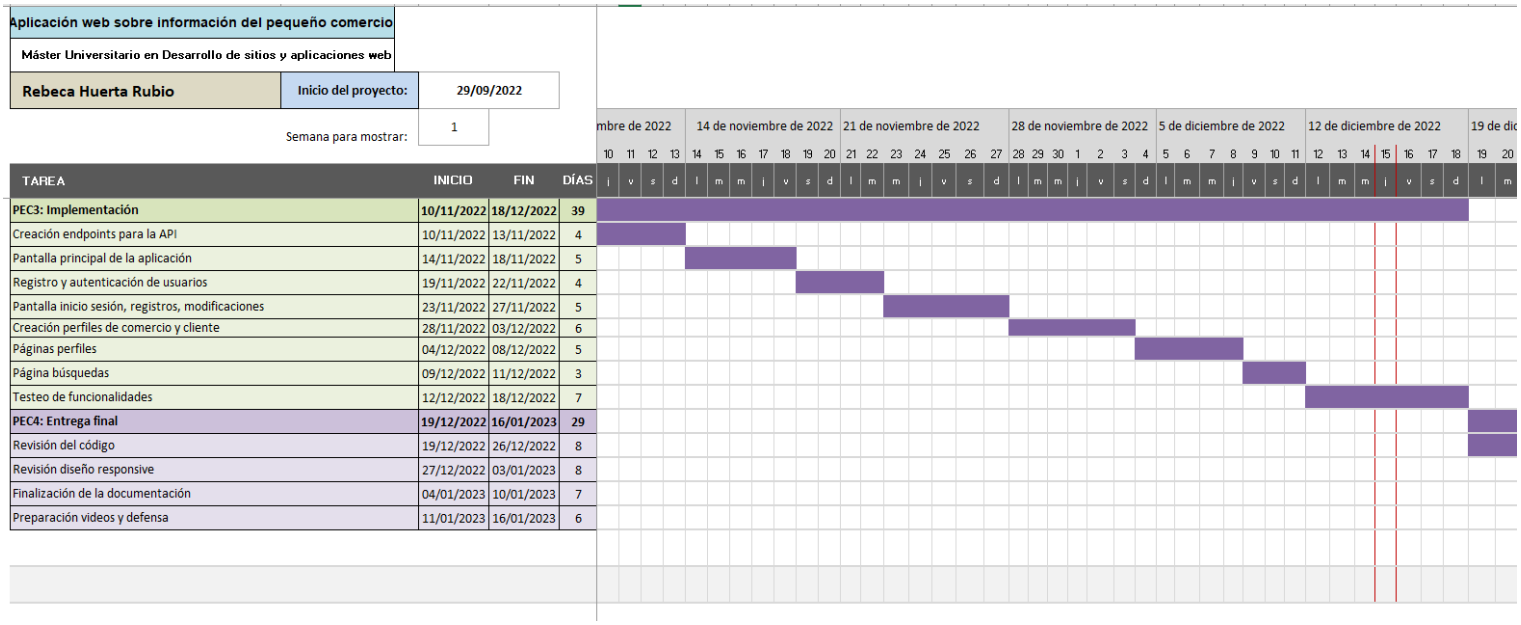
A continuación, he creado una tabla de hitos basándome en las entregas de cada una de las PEC, donde cada una tiene una fecha inicio y fin. Cada una de estas PEC las he dividido en subtareas y he puesto unas fechas estimadas de tiempo.

Nombre	**Duración**	**Inicio**	**Fin**
PEC1: Definición formal del proyecto	14 días	28-09-2022	11-10-2022
Elección del tema	4 días	29-09-2022	02-10-2022
Contexto y justificación del Trabajo	1 días	03-10-2022	03-10-2022
Definición de objetivos	2 días	04-10-2022	05-10-2022
Definición del impacto en sostenibilidad, ético-social y de diversidad	1 días	06-10-2022	06-10-2022
Enfoque y método seguido	3 días	07-10-2022	09-10-2022
Planificación del Trabajo	2 días	10-10-2022	11-10-2022
PEC2: Diseño	29 días	12-10-2022	09-11-2022

Definición de diagramas y detalle de casos de uso	6	12-10-2022	17-10-2022
Estudio de la usabilidad	3	18-10-2022	20-10-2022
Estudio de la arquitectura de la aplicación	3	21-10-2022	23-10-2022
Análisis de mercado	2	24-10-2022	25-10-2022
Prototipos	5	26-10-2022	30-10-2022
Inicio del desarrollo	10	31-10-2022	09-11-2022
PEC3: Implementación	39 días	10-11-2022	18-12-2022
Creación endpoints para la API	4	10-11-2022	13-11-2022
Pantalla principal de la aplicación	5	14-11-2022	18-11-2022
Registro y autenticación de usuarios	4	19-11-2022	22-11-2022
Pantalla inicio sesión, registros, modificaciones	5	23-11-2022	27-11-2022
Creación perfiles de comercio y cliente	6	28-11-2022	03-12-2022
Páginas perfiles	5	04-12-2022	08-12-2022
Página búsquedas	3	09-12-2022	11-12-2022
Testeo de funcionalidades	7	12-12-2022	18-12-2022
PEC4: Entrega final	29 días	19-12-2022	16-01-2023
Revisión del código	8	19-12-2022	26-12-2022
Revisión diseño responsive	8	27-12-2022	03-01-2023
Finalización de la documentación	7	04-01-2023	10-01-2023
Preparación videos y defensa	6	11-01-2023	16-01-2023

Figura 4 – Tabla de hitos sobre la planificación del trabajo

Con el diagrama de Gantt se permite una planificación más detallada, basada en tareas, actividades y volúmenes de trabajo. Permite representar visualmente el inicio y final de cada tarea y, como información adicional, las relaciones de precedencia entre las tareas.



Aplicación web sobre información del pequeño comercio		
Máster Universitario en Desarrollo de sitios y aplicaciones web		
Rebeca Huerta Rubio	Inicio del proyecto:	29/09/2022
	Semana para mostrar:	1

TAREA	INICIO	FIN	DÍAS
PEC3: Implementación	**10/11/2022**	**18/12/2022**	**39**
Creación endpoints para la API	10/11/2022	13/11/2022	4
Pantalla principal de la aplicación	14/11/2022	18/11/2022	5
Registro y autenticación de usuarios	19/11/2022	22/11/2022	4
Pantalla inicio sesión, registros, modificaciones	23/11/2022	27/11/2022	5
Creación perfiles de comercio y cliente	28/11/2022	03/12/2022	6
Páginas perfiles	04/12/2022	08/12/2022	5
Página búsquedas	09/12/2022	11/12/2022	3
Testeo de funcionalidades	12/12/2022	18/12/2022	7
PEC4: Entrega final	**19/12/2022**	**16/01/2023**	**29**
Revisión del código	19/12/2022	26/12/2022	8
Revisión diseño responsive	27/12/2022	03/01/2023	8
Finalización de la documentación	04/01/2023	10/01/2023	7
Preparación videos y defensa	11/01/2023	16/01/2023	6

Figura 5 – Diagrama de Gantt sobre la planificación del trabajo

2. Herramientas de desarrollo

Software		
Base de Datos	MySQL 10.4.24	XAMPP 7.4.30
Interfaz BD	PhpMyAdmin 5.2.0	
Lenguaje Backend	PHP 8.1.10	
Servidor Web	Apache 2.4.54	
IDE Desarrollo	Visual Studio Code 1.72.2	
Pruebas Api	Postman	
Repositorio Git	GitHub	
Framework Backend	Laravel 9.19	
Framework Frontend	Angular, node, npm	
Flowmapp	Sitemap, Diagramas de Flujo	
Creately	Diagramas de Caso de Uso	
Excalidraw	Prototipos	

Figura 6 – Herramientas de desarrollo

3. Metodologías

Las metodologías de desarrollo de software se utilizan como herramientas para conseguir que el proyecto en equipo de realice de una manera organizada y productiva.

Estas metodologías suelen tener una serie de métodos y técnicas organizativas para la creación y diseño de un software. Estos métodos hacen que los equipos se organicen de una manera más eficiente, teniendo en cuenta la dificultad del proyecto, presupuestos, equipo disponible, etc.

Para escoger nuestra metodología adecuada tenemos que tener en cuenta varias cosas, el tiempo disponible que tenemos, los recursos de los que disponemos, número de personas para el equipo.

Las metodologías ágiles son las que más se emplean en la actualidad para el desarrollo de software. Como pueden ser Scrum o Kanban.

En nuestro caso, al realizar el proyecto una persona sola es más difícil aplicar una metodología de forma estricta, por esto he decidido elegir una metodología de desarrollo en cascada.

El desarrollo del modelo en cascada se atribuye al teórico de la informática Winston W. Royce.

El modelo en cascada de cinco niveles, basado en las propuestas de Winston W. Royce, divide los procesos de desarrollo en las siguientes fases de proyecto:

Figura 7 – El modelo en cascada de cinco niveles

1. **Análisis:** planificación, análisis y especificación de los requisitos.
2. **Diseño:** diseño y especificación del sistema.
3. **Implementación:** programación del software, la búsqueda de errores y las pruebas unitarias.
4. **Verificación:** integración de sistemas, pruebas de sistema y de integración.
5. **Mantenimiento:** entrega, mantenimiento y mejora.

En las ampliaciones de esta metodología se añaden funciones iterativas al modelo básico como, los saltos hacia atrás, que permiten comparar los resultados de cada una de las fases con los resultados obtenidas en la fase anterior, de modo que se puedan verificar.

Información consultada en la web [3]

4. Arquitectura de la aplicación

La aplicación consta de 3 partes:

- ✓ Base de datos MySQL donde se almacenará la información necesaria para el funcionamiento de la aplicación.
- ✓ Backend, para conectarnos a la base de datos y responder a las llamadas de los usuarios, mediante una APIRest.
- ✓ Frontend, que realizará las llamadas a la base de datos y mostrará los resultados en la pantalla de los usuarios.

En la implementación se utilizará una APIRest que se comunica a través de solicitudes HTTP para ejecutar funciones de base de datos estándar como, crear, leer, actualizar y suprimir registros dentro de un recurso. El formato de los datos será en JSON, ya que es de fácil lectura y tiene una velocidad de procesamiento alta.

Para la implementación del Backend, se utilizará Laravel que es un framework de PHP del tipo MVC (Modelo-Vista-Controlador). Laravel crea un entorno de trabajo y proporciona herramientas a los desarrolladores para ayudarles a desarrollar en PHP sus aplicaciones web.

Laravel es sencillo de utilizar y aprender, tiene una documentación extensa que el desarrollador puede utilizar y dispone de una gran cantidad de librerías y paquetes.

Se utilizará Json Web Token (JWT) para crear un método de autenticación en servicios API, para que nuestra conexión de nuestro Backend y el cliente sean seguras. El cliente envía su usuario y contraseña, la API le retorna un token que enviará en todas las peticiones, para que ésta compruebe que tiene acceso a las acciones que se quieran realizar.

Para la implementación del Frontend, se utilizará Angular que es un Framework de JavaScript de código abierto escrito en TypeScript.

Angular puede ejecutarse en la mayoría de los navegadores web y dispositivos móviles. Una de sus principales ventajas es que Angular enlaza JavaScript y HTML, el código de ambos está sincronizado, lo que ahorra mucho tiempo para los desarrolladores web.

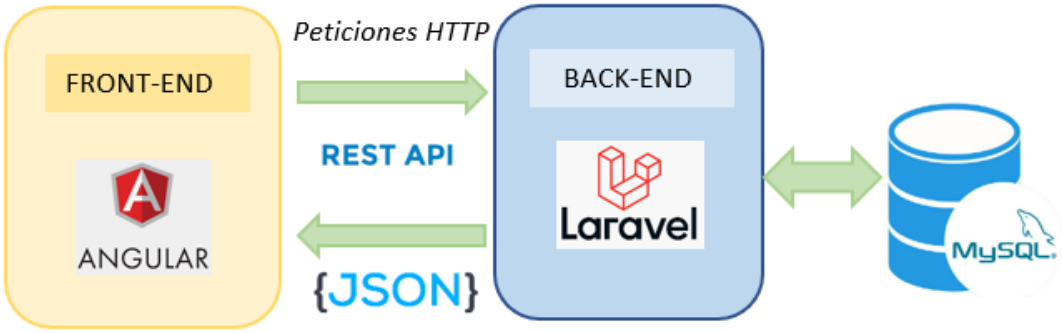


Figura 8 – Arquitectura de la aplicación

5. Diagramas UML

5.1. Base de Datos, diagrama entidad-relación

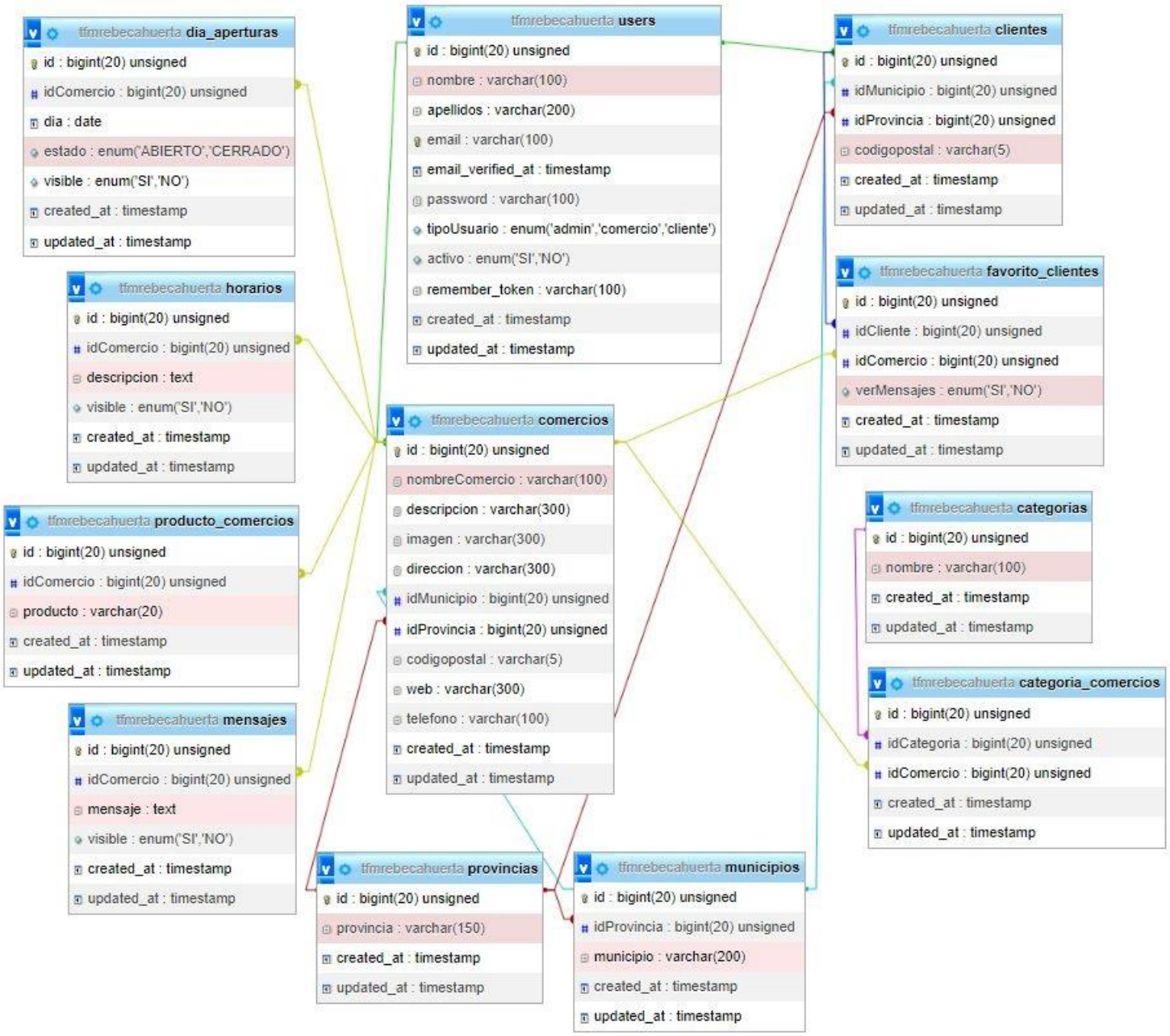


Figura 9 – Diagrama Entidad-Relación

users: esta tabla contiene los datos de acceso de los usuarios, de los tres tipos, administrador, comercio y cliente.

comercios: datos de los comercios que se han registrado en la aplicación, con su dirección, municipio y datos más relevantes del comercio. Está relacionada con la tabla users mediante id.users = id.comercios.

clientes: datos de los clientes que se han registrado en la aplicación. Está relacionada con la tabla users mediante, id.users=id.clientes.

provincias: datos de todas las provincias de España. Está relacionada con la tabla clientes y comercios por id.provincias=idProvincia.comercios y id.provincias=idProvincia.municipios.

municipios: datos de todos los municipios de España. Está relacionada con la tabla provincias por idProvincia.municipios=id.provincias y con la tabla clientes y comercios por id.municipios=idMunicipio.comercios y id.municipios=idMunicipio.clientes.

categorías: tipos de categorías que pueden tener los comercios datos de alta, estos datos serán dados de alta por el administrador de la aplicación.

categorias_comercios: relación del comercio con una o varias categorías. Está relacionada con la tabla categorías por idCategoria.categorias_comercios=id.categorias y con la tabla comercios por idComerio.categorias_comercios=id.comercios

favoritos_clientes: comercios que los clientes han marcado como favoritos. Está relacionada con la tabla clientes por idCliente.favoritos_clientes=id.clientes y con la tabla comercios por idComercio.favoritos_clientes=id.comercios. Tiene un campo llamado *verMensajes* donde el cliente puede elegir si ver o no los mensajes que va poniendo el comercio.

dia_aperturas: datos que añade el comercio sobre sus días festivos de apertura al público. Está relacionada con la tabla comercios por idComercio.dia_aperturas=id.comercios**.** Tiene un campo llamado *visible* donde el comercio puede elegir cuando estará visible este dato para que lo vea el cliente.

horarios: datos que añade el comercio sobre sus diferentes horarios. Está relacionada con la tabla comercios por idComercio.horarios=id.comercios**.** Tiene un campo llamado *visible* donde el comercio puede elegir cuando estará visible este dato para que lo vea el cliente.

producto_comercios: productos que ofrece el comercio, se puede poner los productos más vendidos. Está relacionada con la tabla comercios por idComercio.producto_comercios =id.comercios**.**

mensajes: mensajes que el comercio puede añadir para que los vea los clientes que tienen su comercio como favorito. Está relacionada con la tabla comercios por idComercio.mensajes=id.comercios**.** Tiene un campo llamado *visible* donde el comercio puede elegir cuando estará visible este dato para que lo vea el cliente.

5.2. Diagrama de Casos de Uso

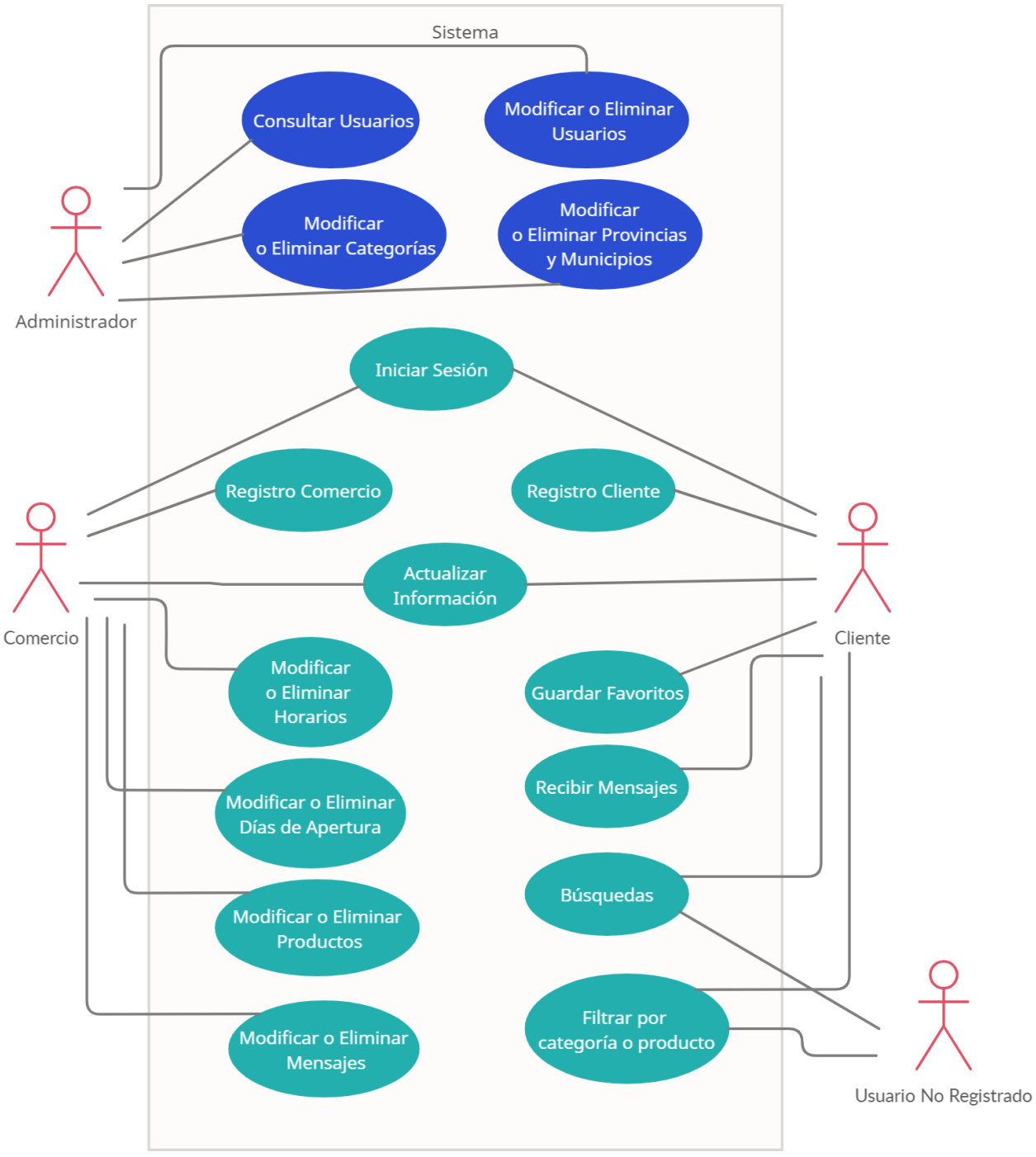


Figura 10 – Diagrama Casos de Uso

6. Análisis de mercado, usabilidad.

6.1. Definición teórica del público objetivo

Nuestra aplicación tiene que ofrecer al pequeño comercio un sistema sencillo y ágil de utilizar, para tener actualizado sus horarios, días festivos abiertos y mostrar al cliente un listado de los productos que ofrece, y así poder aumentar la fidelidad con sus clientes.

Que el cliente pueda consultar de una manera rápida y sencilla, que horarios tiene su establecimiento de confianza, que días tiene de apertura, que producto puede comprar en un comercio cercano.

Tenemos que presentar un contenido, que sea el atractivo principal para las primeras y repetidas visitas al sitio. Debe tener su propia presencia de marca, proporcionar contenido y facilitar las tareas.

Si nuestra aplicación no tiene presencia de marca, tendremos aumentar esfuerzos para construir su marca, desde ejecutar campañas publicitarias hasta expresar rasgos de la marca a través de las características y el diseño del sitio web.

En el desarrollo debemos considerar si nuestra aplicación ira orientada a una solución móvil o una solución orientada a PC o ambas. Debemos valorar que el uso de dispositivos móviles y tabletas se han convertido en una parte cada vez más esencial de la estrategia de la empresa y la planificación de productos, y un conjunto de habilidades extremadamente importante que los diseñadores debemos desarrollar.

Hay que tener en cuenta la cantidad de personas que tienen acceso a dispositivos móviles, así como la cantidad de tiempo y la cantidad de actividades realizadas en plataformas móviles en lugar de ordenadores de escritorio. La base de usuarios puede ser mucho mayor con un dispositivo móvil que con un ordenador de escritorio.

Para nuestra aplicación nos vendría muy bien los servicios del móvil basados en la ubicación determinadas por GPS.

Podemos hacer uso de las técnicas de diseño receptivo, utilizando cuadrículas flexibles que expanden o contraen contenido según la resolución de la pantalla, imágenes flexibles que disminuyen de tamaño en pantallas más pequeñas o aumentan a un tamaño máximo establecido en pantallas más grandes.

El uso de estas técnicas puede ayudarnos a coger un sitio común y adaptar el diseño a múltiples dispositivos, brindando una flexibilidad que cubre una variedad de resoluciones desde teléfonos inteligentes hasta tabletas y navegadores de escritorio.

Los sitios web optimizados para móviles aportan mayor grado de alcance y en la mayoría de los casos, una solución más rentable.

Para definir nuestro público objetivo tenemos que pensar en nuestra aplicación basada en la comunicación con el pequeño comercio. Debemos tener en cuenta datos demográficos como la edad, sexo, familia (soltero, casado, hijos), nivel de ingresos, nivel de educación, el nivel de familiaridad con las tecnologías relevantes, hábitos de compra y de consumo.

Para el análisis del público objetivo la meta es extraer, analizar y cruzar todos los datos posibles y disponibles, para obtener diferentes perfiles.

Nuestro público objetivo o target, serían todas las personas a las que vendemos o queremos vender nuestros productos y servicios. Podemos tener en cuenta los siguientes puntos:

- Rango de edad: puede ser un rango de edad bastante amplio, desde jóvenes que querían comprar videojuegos hasta mayores que quieran comprar pan o algún artículo de primera necesidad.
- Ingresos promedio: también puede ser amplio, desde un nivel de ingresos bajos a un nivel de ingresos altos.
- Actores del mercado: puede ser cualquier cliente que viva en el mismo barrio, en la misma ciudad o incluso en otra ciudad que estando de vacaciones, quiera utilizar nuestra aplicación para buscar cualquier producto en algún pequeño comercio. También los pequeños comercios que quieran darse a conocer y estar más en contacto con sus clientes, mostrando información sobre sus horarios, días de apertura, productos más vendidos.

Con toda esta información podemos elaborar el perfil de dos “User Personas”, que serán representaciones de un grupo de usuarios con comportamientos, objetivos y motivaciones similares.

INÉS – 32 AÑOS - PELUQUERA	
	**EXPERIENCIA PERSONAL** Inés es peluquera, tiene 32 años, tiene un trabajo a tiempo completo. No está casada y no tiene hijos. En su tiempo libre le gusta realizar actividades de riesgo como paracaidismo, rafting, escalada, entre otras. Le gusta hacer viajes en grupos, para conocer gente nueva y adquirir nuevas experiencias. Inés es una persona activa, aventurera y extrovertida.
OBJETIVOS	**TECNOLOGÍA**
Inés tiene pocas horas libres, pero le gusta mucho salir de compras, sobre todo por el pequeño comercio de su barrio. El día que tiene libre, los domingos, le gustaría también salir de compras, pero es difícil saber qué comercio hay abierto y el horario que tienen esos días.	Inés es una mujer muy tecnológica, tiene un móvil smartphones, ordenador portátil, Tablet. Le gusta mucho las redes sociales, tiene cuenta en Facebook, LinkedIn, Instagram, donde suele mostrar fotos de sus viajes.

MANUEL – 52 AÑOS - PANADERO	
	**EXPERIENCIA PERSONAL** Manuel es panadero y tiene de 52 años, tiene su propia panadería. Está casada con Carmen de 48 años. Tienen 2 hijos, Laura y Jorge. Ocasionalmente suele viajar por trabajo. Aunque lo que más le gusta es viajar con su familia. Le encanta su trabajo, es su pasión y sus hijos suele ayudarle en el negocio. Manuel es una persona responsable, organizado y sociable.
OBJETIVOS	**TECNOLOGÍA**
A Manuel le gustaría estar más en contacto con sus clientes, poderles avisar de cambios en los horarios y los días festivos que tiene abierto la panadería. Le gustaría darse a conocer a personas que hacen turismo en su ciudad y que se acerquen a conocer sus productos.	Manuel tiene un móvil smartphones, ordenador portátil y Tablet. Utiliza mucho las redes sociales como modo de enseñar sus productos.

Imágenes obtenidas de la web [6]

6.2. Arquitectura de la información

La arquitectura de la información nos proporciona contexto para el contenido y nos dice qué podemos hacer mientras estamos aquí, nos dice dónde estamos, nos ayuda a movernos a otras páginas estrechamente relacionadas, nos ayuda a movernos por el sitio jerárquicamente y contextualmente, nos permite manipular el contenido para una mejor navegación y nos dice dónde podemos acudir para obtener servicios básicos, como iniciar sesión en nuestra cuenta u obtener ayuda.

Para poder hacer un sitemap correcto tenemos que tener en cuenta los siguientes puntos:

- En los sitios web, la amplitud y profundidad es una función tanto de la estructura de la información como de la navegación, las estructuras más amplias funcionan mejor que las más profundas.
- La intención y la función de la navegación deben quedar claras de inmediato.
- Al navegar por su sitio, los visitantes deben estar informados sobre lo que está sucediendo. El sistema de navegación que diseñas les da pistas sobre cómo navegar por el sitio. El texto y las etiquetas son la principal forma en que las personas sabrán qué opción es cuál o cuál es el título de la página actual.
- Crear vínculos de navegación, pestañas e íconos que sean fáciles de ver y de hacer clic.
- Crea un mejor sentido de orientación y facilidad de uso de la navegación.
- Defina su grupo objetivo e identifique las necesidades de información clave de cada grupo.

La arquitectura de información se estructura de algunas formas básicas, como sistemas de organización, sistemas de navegación, sistemas de búsqueda y sistemas de etiquetado. Las categorías agrupan páginas y aplicaciones en todo el sitio. Las etiquetas representan sistemáticamente el contenido del sitio. Se pueden utilizar sistemas de navegación y un sistema de búsqueda para moverse por el sitio.

Tenemos que añadir a nuestra aplicación un sistema de búsqueda que permita que los usuarios busquen el contenido y presentar automáticamente un conjunto personalizado de resultados que coinciden con sus consultas. Podemos mejorar el rendimiento de una consulta, incluyendo correctores ortográficos, derivación, búsqueda de conceptos y extracción de sinónimos de un tesauro.

Dentro de las ayudas a la navegación podemos incluir las rutas de navegación que ayudan a los usuarios a comprender dónde se encuentran. La ruta de navegación con migas de pan es un elemento de navegación secundario que se coloca en la parte superior de una página web y que sirve como apoyo al tradicional menú de navegación.

A partir de nuestro público objetivo, la arquitectura de la información y los casos de uso, podemos crear nuestro Sitemap y Diagramas de flujo.

Nuestro Sitemap sería así:

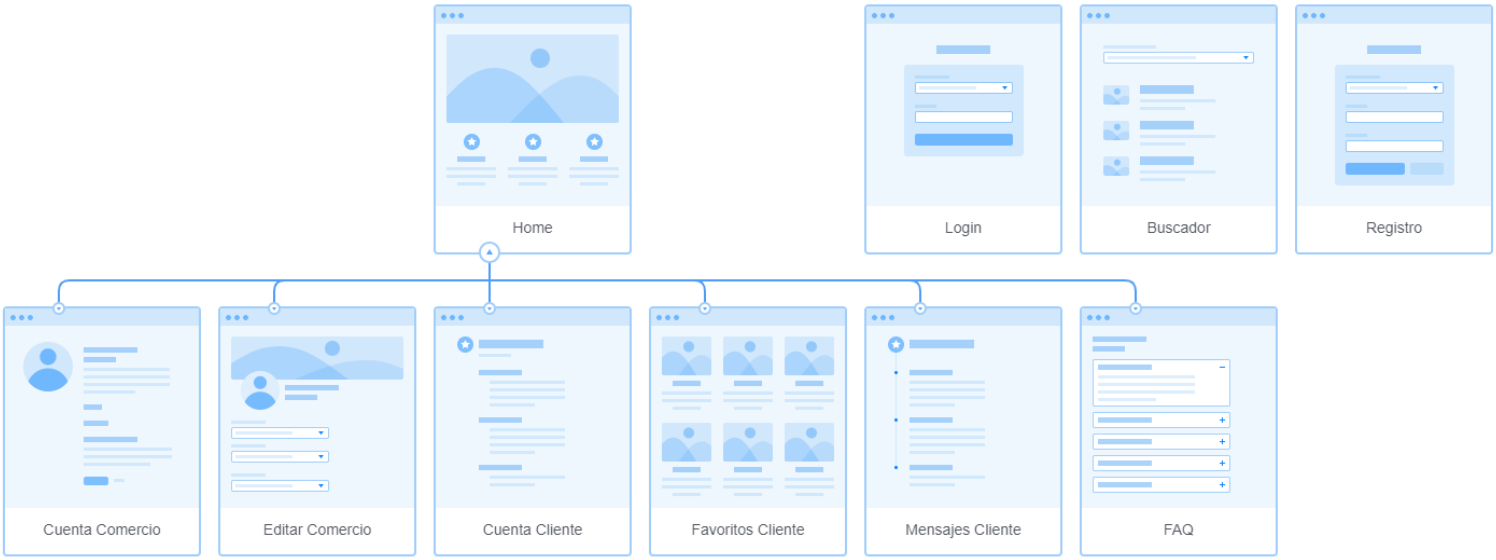


Figura 11 – Sitemap de nuestra aplicación

Nuestro diagrama de flujo de la aplicación para el **cliente** sería así:

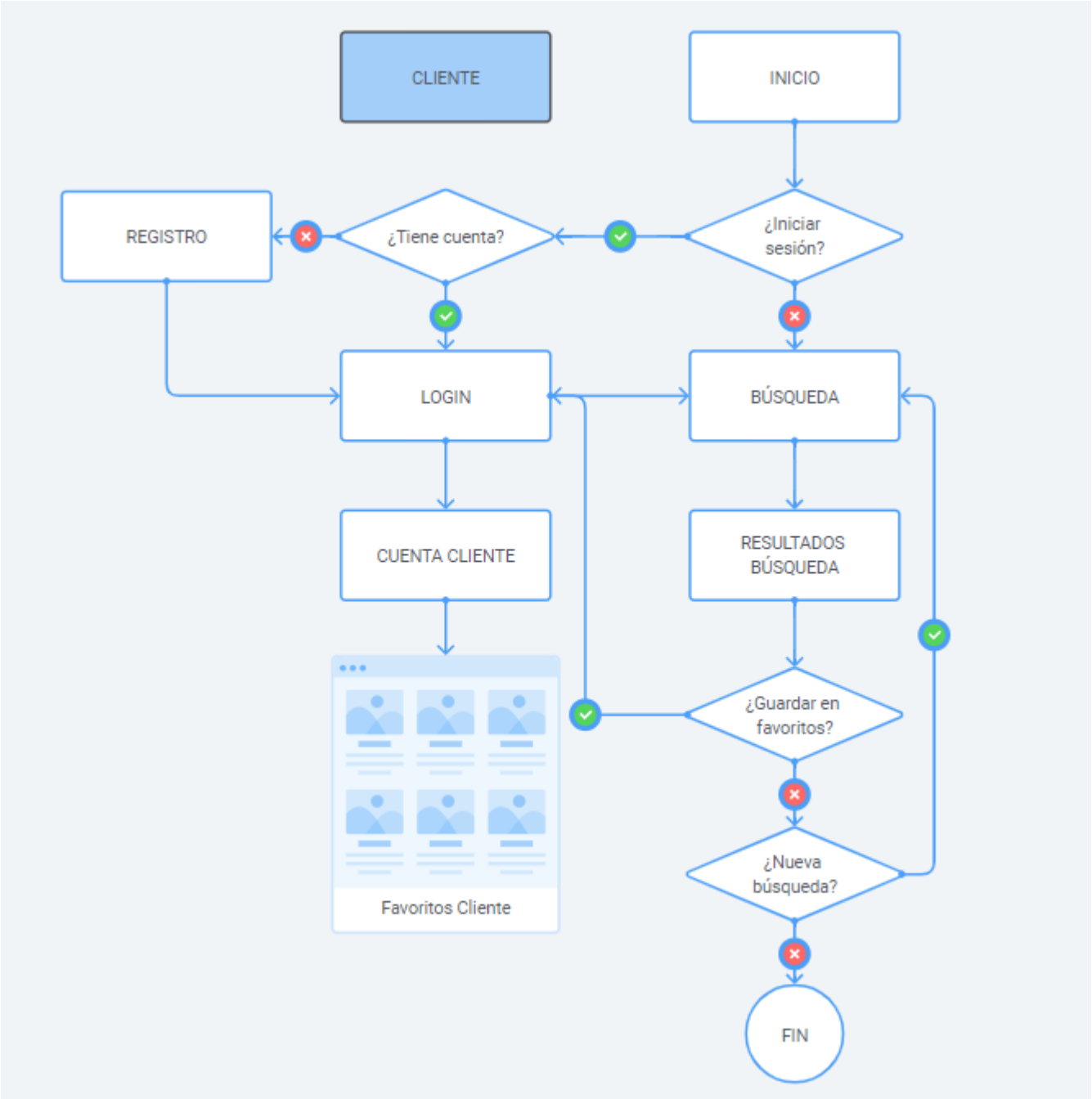


Figura 12 – Diagrama de flujo Cliente

Nuestro diagrama de flujo de la aplicación para el **comercio** sería así:

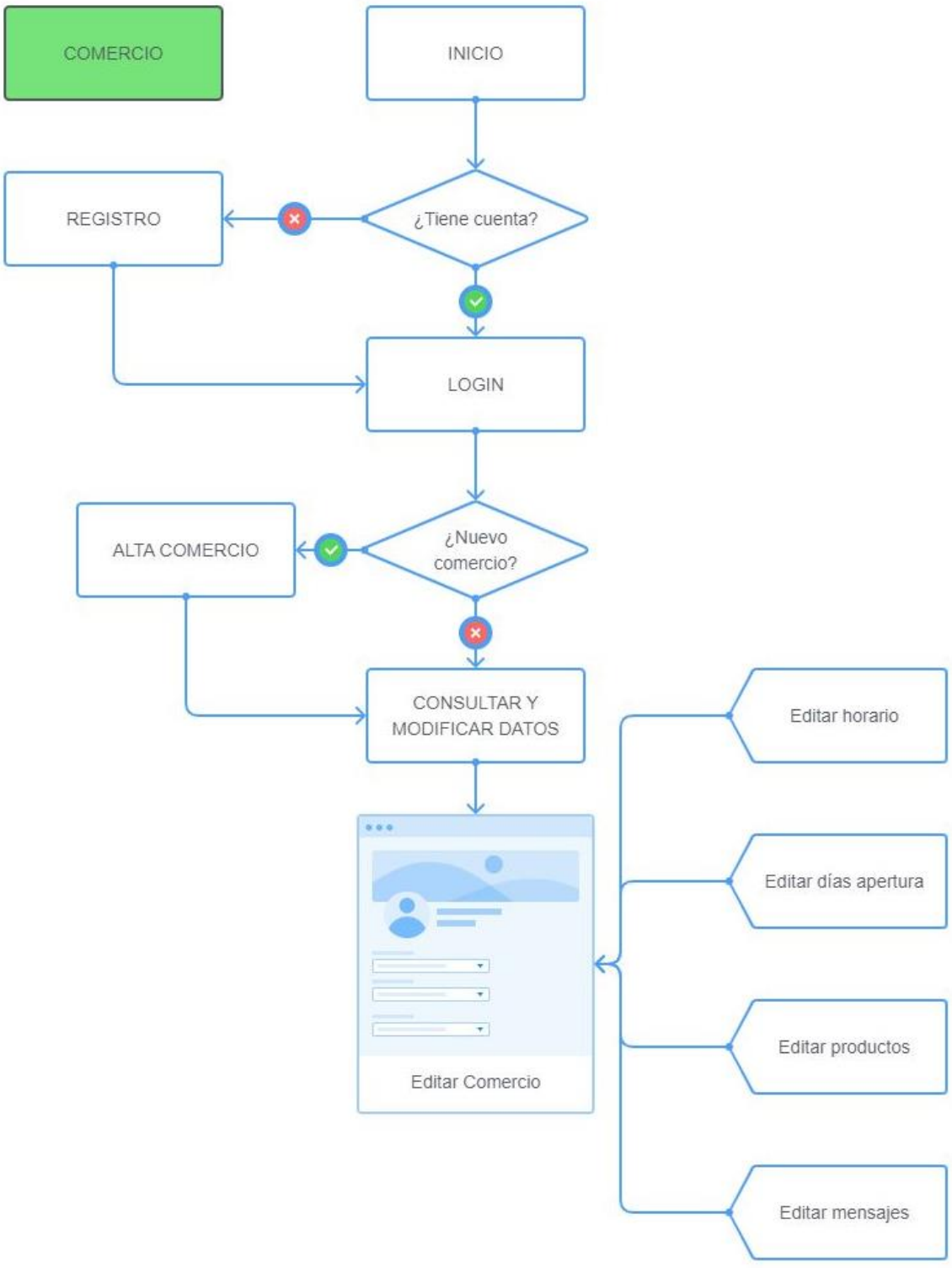


Figura 13 – Diagrama de flujo Comercio

6.3. Usabilidad

Con la usabilidad podemos analizar la experiencia y satisfacción del usuario sobre nuestra interfaz web. Nos permite gestionar y adaptar las herramientas web con el objetivo facilitar a las personas su utilización. Tenemos que estructurar la información de forma simple, racional y lógica.

Para evaluar el diseño de la interfaz de usuario se utilizan las heurísticas creadas por Jakob Nielsen:

1- Visibilidad del estado del sistema: Esto significa que debe asegurarse de que la interfaz siempre le diga al usuario lo que está sucediendo, es decir, todas las acciones necesitan comentarios instantáneos para guiar su copia. Comunicar el estado actual permite a los usuarios sentirse en control del sistema, tomar las acciones adecuadas para alcanzar su objetivo.

2- Relación entre la interfaz del sistema y el mundo real: No utilice palabras del sistema que no tengan sentido para el usuario. Toda la comunicación del sistema debe contextualizarse para el usuario y ser coherente con el llamado modelo mental del usuario. Los sistemas deben hablar el idioma de los usuarios con palabras, frases y conceptos familiares en lugar de términos orientados al sistema.

3- Libertad y control del usuario: Facilitar salidas de emergencia al usuario, permitiendo deshacer o rehacer acciones en el sistema para volver al punto anterior, para cuando el usuario se pierde o en situaciones inesperadas. Los usuarios suelen cometer errores o cambiar de opinión.

4- Consistencia: Habla el mismo idioma todo el tiempo y nunca identifiques la misma acción con diferentes iconos o palabras. Trate las cosas similares de la misma manera, lo que facilita su identificación para el usuario.

5- Prevención de errores: En la traducción gratuita de las propias palabras de Nielsen, incluso mejor que un buen mensaje de error es un diseño cuidadoso que puede prevenir tales errores. Por ejemplo, acciones definitivas como eliminaciones o solicitudes pueden ir acompañadas de una casilla de verificación o un mensaje de confirmación.

6- Reconocimiento en lugar de recuerdo: Evite activar la memoria del usuario todo el tiempo, haciendo que cada acción se revise mentalmente antes de ejecutarla. Permita que la interfaz proporcione ayuda sensible al contexto e información que pueda guiar las acciones del usuario, es decir, los diálogos del sistema con el usuario. Mostrar a los usuarios cosas que pueden reconocer mejora la usabilidad sobre la necesidad de recuperar elementos desde cero porque el contexto adicional ayuda a los usuarios a recuperar información de la memoria.

7- Flexibilidad y eficiencia de uso: El sistema debe ser fácil para los usuarios no profesionales, pero lo suficientemente flexible como para ser ágil para los usuarios avanzados.

8- Estética y diseño minimalista: Evite los textos y las charlas de diseño que sean más de lo que el usuario necesita saber. Los diálogos del sistema deben ser simples, directos y naturales, solo presentes en los momentos en que se necesitan.

9- Ayude a los usuarios a reconocer, diagnosticar y corregir errores: Los mensajes de error del sistema deben tener una redacción clara y simple que, en lugar de intimidar al usuario con el error, indique un resultado constructivo o una posible solución.

10- Ayuda y documentación: Un buen diseño debería evitar la necesidad de ayuda para usar el sistema tanto como sea posible. Aun así, se debe utilizar un buen conjunto de documentación y ayuda para guiar al usuario en caso de duda. Debe ser visible, de fácil acceso y debe ofrecerse una herramienta de búsqueda en la ayuda.

Información consultada en web [5], [6] y asignatura Diseño de Interfaces interactivas.

6.4. Viabilidad

Se entiende por viabilidad, determinar la probabilidad de que un determinado proyecto se pueda realizar. Nuestro proyecto es algo sencillo y fácil de implementar tecnológicamente, ya que no se necesitan muchos medios técnicos y ni de muchos recursos humanos.

Desconozco si existe actualmente en el mercado algún tipo de aplicación parecida al proyecto desarrollado. Nuestra mayor competencia sería Google, donde se puede hacer una búsqueda de algún comercio en concreto para obtener el horario o días de apertura del comercio buscado.

Algunos de estos resultados que nos devuelve Google, puede ser que no estén actualizados correctamente, por esto, un factor importante de nuestra aplicación es que nuestros datos están constantemente actualizados por el comercio.

Otra de las fortalezas de nuestra aplicación será la directa comunicación entre comercio y cliente. El comercio podrá enviar mensajes sobre cambios de horarios y nuevos días de apertura en festivos.

La viabilidad economía de nuestra aplicación tampoco será muy elevada, no necesitamos de muchos recursos económicos para llevarla a cabo.

Tenemos que ver que oportunidades tenemos actualmente en el mercado, si tenemos una competencia débil, hará falta la necesidad de nuestro producto. Con una buena calidad del producto final podremos entrar más fácilmente en el mercado.

7. Prototipos

7.1 Prototipos Lo-Fi

En los prototipos Lo-Fi o de baja fidelidad mostramos diseños simples en blanco y negro, donde se implementan aspectos generales sin entrar en detalles. Tenemos que mostrar los elementos más importantes de cada interfaz de usuario.

A continuación, se muestran algunas de las pantallas más importantes para la aplicación del usuario tanto para dispositivos móviles como de escritorio.

Se han realizado con el programa Excalidraw. (Se adjuntan los archivos en la carpeta PEC_FINAL_HuertaRubio_Rebeca\Documentacion\Prototipos\Baja Calidad Lo-Fi)

Versión móvil y escritorio

Pantalla Home: esta pantalla será la principal (Inicio), donde aparecerán varias imágenes de publicidad para comercios. En el Header tenemos el botón para hacer una Búsqueda, el botón de Login para poder iniciar sesión y el botón Crear cuenta, para darnos de alta.

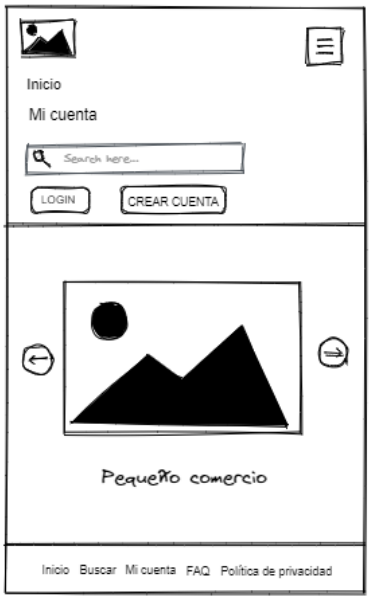


Figura 14-Prototipo Lo-Fi móvil – Home

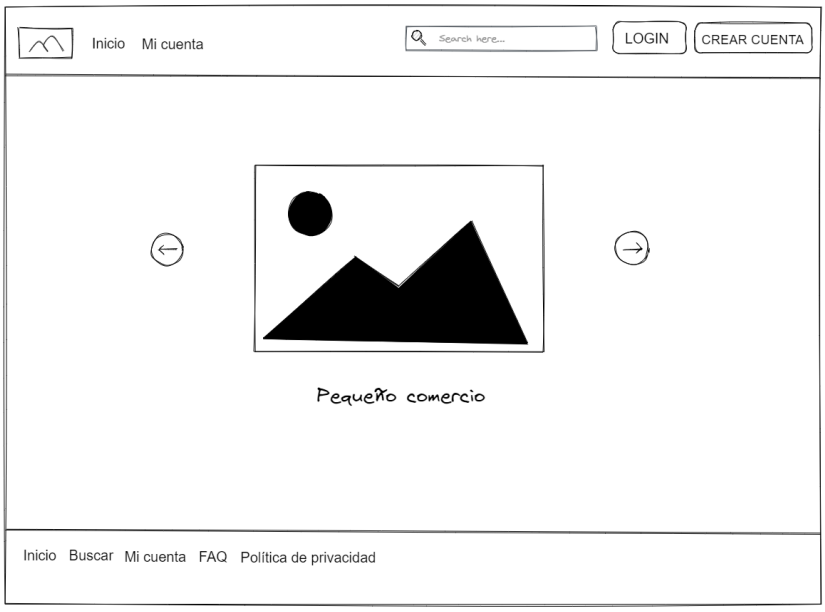


Figura 15-Prototipo Lo-Fi escritorio– Home

Pantalla Login: tenemos el formulario para meter nuestros datos de usuario, metiendo el Email y Password. El Login vale tanto para usuarios tipo cliente, como usuarios tipo comercio.

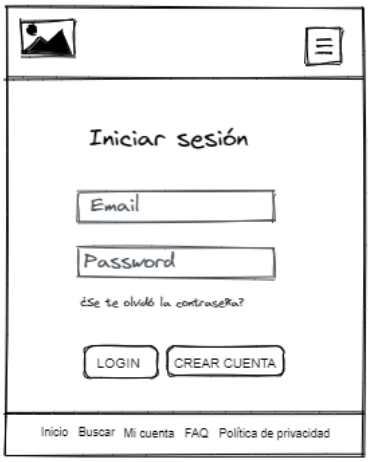


Figura 16-Prototipo Lo-Fi móvil – Login

Figura 17-Prototipo Lo-Fi escritorio – Login

Pantalla Registro: crear una nueva cuenta, tenemos que elegir si queremos darnos de alta como cliente o como comercio, según la opción elegida, tendremos que rellenar unos datos u otros.

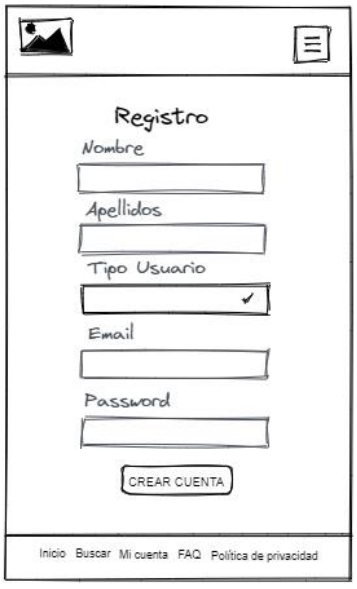


Figura 18-Prototipo Lo-Fi móvil – Registro

Figura 19-Prototipo Lo-Fi escritorio – Registro

Pantalla Buscador: buscar productos en los comercios de proximidad, podemos buscar por el nombre del comercio, por municipio, provincia, por algún producto en concreto, etc.

Figura 20- Prototipo Lo-Fi móvil – Buscador

Figura 21-Prototipo Lo-Fi escritorio – Buscador

Pantalla Cuenta Comercio: el usuario tipo comercio, podrá crear, editar o eliminar la información sobre su negocio, como su dirección, municipio, provincia, categoría del comercio, sus horarios, sus días de apertura en festivos, sus productos más vendidos, y mensajes para mostrar a sus clientes que tengan como favorito su comercio.

Figura 22-Prototipo Lo-Fi móvil – Cuenta Comercio

Figura 23-Prototipo Lo-Fi escritorio – Cuenta Comercio

Pantalla Detalle Comercio: es toda la información sobre un comercio, donde está localizado, sus horarios, días de apertura, etc. Esta información estará disponible tanto para usuario tipo clientes, como usuarios no registrados. Si el usuario ya es cliente, tendrá la opción de marcar este comercio como favorito.

Figura 24-Prototipo Lo-Fi móvil – Detalle Comercio

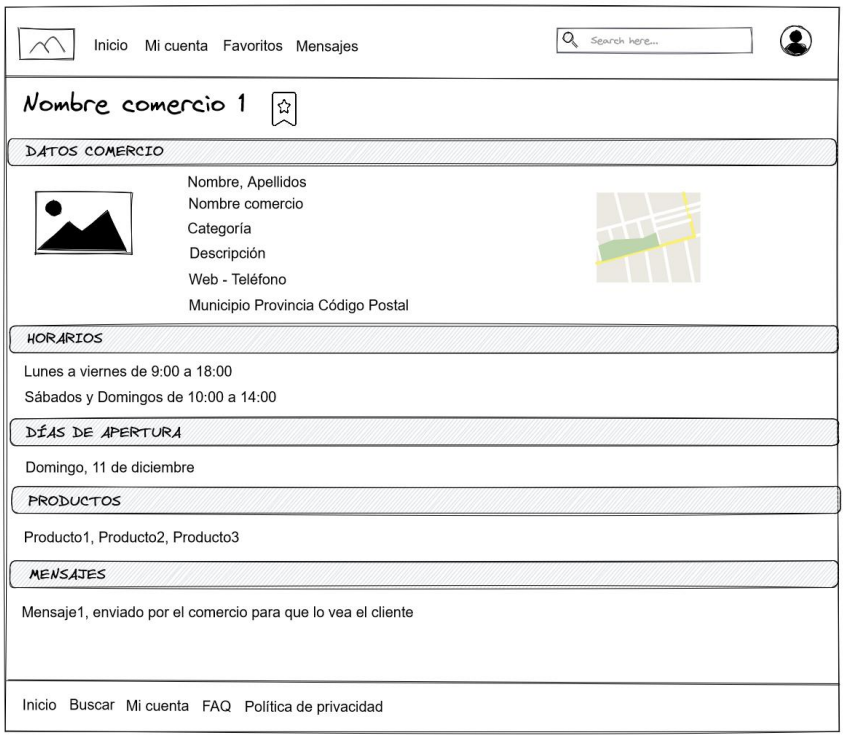


Figura 25-Prototipo Lo-Fi escritorio – Detalle Comercio

Pantalla Cuenta Cliente: información sobre el usuario de tipo cliente, aquí puede ver sus datos de acceso, sus datos personales, sus comercios favoritos.

Figura 26-Prototipo Lo-Fi móvil – Cuenta Cliente

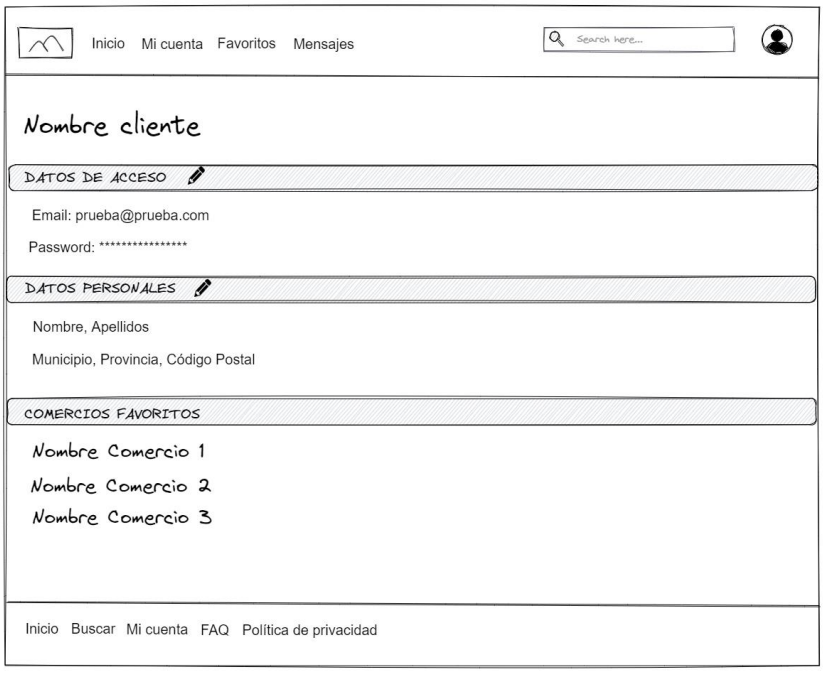


Figura 27-Prototipo Lo-Fi escritorio – Cuenta Cliente

Pantalla Favoritos: listado de los comercios favoritos del cliente, desde aquí tendrá la opción de ver en detalle el comercio o quitar el comercio como favorito.

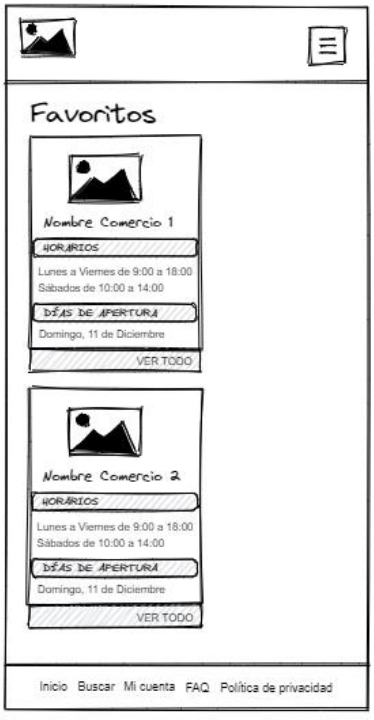


Figura 28-Prototipo Lo-Fi móvil – Favoritos

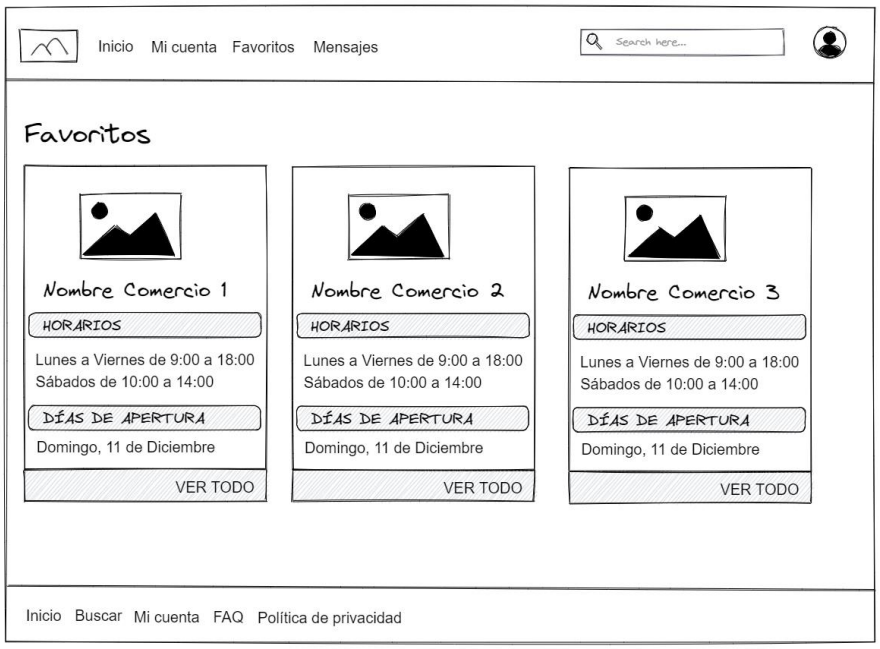


Figura 29-Prototipo Lo-Fi escritorio – Favoritos

Pantalla Mensajes: listado de los mensajes de los comercios que el cliente tiene como favoritos. El cliente tendrá la opción de elegir si quiere que esos mensajes estén visibles o no.

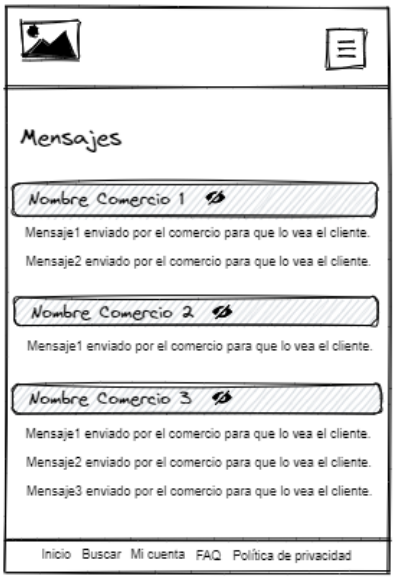


Figura 30-Prototipo Lo-Fi móvil – Mensajes

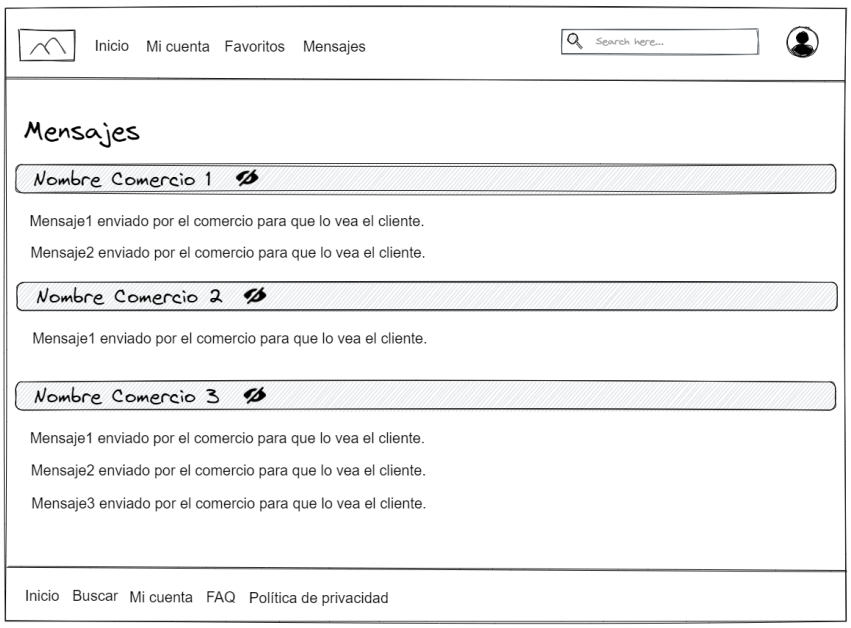


Figura 31-Prototipo Lo-Fi escritorio – Mensajes

7.2. Prototipos Hi-Fi

En los prototipos Hi-Fi o de alta fidelidad se representan aspectos más precisos, será una representación similar al producto final en cuanto a su funcionalidad e interacción.

A continuación, se muestran algunas de las pantallas más importantes para la aplicación del usuario tanto para dispositivos móviles como de escritorio.

Para la representación de estos prototipos se utiliza HTML, CSS y el framework de Bootstrap v.5.2.2. (Se adjuntan los archivos en la carpeta PEC_FINAL_HuertaRubio_Rebeca\Documentacion\Prototipos\Alta Calidad Hi-Fi)

Versión móvil y escritorio

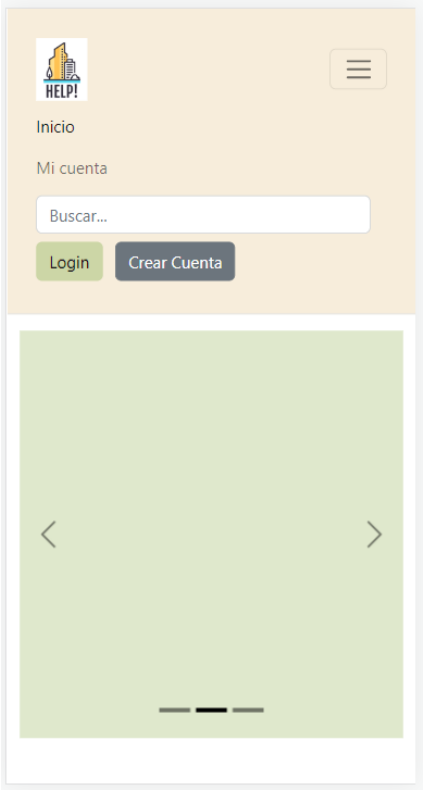


Figura 32-Prototipo Hi-Fi móvil – Home

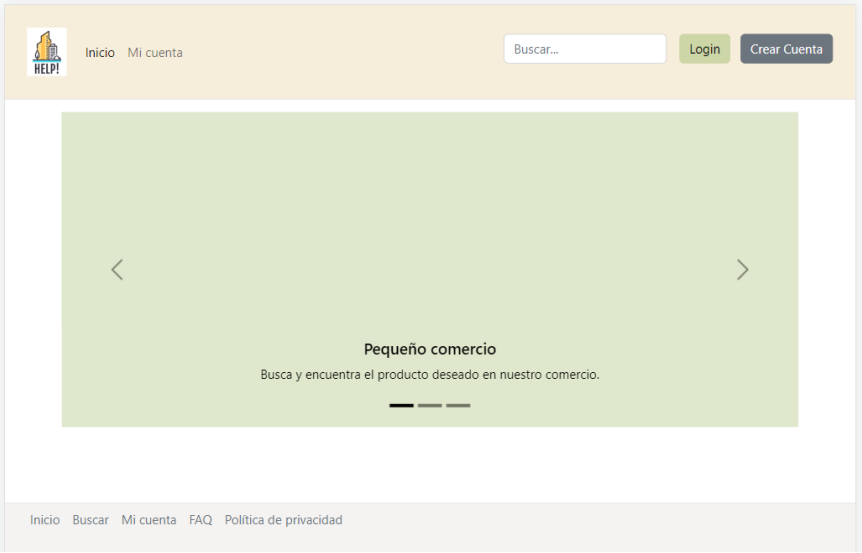


Figura 33-Prototipo Hi-Fi escritorio– Home

Figura 34-Prototipo Hi-Fi móvil – Login

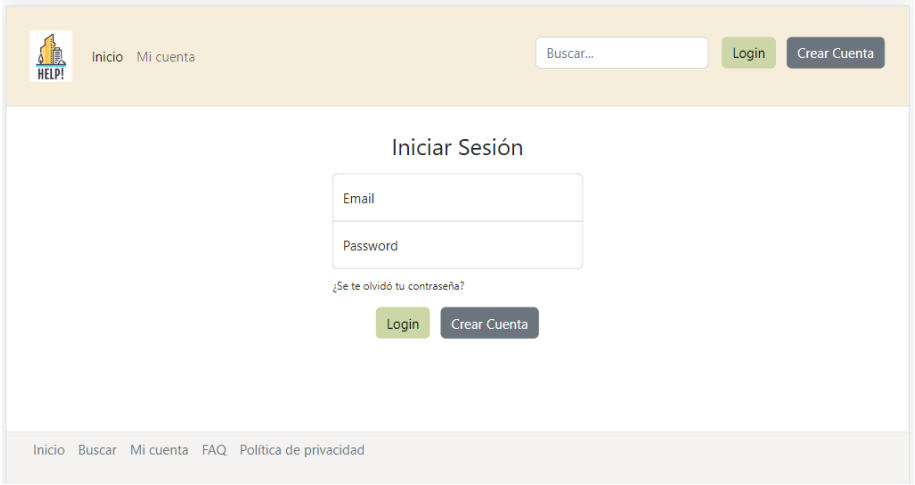


Figura 35-Prototipo Hi-Fi escritorio – Login

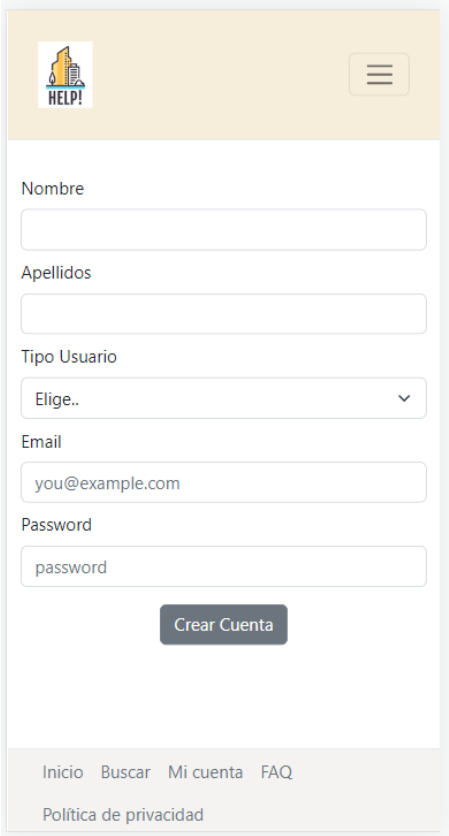


Figura 36-Prototipo Hi-Fi móvil – Registro

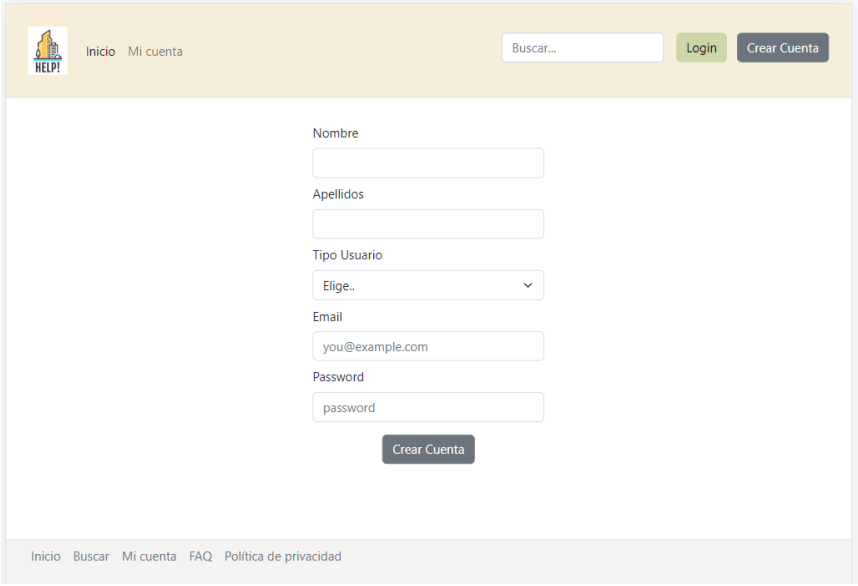


Figura 37-Prototipo Hi-Fi escritorio – Registro

Figura 38-Prototipo Hi-Fi móvil – Buscador

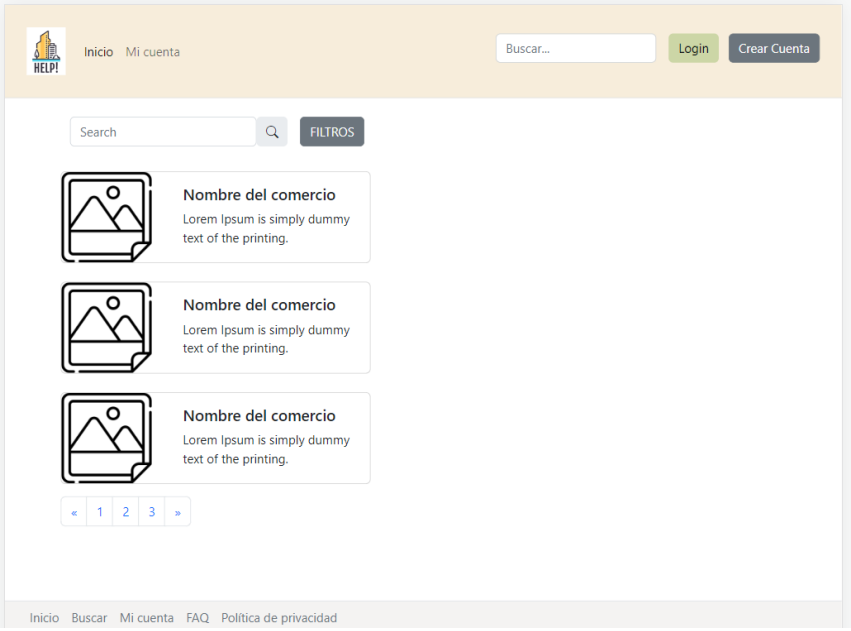


Figura 39-Prototipo Hi-Fi escritorio – Buscador

Figura 40-Prototipo Hi-Fi móvil–Cuenta Comercio

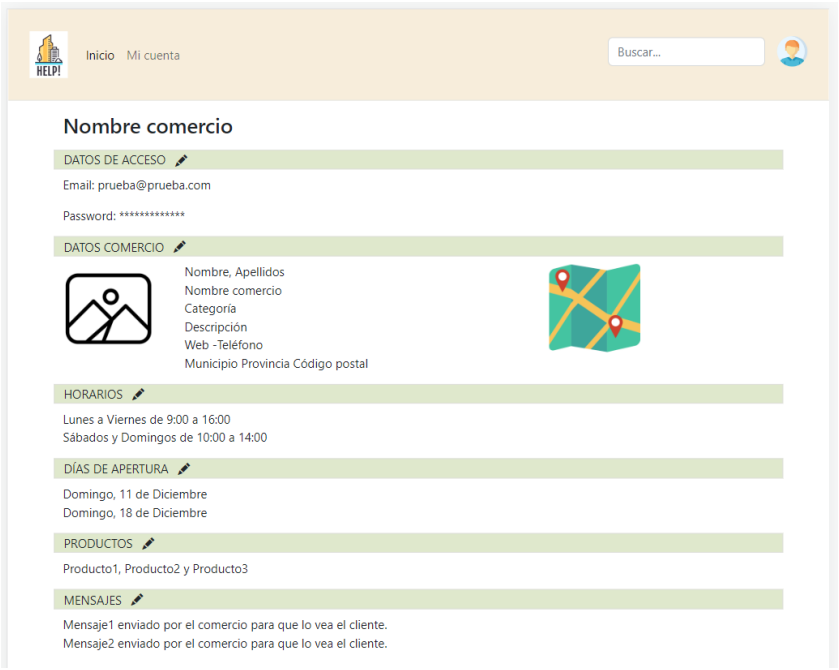


Figura 41-Prototipo Hi-Fi escritorio – Cuenta Comercio

Figura 42-Prototipo Hi-Fi móvil – Detalle Comercio

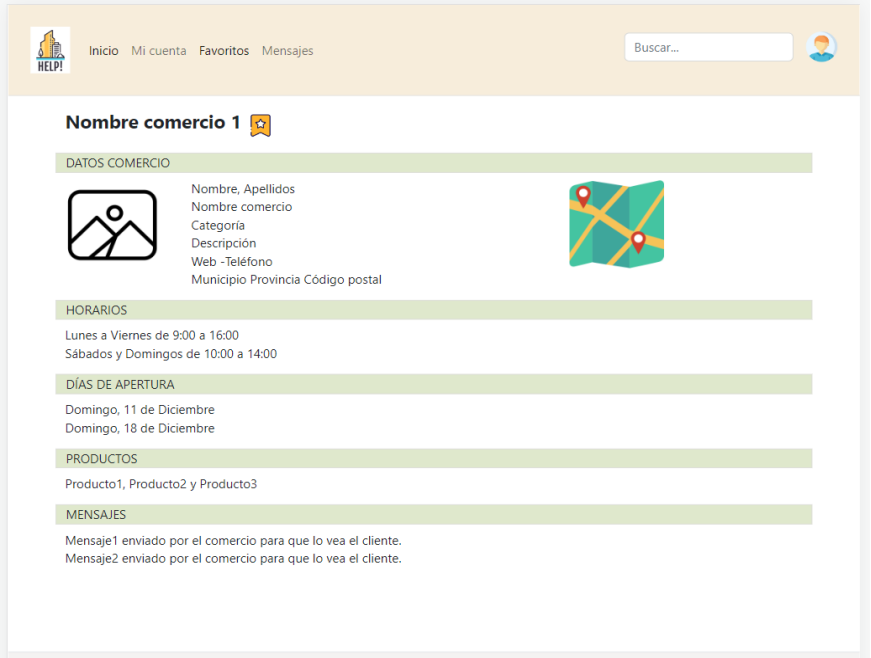


Figura 43-Prototipo Hi-Fi escritorio – Detalle Comercio

Figura 44-Prototipo Hi-Fi móvil – Cuenta Cliente

Figura 45-Prototipo Hi-Fi escritorio – Cuenta Cliente

Figura 46-Prototipo Hi-Fi móvil – Favoritos

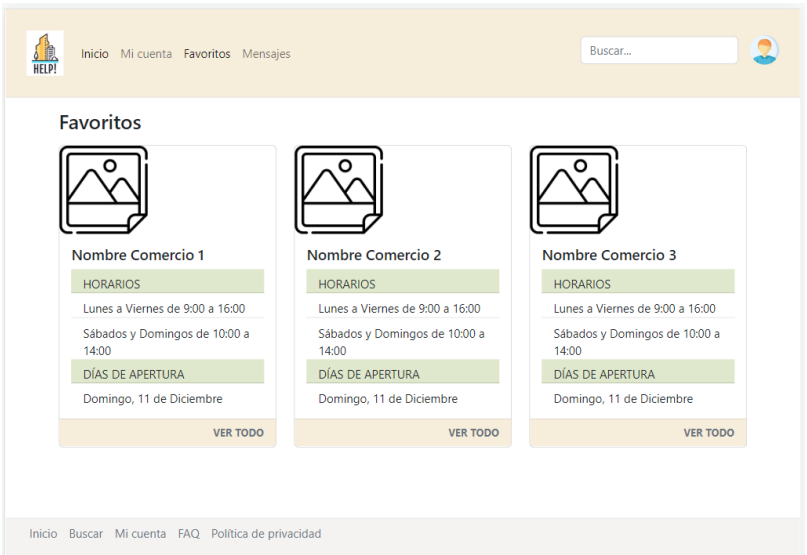


Figura 47-Prototipo Hi-Fi escritorio – Favoritos

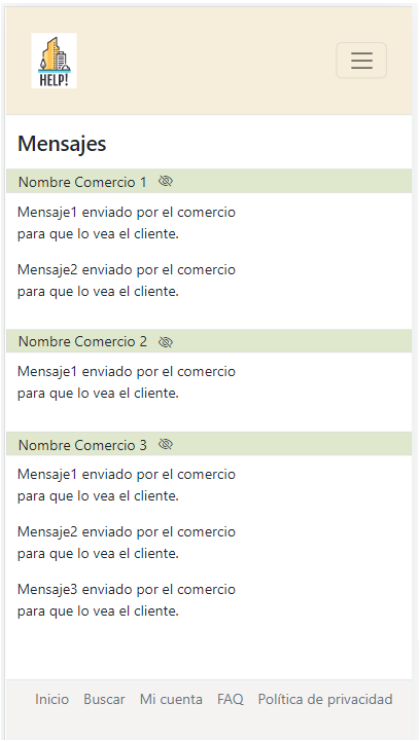


Figura 48-Prototipo Hi-Fi móvil – Mensajes

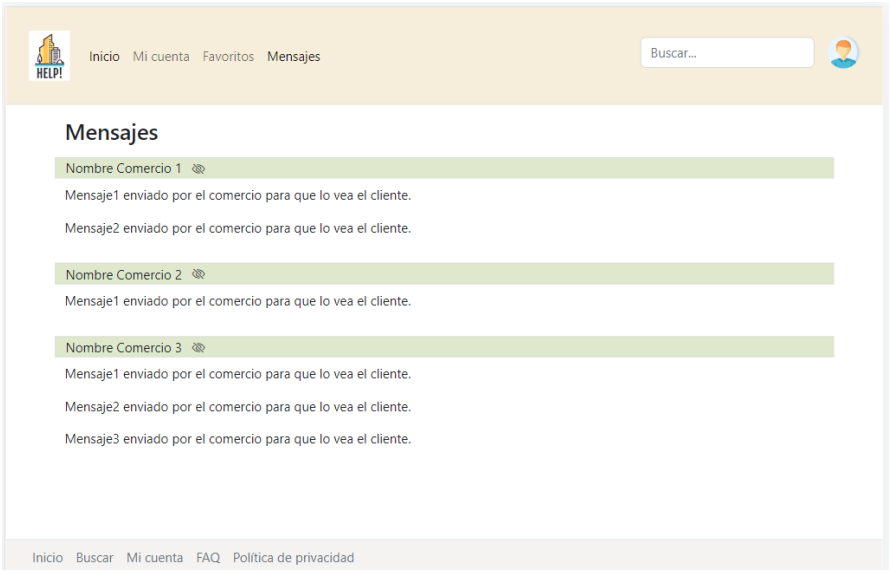


Figura 49- Prototipo Hi-Fi escritorio – Mensajes

8. Descripción de perfiles

A continuación, se explican las distintas funcionalidades de cada usuario según su perfil.

8.1. Perfil Administrador

El usuario con perfil de administrador será el encargado de la gestión de la aplicación, podrá realizar las siguientes acciones:

- Dar de alta, modificar o eliminar cualquier tipo de usuario.
- Dar de alta, modificar o eliminar las categorías de los comercios.
- Dar de alta, modificar o eliminar los municipios o provincias.
- Consultar un listado de todos los comercios o clientes.
- Podrá activar o desactivar el acceso a la aplicación de cualquier usuario, mediante el campo "activo" de la tabla Users, si hay algún problema con algún usuario.

8.2. Perfil Comercio

El usuario con perfil de comercio podrá realizar las siguientes opciones:

- Darse de alta en la aplicación, tendrá que introducir los siguientes datos:
    - Tipo Usuario (comercio)
    - Nombre
    - Apellidos
    - Email
    - Password
    - Nombre Comercio
    - Descripción
    - Imagen *
    - Dirección
    - Municipio
    - Provincia
    - Código Postal
    - Web *
    - Teléfono *

    * campos opcionales

- Dar de alta, modificar o eliminar la categoría al que pertenece su negocio, el mismo comercio podrá tener relación con una o varias categorías.

- Dar de alta, modificar o eliminar los días de apertura de su comercio, tendrá que elegir el día, elegir el estado (“Abierto, “Cerrado”), y elegir si es visible o no para el cliente.

- Dar de alta, modificar o eliminar los horarios del comercio, tendrá que completar el campo descripción donde pondrá el horario y elegir si es visible o no para el cliente.

- Dar de alta, modificar o eliminar los productos que ofrece su comercio. Tendrá la opción de poner sus productos más destacados.

- Dar de alta, modificar o eliminar los mensajes que verán los clientes que tengan su comercio como favorito, y elegir si es visible o no para el cliente.

8.3. Perfil Cliente

El usuario con perfil de cliente podrá realizar las siguientes opciones:

- Darse de alta en la aplicación, como mínimo tendrá que introducir los siguientes datos:

    - Tipo Usuario (cliente)
    - Nombre
    - Apellidos
    - Email
    - Password
    - Municipio
    - Provincia
    - Código Postal

- Realizar una búsqueda en la aplicación, podrá buscar por el nombre del producto, el nombre del comercio o filtrar por categorías del comercio, municipio, provincia o código postal.

- Dar de alta, modificar o eliminar sus comercios favoritos, tendrá la opción de elegir si quiere ver o no los mensajes que el comercio vaya poniendo.

8.4. Perfil Usuario no registrado

El usuario no registrado podrá realizar las siguientes opciones:

- Realizar una búsqueda en la aplicación, podrá buscar poniendo el nombre del producto, el nombre del comercio o filtrar por categorías del comercio, municipio, provincia o código postal.

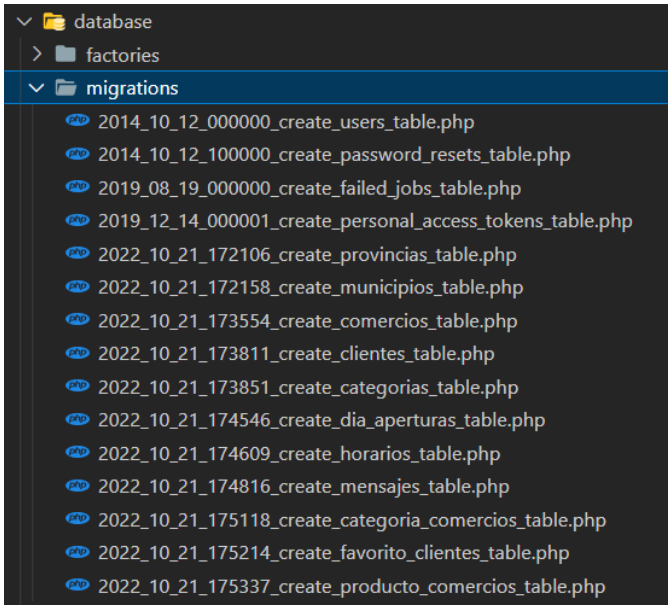
9. Proceso de desarrollo

9.1. Desarrollo Backend

Creación de las tablas

Nuestra base de datos va a tener 12 tablas, utilizamos la tabla Users que viene por defecto en Laravel, para los datos de acceso de nuestros usuarios.

Mediante **php artisan make:migration create** vamos creando las tablas. Se irán creando en la carpeta Database->migrations.



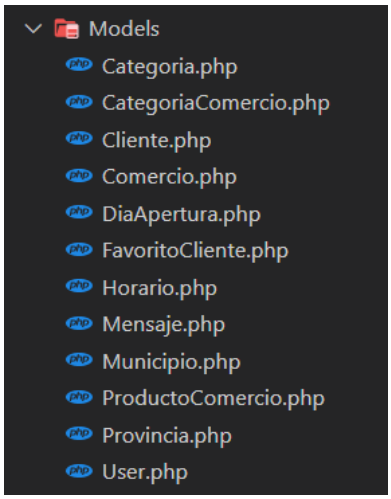
Configuramos las columnas necesarias para cada tabla, por ejemplo, para la tabla comercio, añadimos los siguientes campos, juntos con sus respectivas clave primaria y claves foráneas.

```php
public function up()
{
    Schema::create('comercios', function (Blueprint $table) {
        $table->id();
        //definimos la clave foranea con la tabla users
        $table->foreign('id')->references('id')->on('users');
        $table->string('nombreComercio', 100);      You, hace 2 semanas • segu
        $table->string('descripcion', 300)->nullable();
        $table->string('imagen', 300)->nullable();
        $table->string('direccion', 300);
        //el idMunicipio de la tabla Municipios.
        $table->unsignedBigInteger('idMunicipio');
        //definimos la clave foranea con la tabla Municipios
        $table->foreign('idMunicipio')->references('id')->on('municipios');
        //el idProvincia de la tabla Provincias.
        $table->unsignedBigInteger('idProvincia');
        //definimos la clave foranea con la tabla Provincias
        $table->foreign('idProvincia')->references('id')->on('provincias');
        $table->string('codigopostal', 5);
        $table->string('web', 300)->nullable();
        $table->string('telefono', 100)->nullable();
        $table->timestamps();
    });
}
```

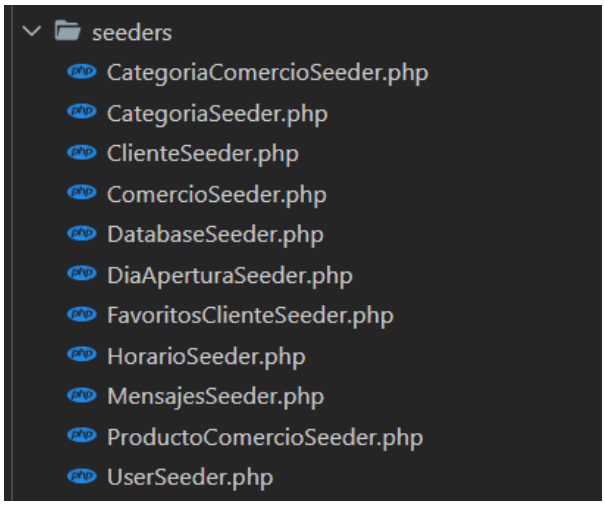
Cuando tenemos todos los campos añadidos, utilizamos **php artisan migrate** para actualizar todos estos cambios en nuestra base de datos.

Creación de los modelos

A continuación creamos nuestros modelos, mediante **php artisan make:model Comercio**, se añadirán en la carpeta app ->Models.



También se van a crear algunos seeders, para meter datos en nuestra base de datos, mediante **php artisan make:seeder CategoriaSeeder**, estos archivos se crean en database->seeders



Para todas estas tablas, user, categoría, cliente, comercio, categoría_comercios, favorito_clientes, dia_aperturas, horarios, producto_comercios y mensajes se añadirán datos desde los seeders.

Por ejemplo, para la tabla Users, se añadirá datos de prueba para usuarios tipo comercio y usuarios tipo cliente, de la siguiente forma:

```php
DB::table('users')->insert([
    'nombre' => 'Carmen',
    'apellidos' => 'Lopez Lopez',
    'email' => 'carmenll@prueba.com',
    'password' => bcrypt('C4781348'),
    'tipoUsuario' => 'comercio',
    'activo' => 'SI'
]);
```

```php
DB::table('users')->insert([
    'nombre' => 'Inés',
    'apellidos' => 'Lopez Rubio',
    'email' => 'ineslr@prueba.com',
    'password' => bcrypt('I7462596'),
    'tipoUsuario' => 'cliente',
    'activo' => 'SI'
]);
```

Con estos dos usuarios iremos haciendo las pruebas desde la parte del Front.

Para actualizar estos datos en nuestra base de datos, completamos el archivo DatabaseSeeder.php con los siguientes datos y ejecutamos **php artisan db:seed**

```php
public function run()
{
    $this->call(UserSeeder::class);
    $this->call(CategoriaSeeder::class);
    $this->call(ComercioSeeder::class);
    $this->call(ClienteSeeder::class);
    $this->call(CategoriaComercioSeeder::class);
    $this->call(FavoritosclienteSeeder::class);
    $this->call(DiaAperturaSeeder::class);
    $this->call(HorarioSeeder::class);
    $this->call(ProductoComercioSeeder::class);
    $this->call(MensajesSeeder::class);
}
```

Como los seeders para las tablas Provincias y Municipios, es bastante amplio, he creado dos archivos .txt (insert_provincias.txt y insert_municipios.txt) para introducir los datos directamente en el base de datos mediante un insert. (se adjuntan estos documentos en la carpeta PEC_FINAL_HuertaRubio_Rebeca\Proyecto)

Instalación de la librería JWT:

Json Web Token (JWT) nos permite crear un método de autenticación en servicios API, para que nuestra conexión entre el cliente y nuestro Backend sea segura. El cliente envía el usuario y contraseña, y la API le retorna un token que enviará en todas las peticiones, para que ésta compruebe que tiene acceso a las acciones que se quieran realizar.

Nos vamos a la página oficial de JWT *[7]* y seguimos las instrucciones de instalación:

Primero instalamos la librería con el siguiente comando (con php 8)

composer require tymon/jwt-auth --ignore-platform-reqs

Agregamos el proveedor de servicios al **providers** matriz en el **config/app.php**

```
'providers' => [
    ...
    Tymon\JWTAuth\Providers\LaravelServiceProvider::class,
]
```

Ejecutamos el siguiente comando para publicar el archivo de configuración del paquete:

php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

Generar una clave, **php artisan jwt:secret** y esto actualizará el archivo .env con el JWT_SECRET.

Añadimos al modelo de User:

```
use Tymon\JWTAuth\Contracts\JWTSubject;
class User extends Authenticatable implements JWTSubject;
```

Y los métodos:

```
public function getJWTIdentifier()
{
    return $this->getKey();
}
public function getJWTCustomClaims()
{
   return [];
}
```

En el archivo **config/auth.php** modificamos los siguiente:

```
'defaults' => [
    'guard' => 'api',
    'passwords' => 'users',
],
...
'guards' => [
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

Creamos el controlador AuthControler, para comprobar el login y el logout. Añadimos la siguiente función para proteger las rutas:

```
public function __construct()
{
    $this->middleware('auth:api', ['except' => ['login']]);
}
```

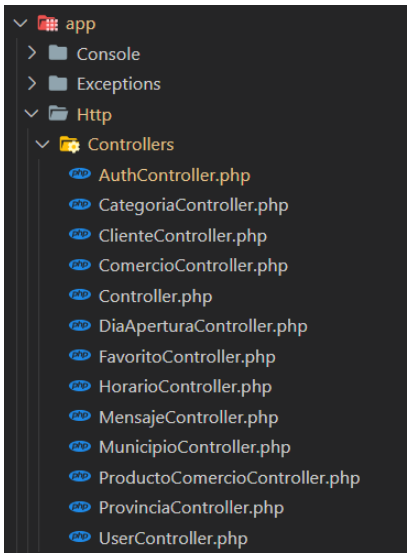
```php
class AuthController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth:api', ['except' => ['login']]);
    }

    public function login()
    {
        $credentials = request(['email', 'password']);

        if (!$token = auth()->attempt($credentials)) {
            return response()->json(['error' => 'Usuario o Password incorrecto.'], 401);
        } else {
            $user = Auth::user();
        }
        return $this->respondWithToken($token, $user);
    }

    protected function respondWithToken($token, $user)
    {
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'user_id' => $user->id,
            'tipo_usuario' => $user->tipoUsuario,
        ]);
    }
```

En la función responWithToken, en el response pasamos el token, y también voy a pasar el user_id y tipo_usuario del user, para que luego se almacene en la parte del localStorage del FrontEnd. Ya que luego necesitamos saber qué tipo de usuario ha realizado el Login, si es comercio o cliente, para mostrar unas vistas u otras.

Creación de la APIRest

Primero creamos nuestros controladores mediante **php artisan make:controller UserController --resource**

Se crean dentro de la carpeta app->Http->Controllers



Añadimos las funciones necesarias para cada controlador, por ejemplo, para el controlador Comercio, necesitamos las siguientes funciones:

- index, retornar todos los valores
- show, retornar un valor por id de la tabla Comercio
- store, para crear un nuevo usuario
- update, para actualizar un usuario por id
- destroy, para eliminar un usuario por id
- buscador, para buscar comercios.

A la hora de crear un nuevo usuario desde la web, tenemos que tener en cuenta, si el usuario es de tipo cliente o comercio. Si es de tipo comercio, utilizaremos la función store del controlador **ComercioController**. Si es de tipo cliente, utilizaremos la función store del controlador **ClienteController**.

En la función store, primero se dará de alta los datos del usuario en la tabla Users, y luego obtenemos el id generado para pasarlo a la tabla Comercios, ya que tendrá el mismo id. Lo mismo para crear un usuario cliente.

```php
$user = User::create([
    'nombre' => $request['nombre'],
    'apellidos' => $request['apellidos'],
    'email' => $request['email'],
    'password' => $request['password'],
    'tipoUsuario' => $request['tipoUsuario'],
    'activo' => $request['activo'],
]);

if ($user) {
    //cogemos el id de user que se ha creado para insertarlo en la tabla de cliente
    $comercio = Comercio::create([
        'id' => $user->id,
        'nombreComercio' => $request['nombreComercio'],
        'descripcion' => $request['descripcion'],
        'direccion' => $request['direccion'],
        'idMunicipio' => $request['idMunicipio'],      You, hace 2 semanas • segundo
        'idProvincia' => $request['idProvincia'],
        'codigopostal' => $request['codigopostal'],
        'web' => $request['web'],
        'telefono' => $request['telefono'],
    ]);
```

Antes de crear el usuario tenemos de codificar el Password, para que se almacene en la base de datos encriptada y proteger esa contraseña.

```php
//codificamos el password antes de crear el registro.
$request['password'] = bcrypt($request->input('password'));
```

Tanto para modificar o eliminar un usuario tendremos en cuenta las dos tablas Users y Comercios o Users y Clientes.

A continuación, definimos las rutas de acceso de nuestra aplicación, para esto accedemos al archivo api.php de la carpeta routes:

```php
//rutas auth
Route::post('auth/login', [AuthController::class, 'login']);
Route::post('auth/logout', [AuthController::class, 'logout']);

//rutas user
Route::get('users', [UserController::class, 'index']);
Route::post('user', [UserController::class, 'store']);
Route::get('user/{id}', [UserController::class, 'show']);

//rutas comercio
Route::get('comercio', [ComercioController::class, 'index']);
Route::get('comercio/{id}', [ComercioController::class, 'show']);
Route::post('comercio', [ComercioController::class, 'store']);
Route::put('comercio/{id}', [ComercioController::class, 'update']);
//rutas buscador
Route::get('comercio/buscador/{dato}', [ComercioController::class, 'buscador']);

//rutas horarios
Route::get('horario/{id}', [HorarioController::class, 'show']);
Route::get('horarios/{idComercio}', [HorarioController::class, 'showHorarios']);
Route::get('horarios/visible/{idComercio}', [HorarioController::class, 'showHorariosVisibles']);
Route::post('horario', [HorarioController::class, 'store']);
Route::put('horario/{id}', [HorarioController::class, 'update']);
Route::delete('horario/{id}', [HorarioController::class, 'destroy']);
```

```php
//rutas dia de apertura
Route::get('dia/{id}', [DiaAperturaController::class, 'show']);
Route::get('dias/{idComercio}', [DiaAperturaController::class, 'showDias']);
Route::get('dias/visible/{idComercio}', [DiaAperturaController::class, 'showDiasVisibles']);
Route::post('dia', [DiaAperturaController::class, 'store']);
Route::put('dia/{id}', [DiaAperturaController::class, 'update']);
Route::delete('dia/{id}', [DiaAperturaController::class, 'destroy']);

//rutas productos
Route::get('producto/{id}', [ProductoComercioController::class, 'show']);
Route::get('productos/{idComercio}', [ProductoComercioController::class, 'showProductos']);
Route::post('producto', [ProductoComercioController::class, 'store']);
Route::put('producto/{id}', [ProductoComercioController::class, 'update']);
Route::delete('producto/{id}', [ProductoComercioController::class, 'destroy']);

//rutas mensajes
Route::get('mensaje/{id}', [MensajeController::class, 'show']);
Route::get('mensajes/{idComercio}', [MensajeController::class, 'showMensajes']);
Route::get('mensajes/visible/{idComercio}', [MensajeController::class, 'showMensajesVisibles']);
Route::post('mensaje', [MensajeController::class, 'store']);
Route::put('mensaje/{id}', [MensajeController::class, 'update']);
Route::delete('mensaje/{id}', [MensajeController::class, 'destroy']);

//rutas cliente
Route::get('clientes', [ClienteController::class, 'index']);
Route::get('cliente/{id}', [ClienteController::class, 'show']);
Route::post('cliente', [ClienteController::class, 'store']);
Route::put('cliente/{id}', [ClienteController::class, 'update']);
```

```php
//rutas favoritos
Route::get('favoritos/{idCliente}', [FavoritoController::class, 'favoritos']);
Route::get('favoritos/vermensajes/{idCliente}', [FavoritoController::class, 'mensajesFavoritos']);
Route::get('favorito/comprobarFav/{idCliente}/{idComercio}', [FavoritoController::class, 'comprobarFavorito']);
Route::post('favorito', [FavoritoController::class, 'storeFavorito']);
Route::put('favorito/vermensaje/{idFavorito}', [FavoritoController::class, 'updateMensajesfavoritos']);
Route::delete('favorito/{idCliente}/{idComercio}', [FavoritoController::class, 'destroyfavorito']);

//rutas categorias
Route::get('categoria', [CategoriaController::class, 'index']);
Route::get('categoria/{id}', [CategoriaController::class, 'show']);

//rutas provincias
Route::get('provincia', [ProvinciaController::class, 'index']);
Route::get('provincia/{id}', [ProvinciaController::class, 'show']);

//rutas municipio
Route::get('municipio/{id}', [MunicipioController::class, 'show']);
Route::get('municipios/{id}', [MunicipioController::class, 'showByIdProvincia']);
```

En todos los controladores se ha añadido en la parte del constructor, la opción de middleware que proporciona un mecanismo para filtrar solicitudes HTTP que ingresan en nuestra aplicación. El middleware de Laravel verifica si el usuario esta autenticado o no.

```php
public function __construct()
{
    $this->middleware('auth:api', ['except' => ['show', 'store', 'buscador']]);
}
```

Si el usuario no está autenticado, el middleware lo redirigirá a la pantalla de inicio de sesión de la aplicación. Sin embargo, si el usuario está autenticado, el middleware permitirá que la solicitud continúe en la aplicación.

En el caso del controlador Comercio, hemos puesto en el “except” las funciones show, store y buscador, ya que para estos casos no es necesario tener un usuario autenticado.

9.2. Desarrollo Frontend

Como vamos a utilizar los estilos de Bootstrap lo añadimos al proyecto, mediante:

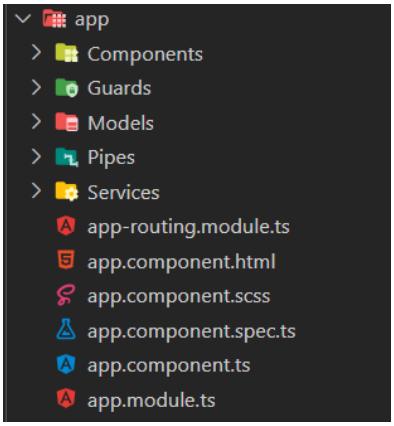
npm install bootstrap jquery @popperjs/core

Luego, en el archivo “angular.json” hay que poner las siguientes instrucciones en los objetos “styles” y “scripts”:

```
"styles": [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "node_modules/bootstrap-icons/font/bootstrap-icons.css",
  "src/styles.scss"
],
"scripts": [
  "node_modules/bootstrap/dist/js/bootstrap.min.js",
  "node_modules/jquery/dist/jquery.min.js",
  "node_modules/@popperjs/core/dist/umd/popper.min.js"
]
```

También añadimos el archivo src/styles.scss para crear nuestros propios estilos o modificar estilos de Bootstrap.

Lo primero de todo creamos las carpetas necesarias para ir creando nuestros archivos posteriormente. Las carpetas son Components, Guards, Models, Pipes y Services.



Dentro de la carpeta Guards, creamos el archivo **auth.guard.ts**, donde crearemos la clase AuthGuard que implementa de CanActivate.

Los Guards en Angular son middlewares que se ejecutan antes de cargar una ruta y determinan si se puede cargar dicha ruta o no. Los hay de cuatro tipos diferentes, nosotros utilizaremos CanActivate para hacer la comprobación antes de cargar los componentes de la ruta.

Estos componentes se ejecutan antes de determinadas acciones y si retorna true la ruta seguiría su carga normal, en caso negativo, el Guard retornaría false y la ruta no se cargaría.

Con **canActicate** comprobamos que exista una sesión abierta mediante el access_token y devolvemos true, o en caso contrario, redireccionamos a la ruta de login.

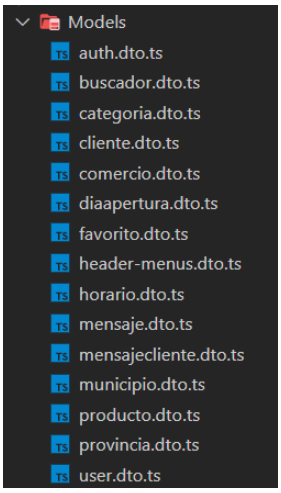
```
canActivate(
  route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot
):
  | Observable<boolean | UrlTree>
  | Promise<boolean | UrlTree>
  | boolean
  | UrlTree {
  const access_token = this.localStorageService.get('access_token');
  if (access_token) {
    // logged in so return true
    return true;
  }

  this.router.navigate(['/login']);

  return false;
}
```

Para utilizar Guard, tenemos que ir al archivo **app-routing.module.ts**, donde definimos las rutas de la aplicación. En algunas rutas, como home, login o register, no hace falta aplicarlo, ya que no es necesario estar logeado para tener acceso a estas rutas. Otras rutas como cliente, comercio, mensajes, si es necesario añadir **canActivate: [AuthGuard]**, para proteger esas rutas:

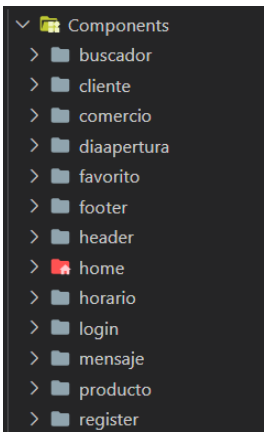
```
{
  path: 'home',
  component: HomeComponent,
},
{
  path: 'login',
  component: LoginComponent,
},
{
  path: 'register',
  component: RegisterComponent,
},
```

```
{
  path: 'mensaje/:id',
  component: MensajeFormComponent,
  canActivate: [AuthGuard],
},
{
  path: 'cliente-cuenta',
  component: ClienteCuentaComponent,
  canActivate: [AuthGuard],
},
{
  path: 'cliente/:idCliente',
  component: ClienteFormComponent,
  canActivate: [AuthGuard],
},
```

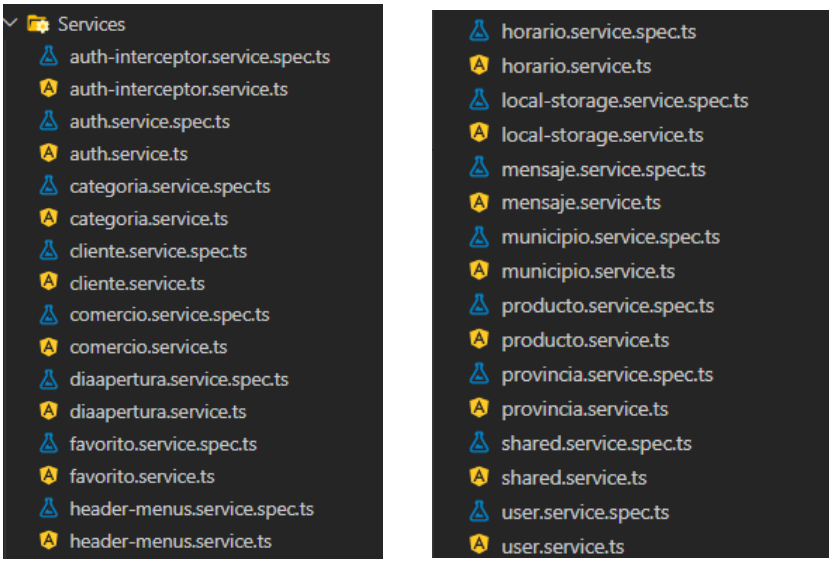
A continuación, creamos los modelos necesarios que utilizaremos en la aplicación:



Los controladores los he separado en carpetas de la siguiente manera:



Y los servicios creados han sido los siguientes:



El servicio de **auth.service.ts**, se utiliza para hacer la llamada a la API **"auth/login"** para realizar el Login, que nos devolverá un JSON con los datos del access_token.

```
export class AuthService {
  private urlApi: string;
  private controller: string;

  constructor(private http: HttpClient, private sharedService: SharedService) {
    this.controller = 'auth/login';       You, hace 2 semanas • segundo commit …
    this.urlApi = 'http://127.0.0.1:8000/api/' + this.controller;
  }

  login(auth: AuthDTO): Observable<AuthToken> {
    return this.http
      .post<AuthToken>(this.urlApi, auth)
      .pipe(catchError(this.sharedService.handleError));
  }
}
```

Desde el controlador **login.component.ts**, haremos la llamada a este servicio, si la respuesta es ok, almacenamos las siguientes variables en el localStorage.

```
this.localStorageService.set('user_id', this.loginUser.user_id);
this.localStorageService.set('tipo_usuario',this.loginUser.tipo_usuario);
this.localStorageService.set('access_token',this.loginUser.access_token);
```

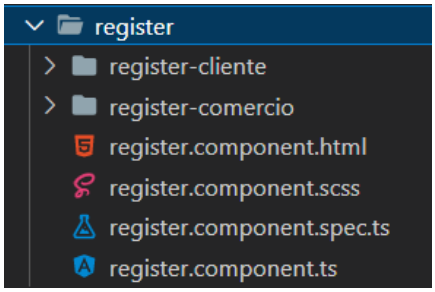
El 'tipo_usuario', lo almaceno para saber si el usuario que ha iniciado sesión es de tipo cliente o comercio, ya que luego lo iremos utilizando en los diferentes controladores, para realizar ciertas acciones, como cargar diferentes rutas o diferentes menús.

Dentro del servicio **header-menus.service.ts**, declaramos estas tres variables para ir utilizándolas en los diferentes controladores:

```
export class HeaderMenusService {
  headerManagement: BehaviorSubject<HeaderMenus> =
    new BehaviorSubject<HeaderMenus>({
      showNoAuthSection: true,
      showAuthSectionCliente: false,
      showAuthSectionComercio: false,
    });
}
```

- showNoAuthSection: para mostrar los menús en el header para cuando ningún usuario ha iniciado sesión.
- showAuthSectionCliente: para mostrar los menús en el header, para los usuarios tipo cliente.
- showAuthSectionComercio: para mostrar los menús en el header, para los usuarios tipo comercio.

Dentro de la carpeta del controlador **register**, vamos a tener dos tipos de registro uno para el tipo de usuario cliente (register-cliente) y otro para el tipo usuario comercio (register-comercio). En principio quería utilizar el mismo formulario para los dos registros, pero al pedir datos diferentes, tenía problemas con las validaciones, al final decidí separar los dos formularios. Para esto pasamos previamente por una vista *register.component.html*, para elegir primero el tipo de usuario, y según la elección, nos llevará a un formulario u otro.



Para la parte vista **Comercio** vamos a utilizar los siguientes controladores:

- Comercio:
  - comercio-cuenta: será la parte que verá el usuario tipo comercio, donde implementamos la parte de los datos de acceso y datos personales, del menú Mi cuenta.
  - comercio-form: para que el usuario pueda modificar sus datos de acceso y personales referentes a su comercio.

- Horario:
  - horario-list: listado de horarios que el comercio ha dado de alta y donde podemos crear nuevos, modificar o eliminar esos horarios.
  - horario-form: formulario para dar de alta o modificar un horario.

- Diaapertura:
  - diaapertura-list: listados de los días de apertura de festivos y domingos que el comercio ha dado de alta y donde podemos crear nuevos, modificar o eliminar esos días.
  - diaapertura-form: formulario para dar de alta o modificar un día de apertura.

- Producto:
  - producto-list: listados de los productos que el comercio ha dado de alta y donde podemos crear nuevos, modificar o eliminar esos productos.
  - producto-form: formulario para dar de alta o modificar un producto.

- Mensaje:
  - mensaje-list: listados de los mensajes que el comercio ha dado de alta y donde podemos crear nuevos, modificar o eliminar esos mensajes.
  - mensaje-form: formulario para dar de alta o modificar un mensaje.

Para la parte vista **Cliente** vamos a utilizar los siguientes controladores:

- Cliente:
  - cliente-cuenta: será la parte que verá el usuario tipo cliente, donde implementamos la parte de los datos de acceso y datos personales del menú Mi cuenta.
  - cliente-form: para que el usuario pueda modificar sus datos de acceso y personales referentes a su usuario.

- Comercio:
  - comercio-view: será la vista del comercio, que verán los clientes o usuarios sin cuenta. Desde esta vista, tendremos que cargar todos los datos del comercio, horarios, días de apertura, productos y mensajes.

- Favorito: listado de los comercios favoritos que tiene ese cliente, desde esta pantalla el cliente podrá ir al detalle del comercio o quitar como favorito ese comercio.

- Mensaje:
  - mensaje-cliente: listado de los mensajes de los comercios marcados como favoritos, desde aquí el cliente tendrá la opción de dejar de ver los mensajes del comercio que desee.

10. Proyección a futuro y posibles mejoras.

Esta aplicación se ha desarrollado en un ámbito académico como parte del TFM, por lo tanto, no tiene ninguna salida comercial.

Aunque he podido desarrollar la mayoría de las funcionalidades deseadas, por falta de tiempo, algunas funcionalidades se han quedado sin desarrollar.

A continuación, se detallan algunas mejoras que se podría hacer en futuras versiones para mejorar la aplicación:

Mejoras funcionales:

- Versión para la parte del Administrador, para poder dar de alta, modificar o eliminar las diferentes categorías de los comercios, poder activar o desactivar cualquier tipo de usuario, actualizar provincias y municipios, y general resolver cualquier problema que surja.
- Funcionalidad de “Recuperar contraseña de usuario”.
- Funcionalidad para que el comercio pueda añadir imágenes en su perfil.
- Funcionalidad para que un comercio se pueda dar de alta en varias categorías a la vez.
- Limitar el número de productos que un comercio puede dar de alta.
- Nuevo menú, para que los clientes puedan evaluar a los comercios, poniendo puntuaciones o comentarios.
- Añadir filtros en el buscador para poder filtrar por municipio, provincia o código postal.
- Implementar una geolocalización, para poder realizar búsquedas más exhaustivas.
- Crear un sistema de envío de mensajes directos entre el comercio y el cliente.
- Que el usuario pueda eliminar tanto la cuenta comercio, como la cuenta cliente.
- Desarrollar las páginas de privacidad y políticas del footer.
- Comprobar la seguridad del producto y los datos que utiliza, como confidencialidad o privacidad.

Mejoras técnicas:

- Desplegar la aplicación en un servidor de pruebas sin necesidad de instalarla de manera local.

- Aplicar el preprocesador de estilos *Sass* en las hojas estilos CSS, para poder usar variables, funciones, etc., y así mantener las hojas de estilos bien organizadas y poder compartir estilo de diseños.

- Crear una guía de estilos y metodologías CSS, ya que es importante que se mantenga un estilo coherente para asegurarnos que el código es flexible, reutilizable, comprensible, manejable y, en definitiva, de mejor calidad. Como para este proyecto hemos utilizado Bootstrap, sería recomendable seguir la guía de estilo de Mark Otto, creador de Bootstrap.

- Optimización del formato de imágenes responsive, para esto tenemos que ofrecer el navegador varias versiones de una imagen, y que sea el mismo navegador quien seleccione aquella más adecuada en función de varios parámetros. Añadir la herramienta *parcel-plugin-imagemin* para optimizar imágenes de forma automática.

- Comprobación de nuestro sitio web para saber si es responsive. Mediante las *Herramientas para desarrolladores web*, con el botón *Modo de diseño receptivo*.

- Comprobar que todas las páginas cumplen con los criterios de accesibilidad al contenido web, mediante la web https://validator.w3.org

- Realizar el test de *Google PageSpeed Insights* sobre la aplicación, para conocer el rendimiento real de las páginas tanto en dispositivos móviles como en ordenadores.

- Implementación de test automatizados para garantizar la integridad de nuestro código. Utilizar test unitarios para asegurar de que el código funciona como se espera que funcione. Realizar testing con *Karma y Jasmine*.

- Revisar los *Code Smells* de nuestra aplicación, para detectar los errores más comunes. Aplicar las técnicas de *Refactoring* para conseguir mejorar nuestro código para que sea más fácil de entender y modificar sin crear nuevas funcionalidades.

- Aplicar técnica *Lazy Loading* para la mejora del rendimiento y velocidad de la aplicación, consiguiendo que el usuario final tenga una experiencia de navegación más agradable y satisfactoria. Las aplicaciones de Angular cargan todos los componentes que están importados en nuestro módulo principal app.module.ts y dependiendo de los módulos que contenga puede ser que tarde más o menos en cargar nuestra aplicación. Para resolver ese problema en el que a veces cargamos módulos que igual no utilizaremos en ningún momento, aplicaremos lo que es Lazy Loading. La carga diferida (lazy loading) consiste en retrasar la carga o inicialización de un objeto hasta el mismo momento de su utilización.

11. Conclusiones

Cuando comencé el TFM, me sentí por poco agobiada de como empezar a desarrollar la aplicación. Después de dos años cursando las asignaturas y aprendiendo todos los recursos adquiridos, no tenía muy claro como implementar todo ese conocimiento en la aplicación ni por dónde empezar.

Luego conforme avanza las semanas, el TFM va cogiendo forma, la idea inicial va evolucionando y cambiando a lo largo de su desarrollo. Poco a poco se va creando toda la lógica del proyecto, se crean los prototipos para tener la primera idea clara de cómo será la aplicación. Aunque esos prototipos se van modificando con el desarrollo del Frontend para acabar con un diseño más exhaustivo y propio.

He intentado aplicar al TFM, el mayor número de recursos, técnicas, herramientas y framework aprendidos en las asignaturas, muchas cosas se han quedado por el camino por falta de tiempo, sobre todo las mejoras técnicas definidas en el punto de proyección a futuro y posibles mejoras. Al final las obligaciones laborales y familiares me impiden llevar a cabo todo lo que en un principio tenía en mente para el desarrollo de este TFM.

Aunque se han alcanzado los objetivos principales, me hubiera gustado dedicarle más tiempo al estilo del diseño de las paginas, he priorizado la funcionalidad al diseño, y esto ha creado una aplicación muy sencilla en el resultado de los estilos. Y cuanto, al diseño responsive, lo mismo, no he podido mejorar la aplicación para adaptarla a los distintos dispositivos.

Aun así, estoy muy satisfecha con el TFM presentado y a ver podido resolver todos los problemas que han ido surgiendo, para completar una aplicación con los objetivos marcados, dando un resultado óptimo, donde seguir mejorando.

12. Bibliografía

Web:

[1] - https://localy.es/razones-para-comprar-pequeno-comercio/ (05/10/2022)

[2] - https://www.aseyacovi.org/2021/07/10-ventajas-al-comprar-en-el-comercio-local/ (05/10/2022)

[3] - https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/ (10/10/2022)

[4] - https://www.yoseomarketing.com/blog/reglas-usabilidad-web-de-jakob-nielsen/ (23/10/2022)

[5] - https://www.uifrommars.com/10-reglas-heuristicas-como-aplicarlas/ (23/10/2022)

[6] - https://www.gettyimages.es/ (26/10/2022)

[7] - https://jwt-auth.readthedocs.io/en/develop/laravel-installation/ (25/11/2022)

Anexo 1. – Entregables del proyecto

- Carpeta PEC_FINAL_HuertaRubio_Rebeca, que contiene:

  ✓ Carpeta Documentación:

  - Carpeta Diagramas: contiene los diagramas de casos, de usos, entidad-relación y el sitemap.
  - Carpeta Prototipos: contiene los diseños de los prototipos Lo-Fi, realizados con Excalidraw y los diseños de los prototipos Hi-Fi, realizados con HTML y CSS.
  - Archivo Diagrama de Gantt v_4.xlsx
  - Archivo PEC_FINAL_mem_HuertaRubio_Rebeca.pdf
  - Archivo PEC_FINAL_prs_HuertaRubio_Rebeca.pptx
  - Archivo PEC_FINAL_prs_HuertaRubio_Rebeca.pdf
  - Informe_Autoevaluacion_TFM_HuertaRubio_Rebeca.pdf

  ✓ Carpeta Proyecto:

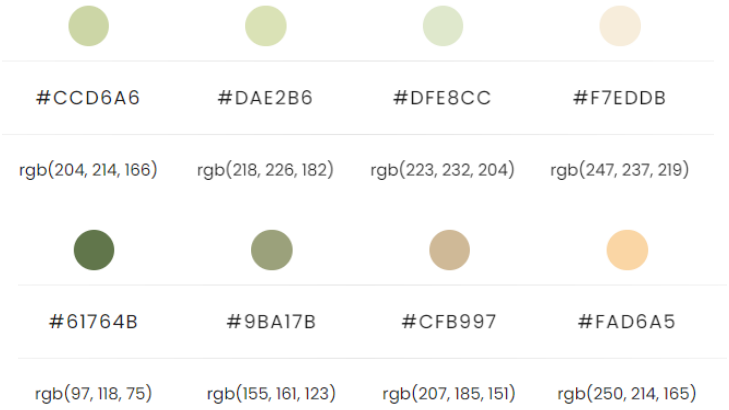
  - Carpeta PEC_FINAL_pry_HuertaRubio_Rebeca:
    - Carpeta API: archivos fuente del Backend realizados con Laravel (no incluye carpeta vendor).
    - Carpeta FRONT: archivos fuente del Frontend realizado con Angular (no incluye carpeta node-modules)
  - Archivos insert_municipios.txt y insert_provincias.txt

Anexo 2. – Libro de estilos

Con el libro de estilos definimos las pautas y criterios seguidos para desarrollar la parte visual del proyecto.

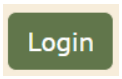
- Icono: se ha creado un logotipo con la combinación de una imagen y un texto.

- Paleta de colores: se ha usado la web https://colorhunt.co/, para obtener las siguientes paletas de colores.



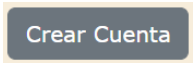
- Tipografía: para el texto se ha utilizado la font-family: Verdana, Geneva, Tahoma, sans-serif;

- Botones: se han creado algunos estilos propios y otros se han utilizado los estilos de los botones personalizados de Bootstrap.



estilo: btn btn-btv

estilo btn btn-form



estilo: btn btn-secondary

- Imágenes: las imágenes que se han utilizado en la aplicación están obtenidas de las webs www.gettyimages.es y www.freepik.es

Anexo 3. – Manual de instalaciones

1- Instalación servidor local – XAMPP

XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl.

Descargamos la versión de Xampp (XAMPP para Windows 7.4.30, 8.0.23 & 8.1.10) desde la web https://www.apachefriends.org/es/index.html

Incluye: Apache 2.4.54, MariaDB 10.4.24, PHP 8.1.10, phpMyAdmin 5.2.0, OpenSSL 1.1.1, XAMPP Control Panel 3.2.4, Webalizer 2.23-04, Mercury Mail Transport System 4.63, FileZilla FTP Server 0.9.41, Tomcat 8.5.78 (with mod_proxy_ajp as connector), Strawberry Perl 5.32.1.1 Portable

2- Instalación Laravel

Primero instalamos Composer que es el gestor de dependencias de PHP. Para instalarlo se deben seguir las instrucciones desde la web https://getcomposer.org/doc/00-intro.md

Para la instalación de Laravel (https://laravel.com/docs/9.x) se puede crear nuevos proyectos instalando globalmente el instalador de Laravel a través de Composer:

```
C:\xampp\htdocs>composer global require laravel/installer
C:\xampp\htdocs> laravel new TFMRebecaHuerta
C:\xampp\htdocs> cd TFMRebecaHuerta
C:\xampp\htdocs\TFMRebecaHuerta> php artisan serve
```

Configuramos el archivo. env con los datos de nuestra base de datos creada para el proyecto:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tfmrebecahuerta
DB_USERNAME=rhuertar
DB_PASSWORD=NrmB8x_mV5SCgNLa
```

El código para la parte de Laravel, se puede obtener desde este repositorio:
https://github.com/rebecahuertar/project-api

Meter la carpeta dentro del directorio C:\xampp\htdocs, ejecutar **composer install**, y a continuación, ejecutamos las migraciones para crear las tablas en la Base de datos, mediante:

php artisan migrate

Antes de pasar los datos con los seeders, vamos a pasar los datos de las provincias y municipios desde la aplicación PhpMyAdmin, ya que serían muchos datos para pasarlos en un seed.

Copiar el contenido de los archivos insert_provincias.txt y insert_municipios.txt, que se encuentran en la ruta PEC_FINAL_HuertaRubio_Rebeca\Proyecto y ejecutar el insert desde la pestaña SQL.

Una vez que ya tenemos los datos de las provincias y municipios, ya podemos ejecutar el seed, para meter datos en las demás tablas.

php artisan db:seed

Ejecutar **php artisan serve**

3- Instalación Angular

Necesitamos instalar Node.js para poder gestionar todas las dependencias de paquetes y herramientas que utiliza el proyecto, por lo tanto, iremos a la página oficial de Node:

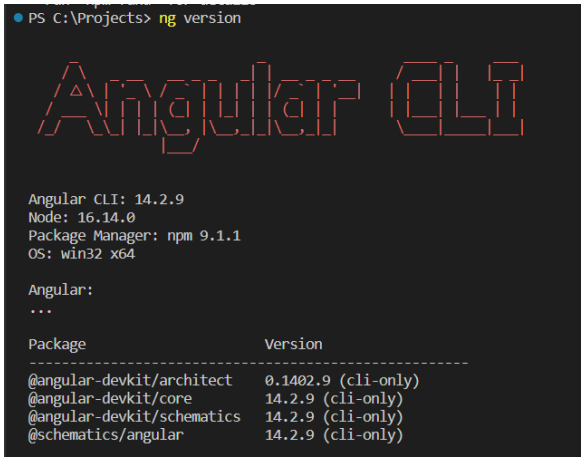
https://nodejs.org/en/

Instalad la versión LTS y ejecutamos el instalable que se nos ha descargado.

Instalar Angular mediante el siguiente comando:

npm install -g @angular/cli

Validamos las instalaciones: **ng version**



El código para la parte de Angular, se puede obtener desde este repositorio: https://github.com/rebecahuertar/project-front

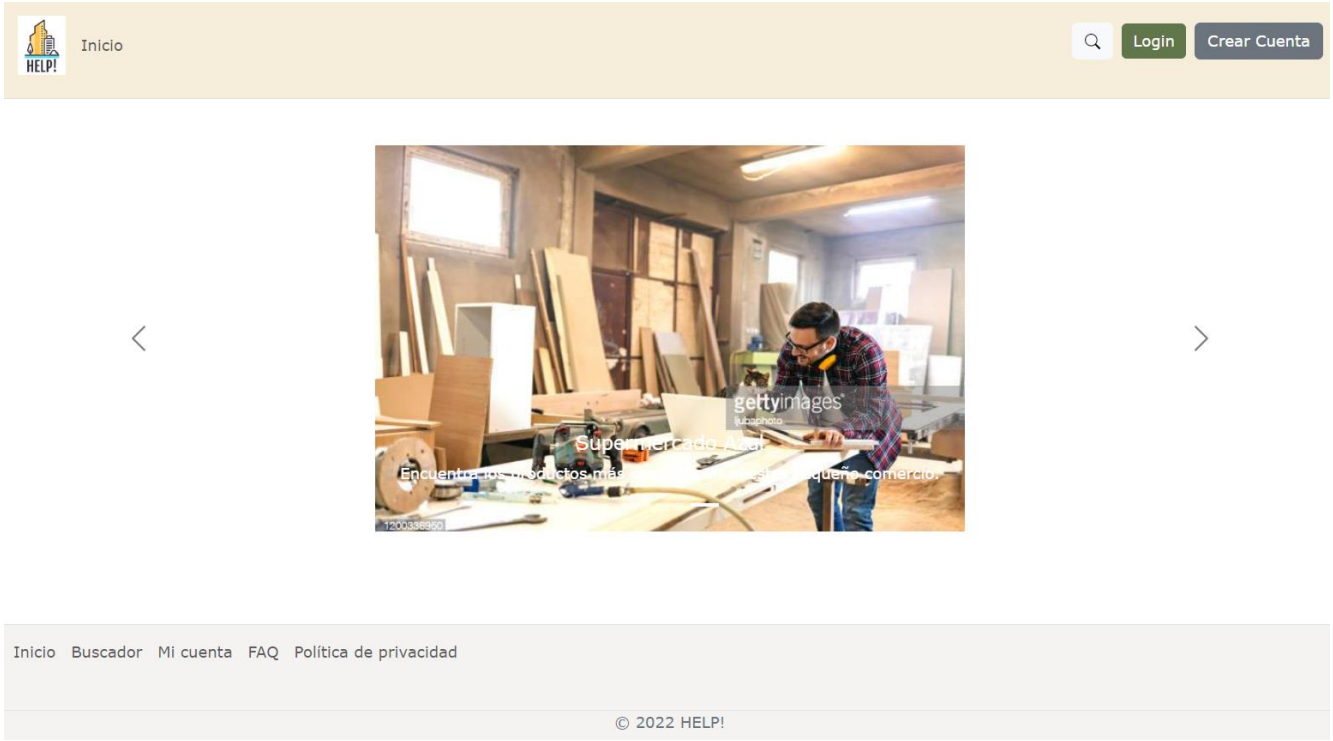
Meter la carpeta dentro del directorio creado para Angular y ejecutar **npm install**.

Ejecutar **ng serve –open** y el Frontend se abrirá desde la dirección: http://localhost:4200/

Anexo 4. – Manual de usuarios

Aunque la aplicación es una web sencilla, en este apartado se detallaran las instrucciones de uso para los diferentes tipos de usuarios.

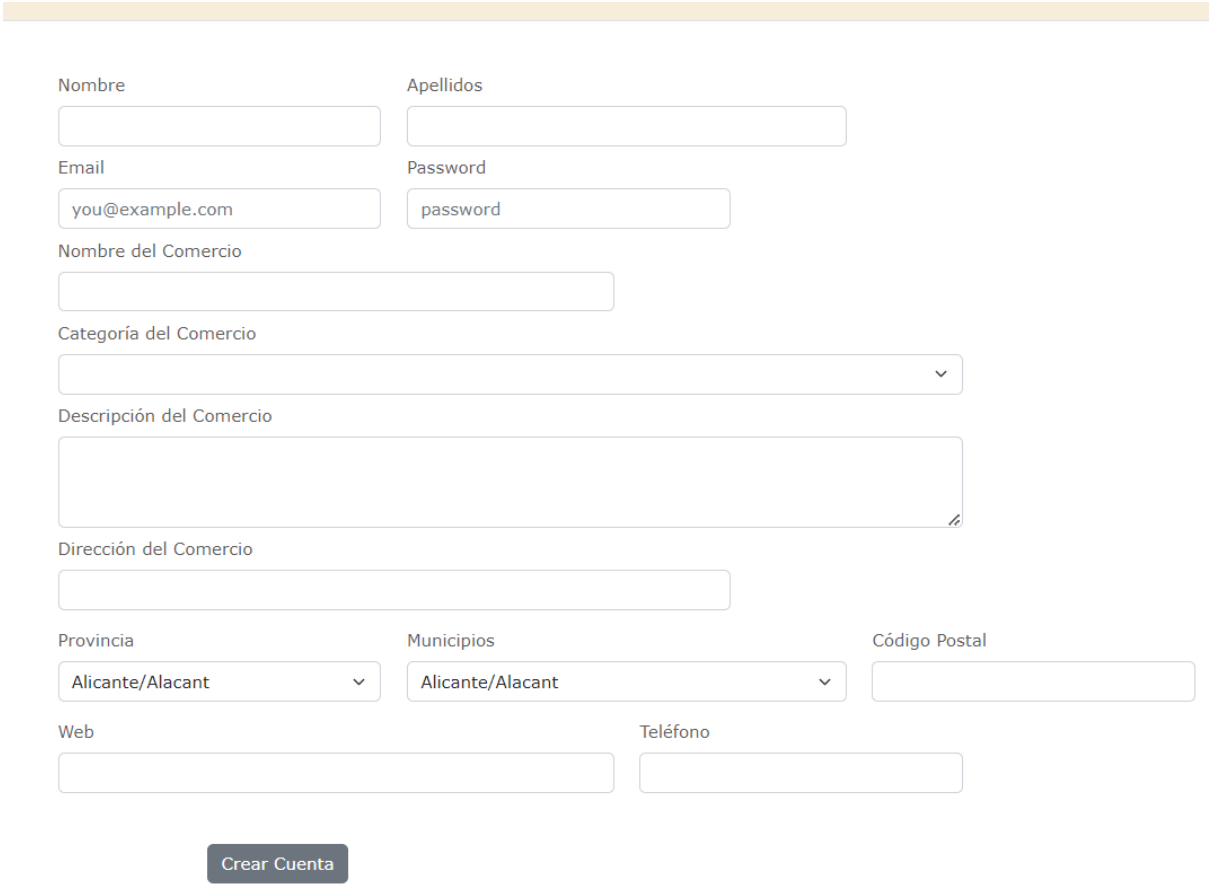
La primera página es la página de Inicio:



En el *Header* tenemos en la parte izquierda el logo y menú Inicio, y en la parte derecha, un **Buscador** y los botones **Login** y **Crear cuenta**.

Empezamos por el botón **Crear cuenta**, desde aquí el usuario puede crear una cuenta de comercio o una cuenta de cliente. En la primera pantalla tiene que elegir qué tipo de usuario quiere dar de alta.

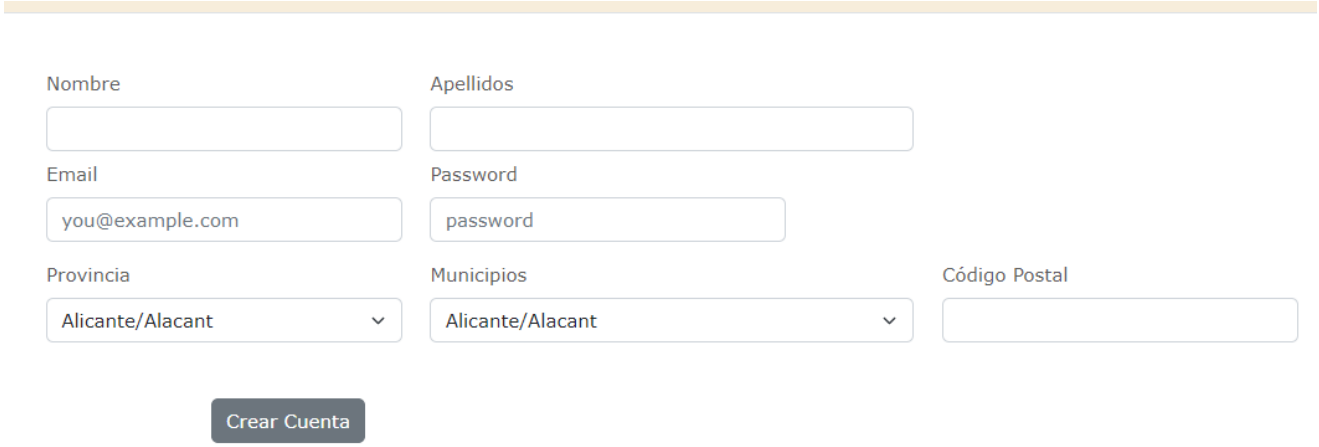
- Si elegimos “Comercio”, nos saldrá el siguiente formulario:



Todos los campos son obligatorios, menos en campo “Web” y “Teléfono”. Los campos “Nombre” y “Apellidos”, se refiere al dueño o encargado del comercio. Dentro de “Categoría del Comercio”, tenemos un desplegable donde tenemos varias opciones para elegir el tipo de comercio que se quiere dar de alta, solo se puede elegir una opción.

Una vez que completemos todos los datos, estos sean correctos y se compruebe que el email no está dado de alta, se efectuará el registro correctamente y nos llevará a la página de Login.

- Si elegimos “Cliente”, nos saldrá el siguiente formulario:



Para este tipo de usuario tenemos que completar menos información. Todos los campos son obligatorios. “Nombre” y “Apellidos” del usuario, un “Email” y “Password” para el Login, “Provincia”, “Municipio” y “Código Postal”, donde reside el cliente.

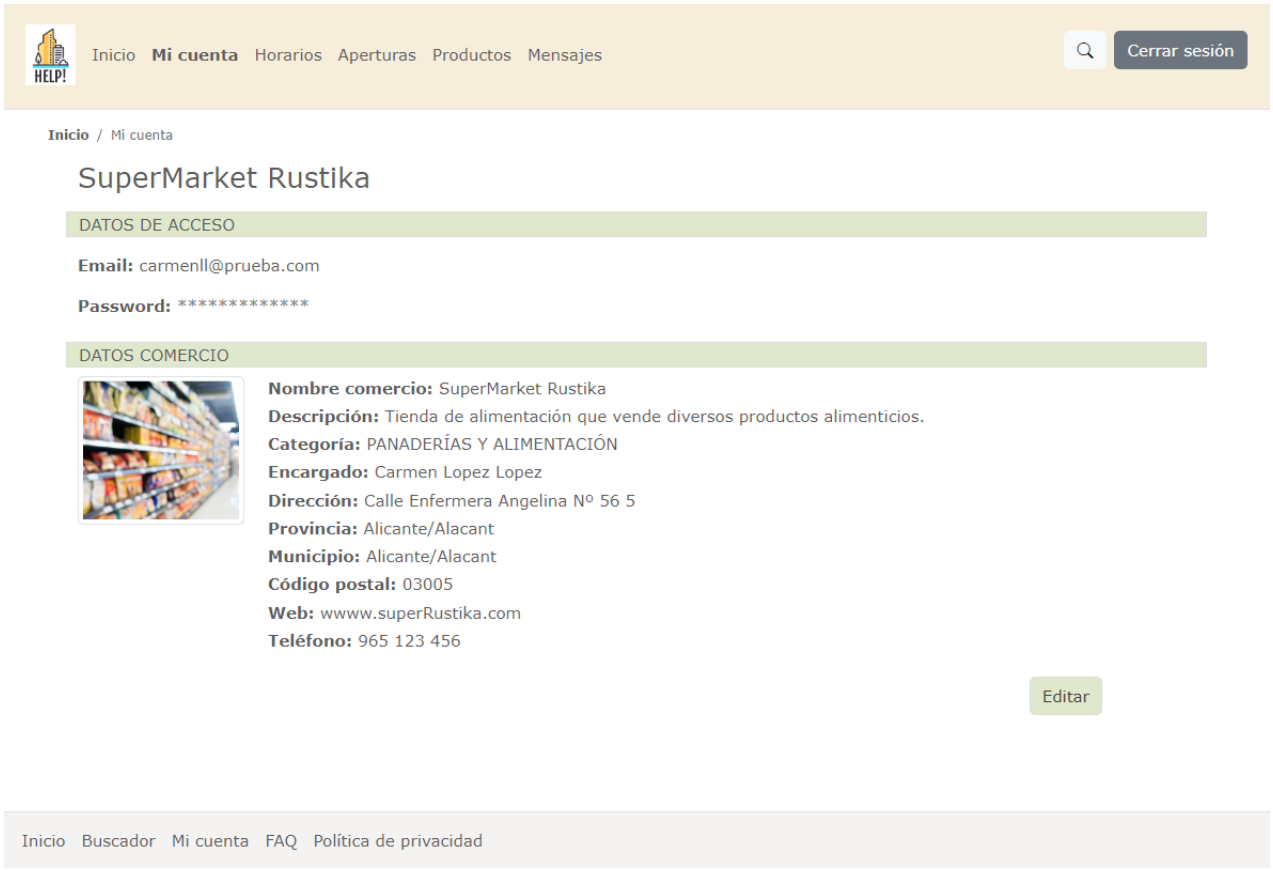
Una vez que completemos todos los datos, estos sean correctos y se compruebe que el email no está dado de alta, se efectuará el registro correctamente y nos llevará a la página de Login.

Dentro del botón **Login**, tenemos el formulario para iniciar sesión.

Vamos a iniciar sesión con un usuario de tipo **comercio**, los datos los siguientes:

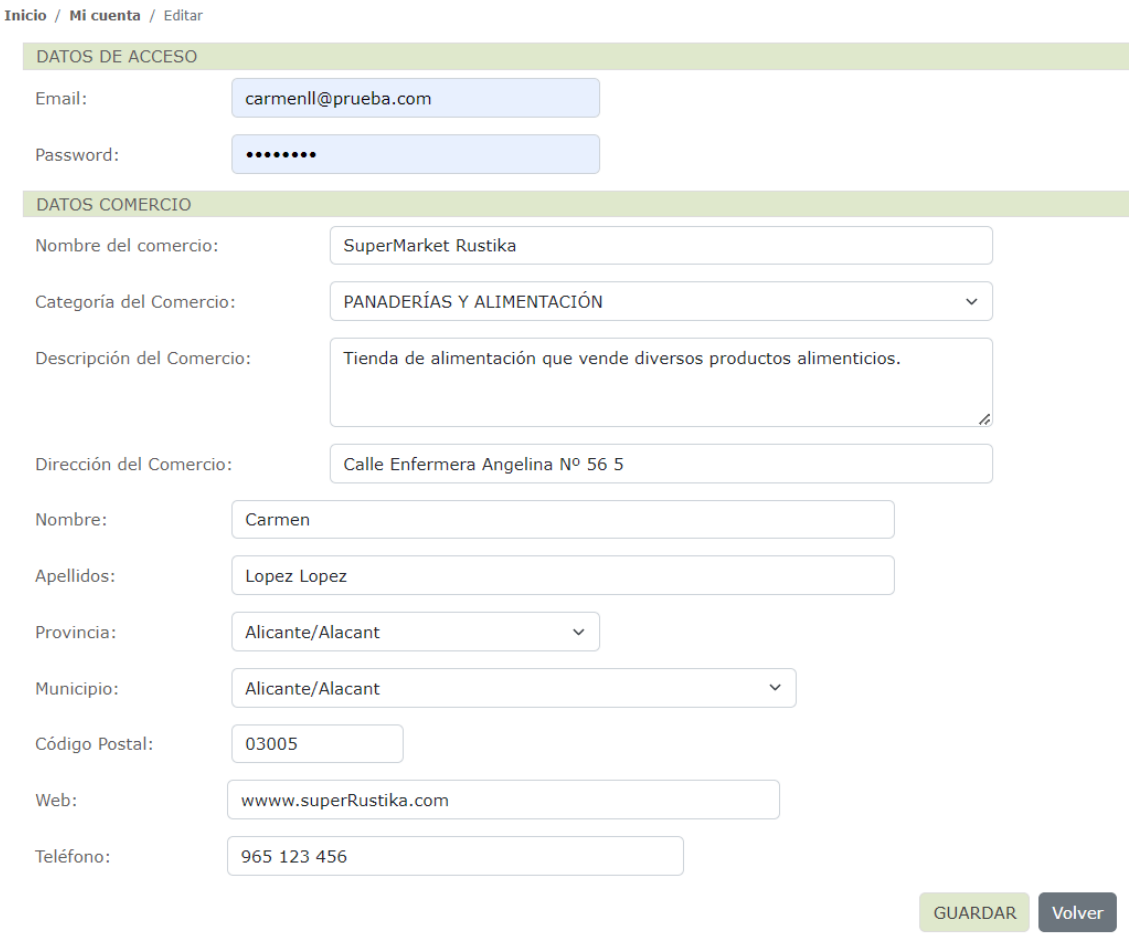
Email: carmenll@prueba.com
Password: C4781348

Una vez que iniciamos sesión, nos aparece la siguiente pantalla:

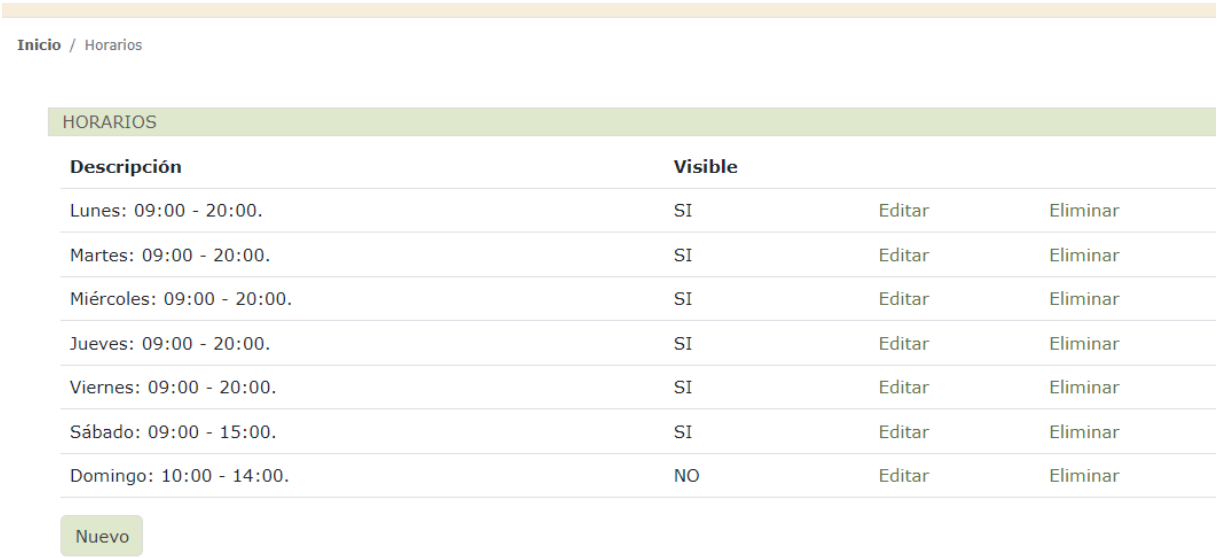


Como podemos observar en el menú izquierdo, se han añadido más menús:

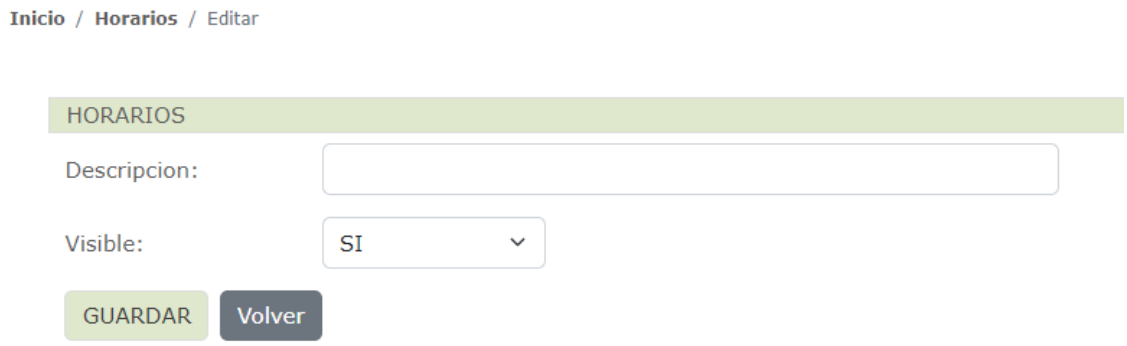
- Menú **Mi cuenta**: desde esta pantalla podemos ver nuestros datos de acceso y nuestros datos del comercio, con los cuales nos hemos dado de alta. Tenemos un botón de Editar, para poder modificar estos datos.



- Menú **Horarios**: listado con los horarios que el comercio ha dado de alta. Tenemos una Descripción y un campo Visible, con los valores Si o No, esto sirve para el comerciante pueda dar de alta un horario y pueda elegir cuando será visible para el cliente.



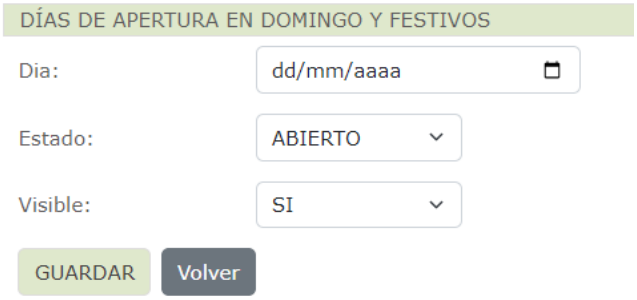
Tenemos la opción de Editar o Eliminar ese horario, y también dar de alta uno nuevo, mediante el siguiente formulario:



Debemos rellenar el campo “Descripción”, y elegir la opción Si o No será visible para el cliente o usuario que busque este comercio.

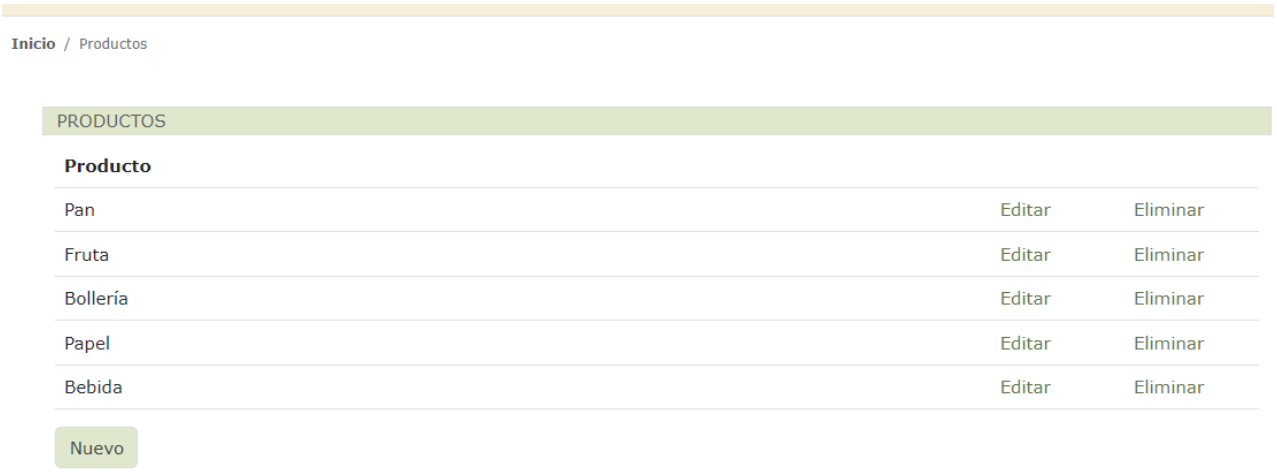
- Menú **Aperturas**: listado con los días de apertura en domingos y festivos que el comercio ha dado de alta. Tenemos el campo Día, Estado y un campo Visible, con los valores Si o No, esto sirve para el comerciante pueda dar de alta un día y pueda elegir cuando será visible para el cliente.

Tenemos la opción de Editar o Eliminar ese día, y también dar de alta uno nuevo, mediante el siguiente formulario:

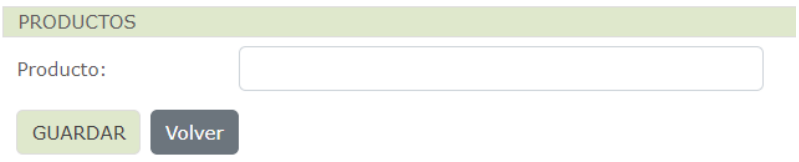


Debemos rellenar el campo “Día”, el “Estado” si estará Abierto o Cerrado y elegir la opción Si o No será visible para el cliente o usuario que busque este comercio.

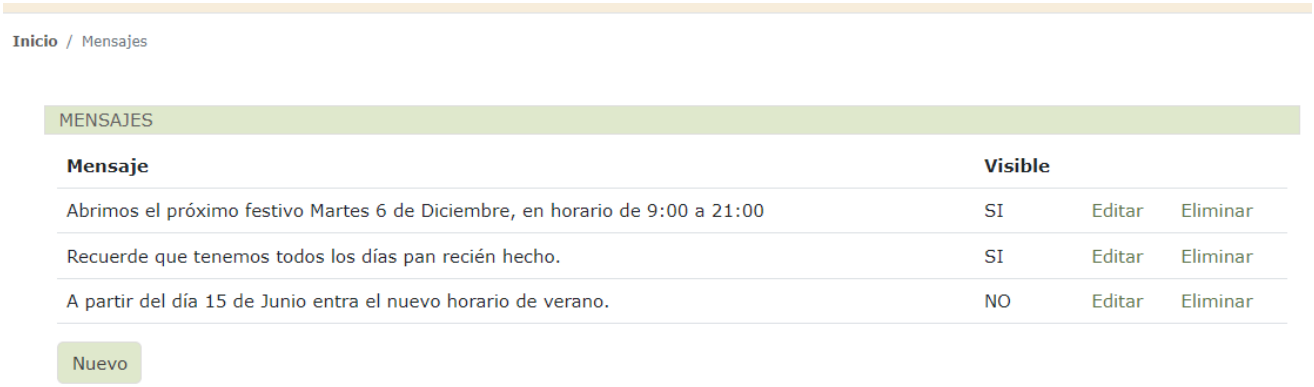
- Menú **Productos**: listado con los productos de más demanda del comercio. Este listado será importante, ya que estos productos se podrán buscar en el buscador, y así podrá salir este comercio en los resultados.



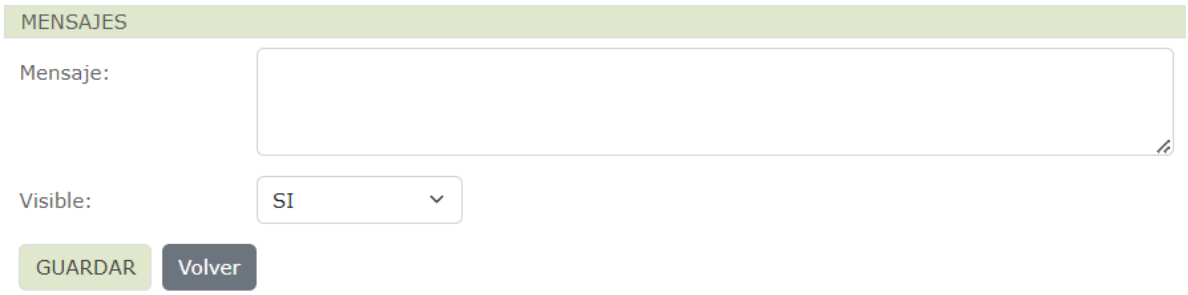
Tenemos la opción de Editar o Eliminar ese producto, y también dar de alta uno nuevo, mediante el siguiente formulario:



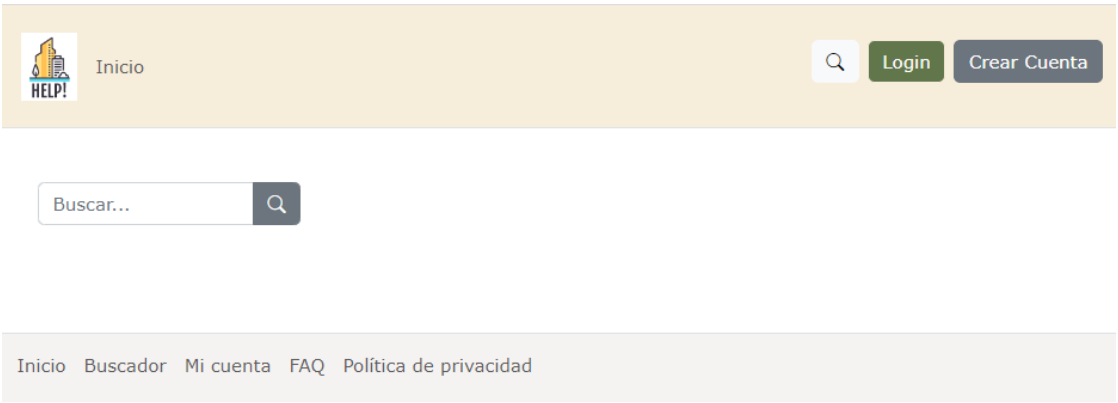
- Menú **Mensajes**: listado con los mensajes que el comercio ha dado de alta. Tenemos el campo Mensaje y un campo Visible, con los valores Si o No, esto sirve para el comerciante pueda dar de alta un mensaje y pueda elegir cuando será visible para el cliente.



Tenemos la opción de Editar o Eliminar ese mensaje, y también dar de alta uno nuevo, mediante el siguiente formulario:



Ya hemos visto toda la parte del menú del usuario tipo comercio, vamos a explicar ahora como se verían estos datos desde el buscador.



Para realizar una búsqueda no será necesario iniciar sesión. Si, por ejemplo, ponemos en el buscador la palabra “super”, nos dará un resultado de dos comercios, que tienen en el nombre del comercio la palabra “super”. El buscador busca por *Nombre del Comercio*, *Descripción del Comercio* y *Productos* dados de alta.

Si pulsamos sobre el botón Ver, podemos ver en detalle los datos de ese comercio, sus datos, dirección, categoría, web, teléfonos de contacto, horarios, días de apertura en domingos y festivos, mensajes de interés que ha puesto el comerciante y sus productos más demandados.

En el listado de horarios, días de apertura y mensajes, solo se mostrarán los que el comerciante ha elegido la opción Visible a Si, como hemos explicado anteriormente en los apartados correspondientes.

SuperMarket Rustika

DATOS COMERCIO

Nombre comercio: SuperMarket Rustika
Descripción: Tienda de alimentación que vende diversos productos alimenticios.
Categoría: PANADERÍAS Y ALIMENTACIÓN
Encargado: Carmen Lopez Lopez
Dirección: Calle Enfermera Angelina Nº 56 5
Municipio: Alicante/Alacant
Provincia: Alicante/Alacant
Código postal: 03005
Web: wwww.superRustika.com
Teléfono: 965 123 456

HORARIOS

Lunes: 09:00 - 20:00.
Martes: 09:00 - 20:00.
Miércoles: 09:00 - 20:00.
Jueves: 09:00 - 20:00.
Viernes: 09:00 - 20:00.
Sábado: 09:00 - 15:00.

DÍAS DE APERTURA EN DOMINGO Y FESTIVOS

06/12/2022	ABIERTO
08/12/2022	CERRADO

MENSAJES

Abrimos el próximo festivo Martes 6 de Diciembre, en horario de 9:00 a 21:00
Recuerde que tenemos todos los días pan recién hecho.

PRODUCTOS

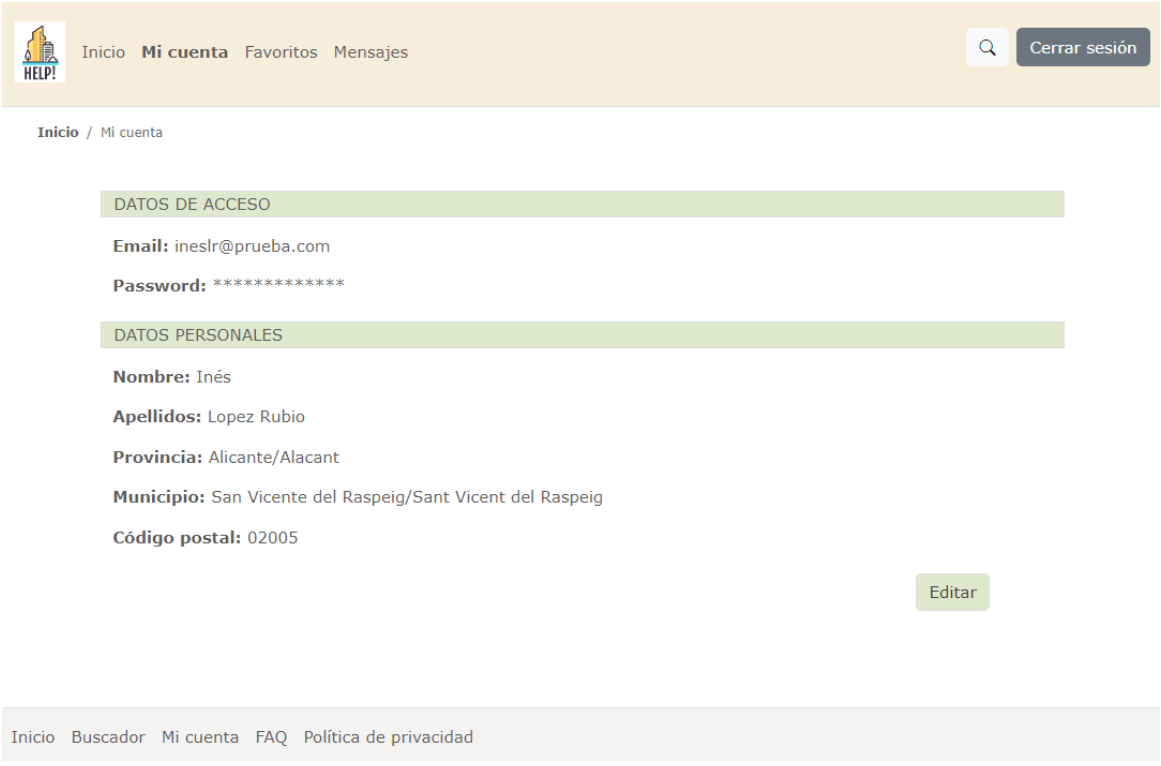
Pan
Fruta
Bollería
Papel
Bebida

Como podemos observar, de momento, no nos aparece ninguna opción para poder guardar este comercio como favorito, para esto, tenemos que iniciar sesión con un usuario tipo cliente.

Volvemos al botón de Login e iniciamos sesión con un usuario de tipo **cliente**, los datos los siguientes:

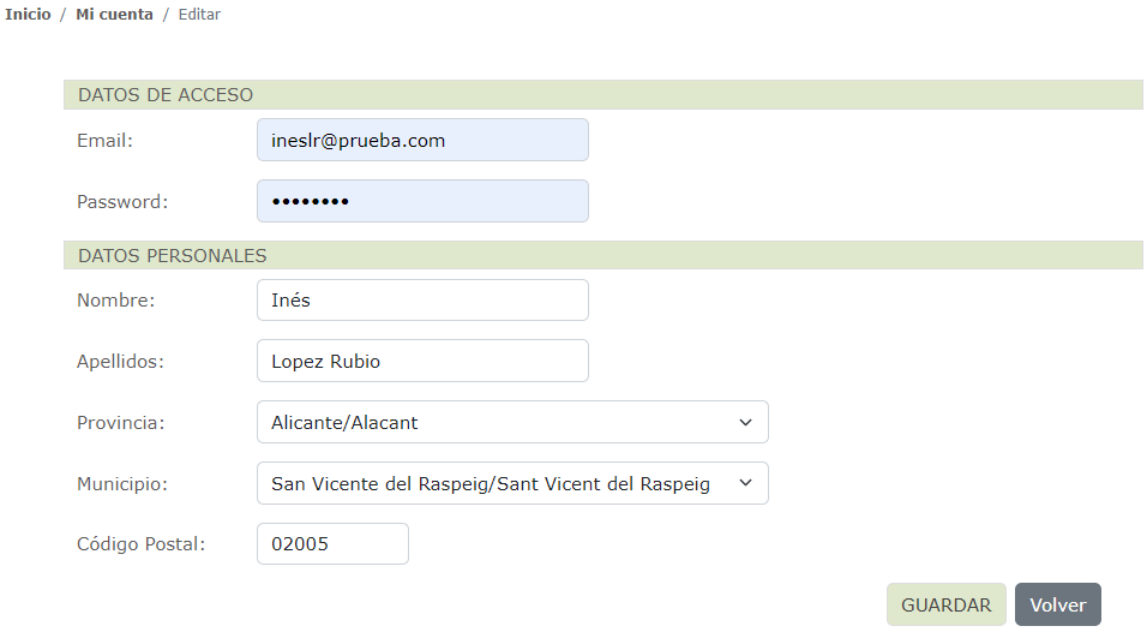
Email: ineslr@prueba.com
Password: I7462596

Una vez que iniciamos sesión, nos aparece la siguiente pantalla:



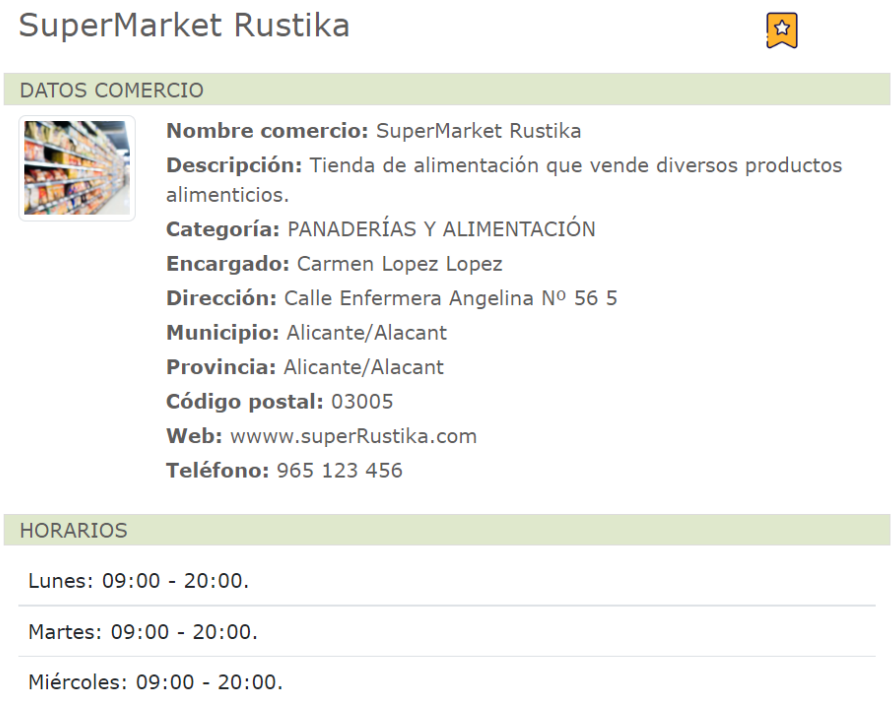
Como podemos observar en el menú izquierdo, se han modificado los menús:

- Menú **Mi cuenta**: desde esta pantalla podemos ver nuestros datos de acceso y datos personales, con los cuales nos hemos dado de alta. Tenemos un botón de Editar, para poder modificar estos datos.



- Menú **Favoritos**: desde aquí podemos consultar el listado de nuestros comercios favoritos.

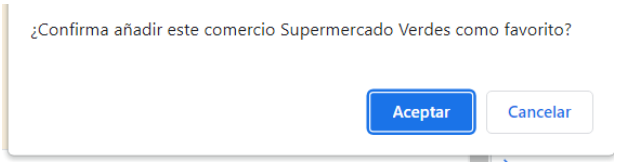
Para ver en detalle el comercio, pulsamos sobre "**Ver**". Como podemos observar nos aparece el icono de favoritos en naranja, eso significa que ya tenemos este comercio como favorito en nuestro listado.



Para añadir un nuevo comercio favorito a nuestro listado, nos vamos al buscador para realizar una búsqueda. Y vamos a añadir este supermercado como favorito. Pulsamos sobre el botón Ver.

Como podemos observar ahora este icono aparece en color blanco, lo cual significa que de momento no tenemos este comercio dentro de nuestros favoritos.

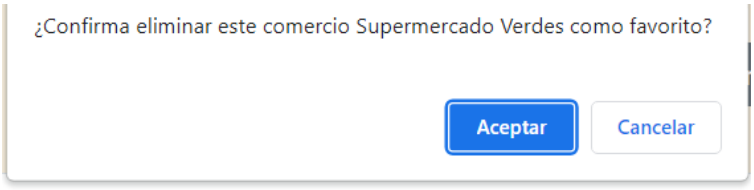
Si pinchamos sobre él, nos saldrá una ventana emergente preguntado si queremos confirmar añadir este comercio como favorito.



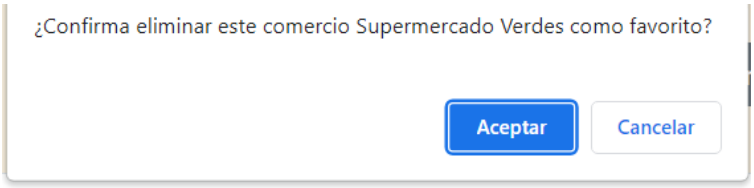
Si aceptamos, el icono se volverá naranja, y al volver al menú de favoritos ya tendremos añadido el nuevo comercio a nuestro listado.

Para quitar algún comercio como favorito, se puede de hacer de dos formas:

- Pulsando sobre “Quitar de favorito” y dando a Aceptar.



- Pulsando sobre el icono naranja , dentro del detalle del comercio.

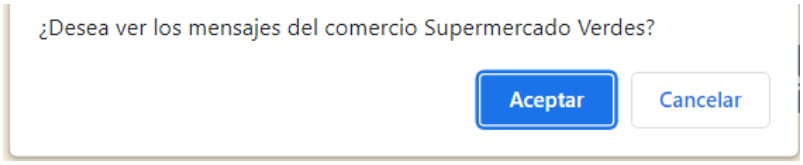


- Menú **Mensajes**: desde aquí podemos consultar los mensajes que van publicando nuestros comercios favoritos. En el primer listado “Mensajes de comercios favoritos” tenemos los mensajes de los comercios que si queremos ver. En el segundo listado “Activar mensajes de comercios favoritos”, tenemos los comercios favoritos que no queremos que aparezcan sus mensajes en el listado de arriba.

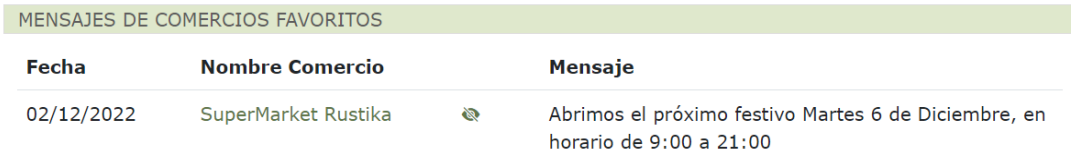
Como podemos activar o desactivar estos mensajes:

- Activar: pinchando sobre el icono que aparece al lado del comercio.

Nos saldrá una ventana emergente para confirmar.



- Desactivar: pinchando sobre el icono que aparece al lado del comercio.



Nos saldrá una ventana emergente para confirmar.