Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Realización de una Tienda On-Line para empresa local

Autor:
D. Pablo Ortega Molinero

Tutores:
Dr. Carlos Enrique Vivaracho Pascual

Resumen

Este Trabajo de Fin de Grado consiste en el desarrollo de una tienda on-line para la venta de productos veganos para un negocio minorista local.

A raíz de la situación provocada por la pandemia de COVID-19 que se originó en el año 2020, muchas empresas de venta de productos alimentarios se han visto obligadas a la búsqueda de nuevos canales de venta para sus productos. Hablando con un amigo de la infancia me propuso la creación de una tienda on-line de productos para su empresa alimentaria.

La idea principal consiste en implementar un sistema basado en web para la gestión de pedidos on-line a la empresa, y gestione el cobro de los mismos.

La propuesta es una pagina web en la que los usuarios podrán visualizar en catálogo de productos proporcionados por la empresa, añadirlos a un carro de compra y realizar un pedido a la empresa para que los productos sean enviados a su casa. El sistema se pretende que esté conectado a alguna plataforma de gestión de pagos on-line para que los pedidos puedan ser abonados directamente.
El sistema contará con un sistema de stock para saber los productos que hay disponibles en la tienda al momento de hacer un pedido.

Se estudiarán diversas tecnologías para la realización del proyecto y se mantendrán diversas entrevistas con los clientes para la creación del sistema.

Abstract

The main objective of this projet is to develop an e-commerce for a local vegan chocolate company. The resulting aplication will allow ald an new users to buy the products of the company for been sent to their homes. The company will have the posibility of managing their product catalog and all their orders.

The tools used for the developement of this project have been Angular as the front-end technology and Firebase as the back-end service provider.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto y motivación

En el año 2020, una pandemia global sin precedentes nunca antes conocida por la sociedad moderna sacudió al mundo entero.

La situación provocada por la cuarentena provocó que los consumidores optasen explorar nuevas vías para la adquisición de productos que no implicase el contacto con otras personas:

- La gente empezó a realizar pedidos de forma telefónica a las tiendas de comestibles para ir a recogerlos y pasar el menor tiempo en ellas o para que fueran entregados en sus casas.
- También los clientes de tiendas de ropa empezaron a realizar sus pedidos de forma on-line.

Con todas estas medidas, muchas industrias comenzaron la búsqueda de nuevas formas de ofertar sus productos. Algunas optaron por informar por vía telefónica a sus clientes de los productos que tenían en venta (numerosas tiendas de comestibles hicieron uso de este medio de comunicación). Otras optaron por la creación de sitios web para poder realizar pedidos para que fueran entregados en las casas de los consumidores, y es este segundo caso el que nos interesa.

En medio de la pandemia, una empresa de mi localidad natal, "Dulces Típicos El Beato"de El Burgo de Osma, dedicada a la producción de chocolates, dulces, bebidas espirituosas y a su distribución por todo el territorio español decidió crear una página web para que sus clientes pudiesen realizar pedidos a la empresa de forma on-line ya que hasta ese momento sus productos solo se ofertaban en tiendas físicas y estaciones de servicio.

Al crear esta nueva web el volumen de ventas, lejos de disminuir a causa de la pandemia, aumentó.

Viendo el éxito obtenido con la creación de la tienda on-line, ahora esta empresa busca aumentar su oferta de productos con una nueva linea de productos veganos, la cual desean que se oferte fuera de su tienda on-line actual. Es por ello que decidieron crear una nueva tienda on-line.

Para esta tarea, el dueño de la empresa se acercó a solicitar mis servicios para llevar a cabo el proyecto de creación esta nueva tienda on-line. Viendo que era un proyecto técnico que se podía presentar Trabajo Fin de Grado acepté la realización de este proyecto.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo de este Trabajo de Fin de Grado es plantear la creación una tienda minorista de venta de productos on-line para una empresa de venta de productos derivados del chocolate vegano.

1.2.2. Objetivos específicos

El objetivo general planteado se divide en los siguientes objetivos específicos:

- **Muestrario de productos**: el sistema desarrollado deberá permitir a los clientes de la tienda on-line ver los productos que hay a la venta.
- **Cesta de la compra**: el sistema deberá implementar un sistema de cesta de la compra a la que los clientes puedan añadir productos o eliminarlos.
- **Categorización**: el sistema deberá permitir agrupar los productos en categorías y mostrar los productos de una categoría.
- **Acceso**: el sistema deberá ser accesible para los usuarios desde cualquier parte del territorio español al menos, y opcionalmente, global.
- **Navegación**: el sistema deberá permitir a los clientes navegar entre los distintos productos y categorías de la tienda
- **Pedidos**: el sistema desarrollado deberá permitir a los usuarios realizar un pedido con los productos que ha añadido a la cesta y deberá permitir a un usuario ver sus pedidos anteriores y el estado de estos.
- **Inventario**: el sistema desarrollado deberá permitir llevar un conteo de las existencias de los productos y modificarlas de forma automática al recibir pedidos.
- **Productos nuevos**: el sistema deberá permitir al propietario añadir nuevos productos al catálogo de productos a la venta.
- **Pago**: el sistema deberá permitir el pago de los pedidos que realicen los clientes.

1.2.3. Objetivos personales

A nivel personal, y teniendo en cuenta que este proyecto se engloba dentro del marco educativo de el Grado en Ingeniería Informática, el principal objetivo es el aprendizaje:

- Reforzar la faceta de licitación de requisitos con un caso de uso real y con un cliente real.
- Conocer distintas tecnologías para el desarrollo de sistemas web.
- Aprender a gestionar los tiempos y a hacer un seguimiento real del desarrollo de un proyecto.
- Mejorar el conocimiento personal actual sobre los frameworks de desarrollo de webs basados en JavaScript.
- Mejorar las habilidades personales en cuanto a diseño gráfico y composición de interfaces personales
- Aprender a trabajar con productos de alojamiento de servidores y conocer distintos proveedores.

1.3. Publico Objetivo

El publico al que va dirigida esta tienda on-line es todos aquellos clientes actuales de la empresa "Dulces Típicos el Beatoz a todos aquellos potenciales clientes que por falta de un sitio web donde encontrar productos derivados del chocolate que sean 100 % veganos.

Una persona vegana es aquella busca excluir en su día a día todos los productos que sean derivados de animales, o que en su producción se haya empleado de forma cruel a animales. En términos dietéticos de los sujetos, denota la practica de no consumir y evitar todos los productos derivados de animales o que contengan partes o trazas de ellos. [55]

Actualmente existen pocas empresas que realicen productos veganos derivados del chocolate, y habiendo cada día más adeptos a este modelo de alimentación, se cree una oportunidad interesante para abrirse a nuevos mercados.

1.4. Aplicaciones Web Similares

Hoy en día existen numerosos ejemplos de tiendas on-line de productos de toda índole. A modo de inspiración y en busca de ideas para realizar la tienda on-line que aquí nos ocupa se han buscado distintos ejemplos:

- **cardetail.be** [34] Cardetail es una tienda on-line especializada en la venta de productos para limpieza y detallado de coches creada por Vermijl Car Detail, un estudio de detallado de coches en Bélgica. En esta tienda venden todos los productos que se emplean en el estudio para que los clientes y consumidores de contenido multimedia de su canal de YouTube puedan comprarlos. De esta tienda cabe destacar la limpieza de la presentación de los productos y las animaciones que poseen los elementos, tales cosas hacen que la experiencia de compra sea muy agradable.
- **pccomponentes.com** [57] PcComponentes es una tienda situada en Alhama de Murcia, España, que hace unos años decidió dar el salto al comercio online de sus productos. Desde hace unos años son líderes a nivel nacional en la venta de productos de informática y tecnología en el canal online. De esta tienda cabe destacar que buscar productos dentro de ella es muy sencillo, debido a la buena categorización de los productos y a que su buscador funciona increíblemente bien.
- **prozis.com** [49] Prozis es una empresa dedicada al desarrollo de alimentación y suplementación deportiva que nació en 2007 y desde entonces son unos de los referentes a nivel nacional en cuanto a los productos que venden. De este sitio web cabe destacar la simpleza de la navegación entre las distintas categorías que posee la tienda, cuyo número es bastante elevado, y la forma que tienen de presentar sus productos. Cabe destacar que cuando se accede a ver un producto, la opción de añadir al carrito siempre se queda en la parte superior, y nunca se pierde de vista.
- **kubekings.com** [54] Kubekings es una tienda online que se dedica a la venta de cubos de rubik y artículos como puzzles y juegos de mesa. Se ubican en la localidad de Moratalla, Murcia, y ha día de hoy son referentes en la venta de productos relacionados con puzzles y cubos de rubik. De esta tienda cabe destacar que la página principal es muy limpia, prácticamente no tiene texto al principio, sino que directamente te muestra productos para que empieces a comprar nada más llegar, haciendo así que el cliente no se distraiga de la tarea principal, la compra de productos.

1.5. Estructura de la memoria

- **Capítulo 1**: capítulo en el que se presenta el proyecto y se exponen los objetivos
- **Capítulo 2**: en este capítulo se expone la planificación del proyecto, la metodología elegida para desarrollar el mismo y la adaptación de la metodología empleada al desarrollo de mismo.
- **Capítulo 3**: en este capítulo se presentan diversas opciones de tecnologías investigadas para ser empleadas en el desarrollo del proyecto.
- **Capítulo 4**: en este capítulo se exponen en profundidad las tecnologías empleadas en el desarrollo del proyecto.
- **Capítulo 5**: en este capítulo se expone el plan de riesgos del proyecto y los presupuestos del mismo.
- **Capítulo 6**: en este capítulo se expone un seguimiento de las tareas, sus estados y duraciones durante todo el proyecto.
- **Capítulo 7**: en este capítulo se exponen todos los diagramas y bocetos asociados y empleados en el desarrollo del proyecto.
- **Capítulo 8**: en este capítulo se expone cómo ha sido realizada la implementación, cómo ha quedado la estructura de paquetes y los cambios realizados en la aplicación.
- **Capítulo 9**: en este capítulo se presentan las pruebas realizadas al producto final.
- **Capítulo 10**: en este capítulo se exponen las conclusiones del devenir del proyecto.

Capítulo 2

Planificación del proyecto

2.1. Infraestructura del proyecto

Las personas que van a tomar parte en el desarrollo de este proyecto conforman la infraestructura del mismo.

De un lado “Dulces Típicos el Beatoz sus propietarios, serán el nexo de comunicación para discutir temas referentes a funcionalidad a abarcar por el sistema, diseño de la imagen del sitio web, suministro de material audiovisual para el sistema web, etc. Todo cambio en la parte de imagen del sitio web será acordado con la encargada de imagen de la empresa.

De otro lado, en la parte de equipo para el desarrollo del proyecto estaránel alumno que realiza el proyecto como el desarrollador y redactor del proyecto el tutor que actuará como asesor técnico y académico.

Como método de comunicación con los clientes por parte del lado de desarrollo se crea un grupo en una aplicación de mensajería instantánea para la rápida comunicación entre las partes. Se opta por esta opción al ser un número reducido de partes a comunicar entre sí y la distancia que existe entre el equipo de desarrollo y la parte cliente. Se opta también por esta opción dado a la naturaleza del trabajo del cliente, el cuál normalmente no está en la oficina de la empresa, sino que suele estar de viaje.

2.2. Scrum

2.2.1. ¿Qué es Scrum?

Scrum [40] [52] [47] es un marco de trabajo ligero que permite el trabajo colaborativo entre equipos. Fue creado en los años 90 como herramienta para el desarrollo y mantenimiento de proyectos complejos. Scrum está especialmente indicado para el desarrollo de proyectos en los que se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, y en los que la productividad, la innovación, y la flexibilidad son características fundamentales. Scrum se trata de un marco para el desarrollo y no de una metodología al uso.

Scrum se basa en tres pilares:

- **Transparencia:** el proceso de trabajo debe ser visible para aquellos que realizan el trabajo y aquellos que van a recibir el resultado. Para la correcta aplicación de esta transparencia es necesaria la creación de un instrumento estándar de comunicación entre las partes.

- **Inspección:** el progreso de las metas acordadas entre las diversas partes debe ser inspeccionado y revisado con frecuencia por estas partes para poder detectar a tiempo posibles variaciones de la planificación o problemas. Scrum proporciona una cadencia estándar en la que estas revisiones debieran producirse.

- **Adaptación:** si en una inspección se detectan desviaciones importantes en el progreso o en el producto, es necesario hacer ajustes para adaptar las desviaciones detectadas. La adaptación debe ser realizada lo antes posible a find e evitar desviaciones mayores a futuro.

2.2.2. Artefactos Scrum

Scrum cuenta con una serie de artefactos diseñados para maximizar la transparencia de información clave y que todos los integrantes del equipo puedan revisarlos como forma de inspección para poder detectar problemas.

Los artefactos que proporciona Scrum son:

- ***Product Backlog* (Pila de producto):** es una lista que comprende todos los necesarios para que el producto esté completo. Puede definirse como la lista de requisitos del producto/proyecto. Esta lista se crea al inicio del proyecto pero no es fija, ya que puede ir modificándose conforme se vayan detectando nuevos requisitos o partes de nuestro producto.

- ***Sprint Backlog* (Pila del Sprint):** es una lista que comprende toda la funcionalidad seleccionada de la Pila de Producto que va a ser desarrollada en un Sprint. Esta lista debe tener un alto nivel de detalle en sus elementos para que pueda ser inspeccionado el progreso en cada reunión que lo precise.

- **Incremento:** suma de elementos completados de la Pila de Producto en un Sprint. Al final del Sprint el incremento entregado debe estar terminado y ser funcional.

2.2.3. Participantes en Scrum

Un proyecto gestionado con Scrum posee una serie de participantes, los cuales podemos dividir en Equipo de Scrum e Interesados.

Interesados

Son todas aquellas personan que forman parte del proyecto pero que no pertenecen al equipo de desarrollo. Solo pueden observar el proyecto y asesorar o dar ideas al equipo de desarrollo. Suelen formar parte de este grupo los promotores del proyecto o los inversores.

Equipo Scrum

Los equipos de Scrum son auto-organizados, es decir, no son gestionados por alguien externo al equipo Scrum (como un *Manager de Proyectos*) lo que permite al equipo proponer internamente su forma de trabajo.

Los integrantes de un equipo de Scrum son:

- ***Scrum Master*:** se trata de un experto en Scrum que se encarga de que Scrum se entienda por todos los integrantes del equipo y que sea puesto en práctica. Actúa de facilitador y moderador durante las reuniones para lograr que sean productivas. Es el encargado de reconducir al equipo durante situaciones de bloqueo y de educar al equipo en materia de autogestión.

- ***Product Owner*:** se trata de el representante de los accionistas y clientes que usarán el software producido. Es el encargado de maximizar el valor del producto y el trabajo del Equipo de Desarrollo. Se focaliza por tanto en la parte de negocio. Es el encargado de mantener el *Product Backlog*, añadiendo nuevas historias de usuario según sea necesario y de priorizarlas en este. También es el encargado de crear las *User Stories*.

- **Equipo de Desarrollo:** se trata de un grupo de profesionales con los conocimientos necesarios para el desarrollo del producto. Su tarea es desarrollar las *Historias de Usuario* elegidas para un Sprint y así entregar un *Incremento*. El tamaño de este grupo de profesionales debe ser suficientemente comedido como para poder realizar trabajos de forma ágil y suficientemente grande como para poder desarrollar los trabajos asignados a tiempo.

2.2.4. Historias de Usuario

Las Historias de Usuario o *User Stories* son un elemento especial dentro del *Product Backlog*. Son descripciones cortas y simples de una característica a implementar contada desde la perspectiva de la persona que desea la nueva característica, generalmente un usuario o cliente del sistema. Su función es la de proporcionar información directa desde el cliente en caso de que algo cambie en la forma en que debe ser llevado a cabo (funcionalidad, diseño...). Se suelen tomar como los requisitos oficiales y más importantes del proyecto.

2.2.5. Eventos de Scrum

Scrum cuenta con una serie de eventos que sirven como oportunidades para adaptar los artefactos de este marco. Son eventos de duración preestablecida que se realizan de forma periódica.

- ***Sprint* o Iteración:** se trata de la unidad básica de trabajo para un equipo Scrum. Se trata de un lapso de tiempo, de duración entre 1 y 4 semanas, en las que el equipo Scrum acuerda realizar un trabajo y entregar un Incremento al final de este periodo. Es altamente recomendable que la duración de los *Sprints* de un proyecto sea fija y acordada entre el *Scrum Master* y el Equipo de desarrollo. Mientras el proyecto no se considere completo, siempre habrá un *Sprint* activo.

- **Reunión de planificación de Sprint o *Sprint Planning*:** reunión que se realiza una vez por *Sprint*, normalmente al inicio de este. en esta reunión se planifican las objetivos a completar en el *Sprint* y se define qué *Incremento* va a ser entregado. Tiene una duración de entre 2 y 4 horas normalmente, aunque puede alargarse a una jornada laboral (8 horas) completa si se precisa.

- ***Daily scrum*:** reunión de corta duración, suele rondar los 15 minutos, en la cuál se resumen las actividades llevadas a cabo por los diferentes integrantes del equipo en las 24h anteriores. En esta reunión se planifican las actividades para la siguiente jornada laboral. Esta reunión sirve para sincronizar el trabajo del equipo y para que se pueda evaluar el progreso a corto plazo.

- **Refinamiento del Backlog o *Backlog Refinement*:** evento en el que el único partícipe es el *Product Owner* y que consiste en revisar cada uno de los elementos del *Product Backlog* con el fin de esclarecer cualquier duda que pudiese surgir del equipo de desarrollo. También sirve para revisar el esfuerzo y el tiempo que se ha de dedicar a cada uno de los requerimientos.

- ***Sprint Review* o Reunión de Revisión de Sprint:** reunión en la que están presentes todos los miembros del equipo Scrum pero que es dirigida por el *Scrum Master* y el*Product Owner*, a la que acuden también los clientes del producto. En esta reunión se muestra a los clientes el trabajo completado en el Sprint y se recibe retro-alimentación de estos.

- ***Sprint Retrospective* o Reunión de Retrospectiva:** esta es la última de las reuniones de un Sprint de Scrum. Se realiza al final del Sprint y en ella el *Product Owner* se reune con el equipo de trabajo y con el *Scrum Master* para hablar sobre lo ocurrido durante el *Sprint*. Se discute sobre qué se ha hecho bien y qué se ha hecho mal, se habla sobre que inconvenientes se encontraron en el desarrollo del *Sprint* y se proponen ideas y mejoras para el desarrollo del siguiente *Sprint*.

2.2.6. Requisitos y compromisos de Scrum

Para el correcto funcionamiento de un proyecto desarrollado con Scrum es preciso que se cumplan una serie de requisitos y compromisos por parte de las partes implicadas [48]:

- **Cultura de empresa** que debe estar estar alineada con la filosofía de una gestión ágil. Se debe por lo tanto fomentar una cultura en la que trabajo en equipo, autogestión, creatividad, transparencia, mejora continua y colaboración sean los pilares fundamentales y estén bien asentados.

- **Compromiso del Cliente** con el proyecto dado que Scrum exige al cliente una alta implicación y una dedicación regular. El cliente debe de estar presente en todo momento y ayudar a tomar las decisiones pertinentes cuando sea necesario, sin imponer sus ideas al ser el que financia el proyecto.

- **Compromiso de la Dirección** la cual debe estar comprometida y apoyar el uso de Scrum, dado que habrán muchos obstáculos y se harán evidentes otros hasta ahora desconocidos en materia de organización. Será necesaria la toma de decisiones, realizar cambios en la organización, recolocar a personal y proporcionar recursos para la impnatación de Scrum.

- **Compromiso del equipo** a tener un alto grado de colaboración y de compromiso, fomentando la transparencia y el apoyo mutuo entre compañeros reforzando así la confianza.

- **Compromiso ganar-ganar:** la relación entre el proveedor y el cliente debe estar basada en una política de victoria conjunta, en vez de estar basada en los clásicos contratos cerrados a tiempo y coste. Esto quiere decir, que se debe remar en igual dirección y que se debe anteponer la calidad a los costes.

2.3. Adaptación de Scrum al proyecto

Para el desarrollo de este trabajo de fin de grado se ha decidido emplear Scrum, y dado que este proyecto va a ser realizado por un solo alumno es necesario hacer una adaptación del concepto de Scrum para poder aplicarlo al proyecto.

El alumno hará las veces de Equipo de Desarrollo y *"Product Owner"*, y sus tareas serán la creación de las historias de usuario y la implementación de estas. El cliente, al ser un cliente real, puede desempeñar de forma plena su papel en Scrum. El tutor del proyecto se encargará de hacer las veces de *"Scrum Master"* y su labor será asesorar al alumno en lo referente a Scrum y en otros aspectos técnicos.

Se decide que la duración de cada Sprint será de dos semanas de jornadas laborales, o sea 14 días de trabajo. Las reuniones de Planificacion y revisión se harán al comienzo y fin de cada Sprint. Se harán en

la medida de lo posible juntas en el mismo día para no interceder en las tareas personales de las partes implicadas.

El Product Backlog, el Sprint Backlog y el incremento se explicarán cada uno en un apartado de esta memoria específico.

2.3.1. Planificación

Este proyecto es un Trabajo de Fin de Grado del Grado en Ingeniería Informática de la Universidad de Valladolid, los cuales tienen una asignación total de 12 créditos ECTS [39]. Viendo que cada crédito ECTS consta de unas 25 horas de trabajo [32], se puede ver que la carga de trabajo debería rondar las 300 horas totales.

La fecha en la que se desea entregar este proyecto es antes del 24 de Junio del año 2022, por lo tanto se planifica este proyecto para que acabe antes de esa fecha, creando así un márgen entre la fecha de finalización prevista y la fecha final de entrega por si, en el transcurso del proyecto, surgiesen imprevistos.

Con lo expuesto anteriormente la planificación inicial de los sprint quedaría de la manera expuesta en la Tabla 2.1:

Sprint	**Fecha de inicio**	**Fecha de fin**	**Carga de trabajo**
1	lun 07/03/22	lun 21/03/22	40 horas
2	lun 21/03/22	lun 04/04/22	40 horas
3	lun 04/04/22	lun 18/04/22	40 horas
4	lun 18/04/22	lun 02/05/22	40 horas
5	lun 02/05/22	lun 16/05/22	40 horas
6	lun 16/05/22	lun 23/05/22	40 horas
7	lun 23/05/22	lun 06/06/22	40 horas
8	lun 06/05/22	lun 13/06/22	20 horas
Total	8 sprints		300 horas

Tabla 2.1: Planificación de los Sprint del proyecto

Una jornada de trabajo se ha acordado que tiene la duración de 4 horas totales. Se ha decidido que el último Sprint sea más corto para ajustarse a las horas totales que se deben emplear en este proyecto, al ser dedicado este Sprint a documentación del proyecto y no considerarse crítico, se procede de esta manera.

2.3.2. Product Backlog - User Stories

Como ya hemos explicado, las Historias de usuario suelen ser tomadas como los requerimientos y funcionalidades principales del sistema. Tras una primera reunión con el cliente se llegó a la conclusión de que las Historias de usuario mostradas en la Tabla 2.2 son representativas del comportamiento final que se desea en el sistema.

ID	Nombre	Descripción
1	Registro	Como usuario quiero poder estar registrado en la tienda.
2	Inicio sesión	Como usuario quiero poder iniciar sesión en la tienda.
3	Cambiar contraseña	Como usuario quiero poder cambiar mi contraseña de acceso.
4	Buscar productos	Como usuario quiero poder buscar productos en la tienda para poder añadirlos a mi carro de la compra.
5	Buscar productos	Como usuario quiero poder buscar productos mediante un buscador.
6	Cesta de la compra	Como usuario quiero poder añadir los productos que voy a comprar a una cesta de la compra.
7	Realizar Pedido	Como usuario quiero poder realizar un pedido y pagarlo de forma remota.
8	Consultar Pedidos	Como usuario quiero poder ver todos los pedidos que he realizado con anterioridad.
9	Visualizar detalle de pedido	Como usuario quiero poder visualizar pedidos realizados con anterioridad.
11	Modificar direcciones	Como usuario quiero poder modificar mis direcciones de envío y facturación.
12	Modificar datos personales	Como usuario quiero poder modificar los datos personales de mi cuenta.
13	Gestionar métodos de pago	Como usuario quiero poder añadir, eliminar o modificar métodos de pago para mis pedidos.
14	Cancelar pedido	Como usuario quiero poder cancelar pedidos.
15	Recuperar contraseña	Como usuario quiero poder recuperar la contraseña si no la recuerdo.
16	Consultar pedidos de clientes	Como usuario administrador quiero poder consultar los pedidos que han realizado los clientes.
17	Cambiar estado de pedido	Como usuario administrador quiero poder modificar el estado de un pedido de un cliente.
18	Modificar catálogo	Como usuario administrador quiero poder añadir, eliminar o modificar productos del catálogo mediante una interfaz simple.
19	Stock producto	Como usuario administrador quiero poder llevar un control del stock de los productos, añadir o eliminar unidades.
Total	19 User Story	

Tabla 2.2: User Stories Iniciales

2.3.3. Tareas realizadas

Al emplear Scrum, las tareas de cada sprint son planificadas al inicio del mismo. Por ello las tareas realizadas se especifican en el apartado 6.1

Capítulo 3

Análisis de Tecnologías para el desarrollo web

Este capítulo se centrará en el estudio de las distintas alternativas en cuanto a frameworks de desarrollo de aplicaciones web con front-end y back-end. También hablará sobre las distintas alternativas que existen para la conexión de nuestra aplicación con métodos de pago online. Por último, se hablará sobre las tecnologías seleccionadas para el desarrollo de este proyecto y el porqué han sido elegidas.

3.1. Front-end

Un front-end es una parte de un sistema de información al que el usuario final accede directamente en interactúa para recibir o utilizar las capacidades back-end del sistema anfitrión. Permite a los usuarios acceder y solicitar las prestaciones y servicios del sistema de información subyacente o back-end. [30] Es la parte visible para el usuario final, la cuál muestra el diseño, los contenidos y la que permite a los visitantes navegar por las diferentes páginas mientras lo deseen. [37]

En este análisis de tecnologías front-end se han tenido en consideración los framework que a día de hoy son considerados los mejores y más usados por la comunidad de desarrolladores. [46] [56]

3.1.1. Angular

Desarrollado por Google y lanzado en el año 2016 [46], Angular se creó para cerrar la brecha entre las demandas de tecnología de la época y los conceptos tradicionales que tan bien habían funcionado hasta el momento. Su lenguaje de programación es TypeScript, el cual podríamos decir que es JavaScript pero con un sistema de tipado. A día de hoy, es una de las opciones más usadas en el desarrollo web debido a que es soportado por Google.

Su estructura básica se basa en un fichero HTML5 en para la parte de vista y un fichero en lenguaje TypeScript para la parte de la lógica. A diferencia de otros framework, la parte de HTML y la parte de TypeScript han de estar estrictamente separadas. Con esto conseguimos una mayor limpieza en el código.

Algunas de las principales características de Angular son:

- Cuenta con enlace de datos bidireccional. Esto permite que cualquier propiedad o input de nuestro HTML pueda ser modificada en tiempo real tanto por usuario como por aplicación.

- Cuenta en su núcleo con el patrón de inyección de dependencias, lo que nos permite, por ejemplo, inyectar servicios externos a nuestras clases. De esta forma podemos asegurarnos de que para todas las clases necesitadas de ese servicio, solo dispongamos de una instancia compartida de el servicio que inyectamos.

- Cuenta con un sistema de *routing*, que no es más que un sistema de gestión de rutas dentro de core de este frontend. Esto nos permite realizar una gestión de rutas gestionada por la aplicación web, a diferencia de la habitual gestión de rutas a nivel de servidor, quitando así carga a nuestro backend.

- Las versiones actuales de Angular utilizan TypeScript, el cuál es un superset de JavaScript que incluye nuevas y diversas características como el tipado, los operadores lambda e iteradores.

- Introduce el CLI como herramienta de trabajo. Con el proyecto se introducen una suite de herramientas para consola de comandos que permite generar nuevos componentes de nuestro proyecto o nuevos proyectos. También permite generar de forma simple servicios o módulos de nuestro proyecto.

- A diferencia de la primera versión de Angular, llamada AngularJS, las versiones actuales desechan la noción clásica de controlador y alcance y se para a una orientación basada en jerarquía de componentes como concepto de arquitectura.

Angular es un muy potente framework de desarrollo web y posee una gran ventaja el cuál es el soporte a largo plazo de Google, lo que nos asegura que la tecnología se va a seguir actualizando y aparecerán nuevas características para poder introducir en nuestro proyecto.

Posee una curva de aprendizaje elevada, dado son necesarios conocimientos de TypeScript previos y hay que aprender a emplear la sintaxis de Angular para poder trabajar con él.En mi caso personal ya he trabajado con Angular anteriormente, por lo tanto la curva de aprendizaje se ve reducida.

3.1.2. Vue.js

Fue lanzado en 2014, lo que lo convierte en el más joven de los frameworks investigados. Es un framework para desarrollo de frontend de código abierto y basado en HTML + JavaScript. Los frontend que pueden ser creados con este marco son del tipo una sola página.

La principal ventaja que presenta Vue.js con respecto a otros marcos de desarrollo es la facilidad de aprendizaje, ya que solo es necesario saber los básicos de HTML, CSS y JavaScript para poder empezar a hacer cosas con el.

También podemos destacar las siguientes características de Vue.js:

- Es un framework modularizado que destaca por lo bien desacopladas que están las partes que lo componen.

- Es muy ligero en cuanto a descarga del sistema se refiere, lo que lo hace muy rápido y flexible, siendo el más rápido al renderizar de los framworks considerados. [35]

- Es progresivo, lo que significa que es posible aumentar su funcionalidad añadiendo módulos o bibliotecas adicionales. Esto hace que proyectos más simples sean más livianos, al no cargar con funcionalidades no necesitadas.

- Los componentes en Vue.js se programan en ficheros independientes, separando en partes dentro de ellos HTML, CSS y JavaScript.
- Da una amplia libertad en cuanto a tecnologías que pueden ser usadas junto a el, pudiendo integrarlo por ejemplo con React.
- Posee una amplia documentación y es mantenido de forma constante por el equipo creador del marco.

En resumen, Vue destaca por la facilidad de aprendizaje, su rapidez y rendimiento.

3.1.3. ReactJS

React no es un framework de trabajo, es más bien una biblioteca para la creación de interfaces web. Es una librería de gestión del renderizado. Fue creado por ingenieros de Facebook y actualmente es mantenido por estos y por una amplia comunidad de aportadores dado que el proyecto es de código abierto.

Algunas de las principales características y ventajas que lo definen son:

- Solo se encarga del renderizado, lo cuál implica que hay que añadir bibliotecas adicionales del lado del cliente para funcionalidad como *routing*.
- Permite crear interfaces de usuario de modo declarativo y simple.
- Al ser una biblioteca puede ser integrado de forma muy sencilla en casi cualquier proyecto web.
- Se basa en la filosofía de programación funcional. Nociones como funciones puras o funciones de orden superior son aplicados a React y a sus componentes. Podemos encontrar componentes en React que se rendericen únicamente por sus propiedades o componentes que reciben otros componentes para ser creados.
- Posee una alta curva de aprendizaje, ya que posee una sintaxis propia y una forma propia de estructurar los contenidos.
- Es compatible con posicionamiento SEO, característica a destacar dado que esto este proyecto busca crear una tienda online de venta de productos.
- Facilita el diseño responsivo, dado que sus componentes pueden ser utilizados tanto para navegadores web como navegadores móvil, reduciendo así el número de componentes a codificar para la aplicación.
- Al ser una biblioteca, no impone ningún tipo de arquitectura, plantilla o patrón.

React se presenta como una muy potente biblioteca de funciones para los proyectos de frontend web, pero su alta curva de aprendizaje hacen que empezar a crear con él sea algo complejo.

3.1.4. Ember.js

Es un framework de código abierto creado en 2011, el cuál se basa en un arquitectura MVVC (Modelo-Vista-Vista-Controlador) y está considerado uno de los framework más populares para el desarrollo de aplicaciones complejas de varias páginas.

Las principales características de este framework son:

- Posee interfaz de línea de comandos (CLI) al igual que Angular, lo que facilita la creación de rutas, plantillas, servicios, componentes...
- Aboga por la utilización de una arquitectura estricta, lo que puede ser limitante a la hora de desarrollar, pero lo hace muy comprensible para cualquiera que emplee el código creado por otra persona.
- Al emplear arquitectura estricta los desarrolladores tienen seguridad sobre como se utiliza su software y pueden asegurar la compatibilidad con nuevas versiones así como pueden asegurar el mantenimiento a largo plazo de aplicaciones a gran escala.
- Posee enlace de datos bidireccional al igual que Angular.
- Tiene una herramienta de depuración propia muy potente llamada Ember Inspector.
- Posee un alto grado de estabilidad debido a su arquitectura estricta.
- La documentación disponible de Ember es muy comprensible y extensa.
- Proporciona una API que permite el uso de funciones complejas de forma simple.

3.1.5. Tecnología seleccionada para Front-end

Tras explorar las principales opciones de tecnologías para el desarrollo del frontend del proyecto, nos hemos decantado por Angular como tecnología a utilizar para el proyecto.

La principal razón de escoger Angular por encima de otros framework es que ya se había utilizado con anterioridad, por lo tanto el aprendizaje para la realización del proyecto será mínimo y en poco tiempo se podrá estar trabajando de forma fluida con él. También es muy importante para mí el enlace de datos bidireccional, ya que de esa manera se pueden ver los cambios en tiempo real e ir corrigiendo los errores que puedan surgir durante el desarrollo.

3.2. Back-end

El back-end es la parte no visible por el usuario final de nuestra aplicación. Es la que se encarga de la conexión con la base de datos y el posterior envío de información procesada al front-end para que esta pueda ser mostrada al usuario. Es muy importante que las aplicaciones web posean un back-end bien construido ya que es clave para su seguridad.

Un back-end puede ser creado a parte y de forma manual para instalarlo y mantenerlo de forma manual en nuestro servidor, o se puede optar por la opción de contratar un BaaS (BackEnd As A Service), que es una plataforma que automatiza el desarrollo del backend y se encarga de proporcionarle a este una infraestructura en la nube. [33] También se ha optado por investigar esta opción debido a que se pretende realizar la conexión de nuestro backend con servicios de pago online, y estos servicios exigen un alto grado de seguridad para su utilización, seguridad que nos proporcionan los BaaS al estar muy mantenidos por sus respectivos propietarios.

Otra gran ventaja que presentan los servicios BaaS es que la configuración de seguridad respecto al Reglamento General de Protección de Datos se hace de forma automatizada, lo que permite trabajar de

manera senciall de acuerdo a la legislación vigente.

Para este proyecto se ha decidido investigar la solución de BaaS, ya que es más rápido su desarrollo y mantenimiento, dado que gran parte de las tareas están automatizadas en estos servicios y su conexión con los principales framework de frontend es bastante sencilla.

3.2.1. Back4App

Back4App [31] un producto para la creación de BaaS de manera rápida y sencilla. Ofrece a los desarrolladores una gran cantidad de herramientas para gestionar la implementación y la gestión de sus aplicaciones. Es de código abierto, lo que permite a los desarrolladores saber como funciona el servicio.

Las principales características y herramientas que posee este servicio son: [31]

- Autenticación de usuarios.
- Base de relacional en tiempo real NoSQL o SQL.
- Notificaciones en tiempo real.
- APIs GraphQL y REST auto-generadas a partir de esquemas de base de datos.
- *Hosting* de archivos en la nube.
- Funciones de Cloud Code para lógica empresarial.
- Escalado automático de la aplicación acorde al numero de peticiones recibidas.

Back4App posee un plan gratuito limitado para el desarrollo de la aplicación, lo cual es muy interesante en el proyecto actual.

3.2.2. Firebase

Firebase [41] es un servicio de BaaS proporcionado por Google. Firebase es en esencia un gran conjunto de herramientas proporcionadas por el servicio que los desarrolladores pueden utilizar para crear sus aplicaciones. Sus herramientas se pueden agrupar en tres grandes sub-grupos, desarrollo, calidad de la aplicación y crecimiento de la aplicación. Aquí solo se han tenido en cuenta las herramientas y características de desarrrollo, de las cuales cabe destacar:

- Autenticación de usuarios.
- Integración con proveedores de identificación federada.
- Almacenamiento de datos en la nube.
- Base de datos en tiempo real.
- Funciones en la nube.
- Sincronización de datos en varios dispositivos.
- Aprendizaje automático gracias a procesos de Machine Learning.

- Hosting gratuito hasta cierto nivel de carga.

El servicio de Firebase cuenta con una versión de prueba con bastantes funcionalidades y libertad de uso. Firebase posee además una serie de funciones que permiten conectarlo de forma simple con Stripe, una plataforma de pagos para Internet, que permite procesar pagos sin la necesidad de desplegar una infraestructura para ello.

3.2.3. AWS Amplify

Amplify [53] es la propuesta de servicio de BaaS de Amazon Web Services. Es un conjunto de herramientas y características creadas específicamente para que los desarrolladores de frontend puedan crear un servicio de backend de manera rápida y sencilla.

Posee numerosas herramientas y características como:

- Autenticación de usuarios.
- Almacenamiento en la nube.
- Granjas de dispositivos.
- Funciones de creación automatizada de APIs.

Ofrece un servicio gratuito de un año de duración, y el modelo de facturación que sigue es el de pagar por lo que usas.

3.2.4. MongoDB Realm

Realm [51] es la solución de BaaS propuesta por MongoDB. Los servicios de Realm reducen la cantidad de código requerida por los desarrolladores para poner en marcha un proyecto.

Entre sus principales características se encuentran:

- GraphQL dentro del servicio.
- Posibilidad de crear funciones propias en JavaScript dentro del Realm.
- Triggers de base de datos simples de usar.
- Manejo de permisos de acceso a datos de manera simple.
- Autenticación de usuarios.
- Para poder usar el servicio de base de datos es necesario configurarlo a parte.

Aunque MongoDB sea el más nuevo de todos los servicios BaaS posee grandes características, pero le falta bastante automatización de procesos, lo cuales deberían ser implementados a mano.

3.2.5. Servicio Backend seleccionado para el proyecto

Para este proyecto se ha decidido utilizar Firebase. Principalmente se ha tomado esta decisión por la gran cantidad de documentación que posee el servicio, los numerosos ejemplos creados por la comunidad de Firebase sobre como integrarlo en una aplicación y porque está respaldado por Google, lo que implica que será mantenido de forma activa por esta empresa durante muchos años.

Se elije también dado que posee una integración muy simple con "'Stripe", una plataforma de pagos on-line.

Otra característica importante que posee Firebase es que permite autenticación de usuarios con cuenta de Google, un aspecto que puede resultar interesante debido a que agilizaría el proceso de registro de nuevos usuarios en nuestra aplicación.

También proporciona un Hosting muy rápido y gratuito hasta que sobrepasamos un límite de carga en nuestra aplicación.

Firebase cuenta además con un paquete de Node.js, llamado "AngularFire " [5], creado y mantenido por desarrolladores de Google, que permite integrarlo de forma muy simple en aplicaciones web desarrolladas con Angular.

Capítulo 4

Tecnologías utilizadas

4.1. HTML5

HTML5 [16] es el acrónimo de "HyperText Markup Lenguage 5ƶ es la quinta revisión importante de el lenguaje que se emplea en la World Wide Web, HTML. HTML es un lenguaje de marcado que se emplea en la elaboración de páginas web. su desarrollo y mantenimiento está a cargo del "W3C, o World Wide Web Consortium", una organización dedicada a la estandarización de las tecnologías empleadas en la web.

HTML [15] es un lenguaje que basa su filosofía en la diferenciación. Para añadir un elemento externo a nuestra página web lo hacemos mediante una referencia en texto plano a la ubicación de ese archivo. De este modo nuestra web contiene únicamente texto plano y nuestro navegador web es el encargado de interpretar ese texto y unir todos los elementos para visualizar en la página final ya compuesta. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

4.2. Git

Git [14] es un sistema de control de versiones gratuitamente distribuido y de código abierto diseñado para trabajar con el en el desarrollo de pequeños proyectos o proyectos a gran escala. Fue diseñada en 2005 por el creador del kernel de linux, Linus Torvarlds, para la realización de esta tarea junto con otros desarrolladores de forma coordinada.

En ingeniería del software se entiende por sistema de control de versiones a un sistema que se encarga de guardar los cambios realizados sobre un archivo o grupo de archivos realizados a lo largo del tiempo para así poder volver a versiones especificas anteriores en un futuro. [36]

Git es un sistema de control de versiones distribuido, los cuales se basan en que todos los aportadores poseen una copia total del repositorio, en el que se guardan los archivos, las versiones y los cambios realizados a el proyecto. Esto significa que todos los desarrolladores poseen en todo momento un clon del repositorio de desarrollos, la cual puede ser usada en un moemento de emergencia como un recurso de respaldo si el servidor central con el repositorio se ve afectado por algún problema. Este tipo de sistemas de control de versiones permiten trabajar de forma muy satisfactoria a varias personas en un mismo proyecto, lo que permite crear varios tipos de flujos de trabajo que serían imposibles de llevar cabo en un sistema de versiones centralizado.

Git además posee un motor de integridad. Todo en git pasa por una función de Checksum antes de ser guardado, y los datos a ser guardados son referenciados por ese valor calculado. Esto significa, que no es posible cambiar los contenidos en un repositorio sin que Git tenga constancia de ello. Esto implica que no se puede perder información o corromper archivos sin que Git lo detecte.

Para este proyecto se ha utilizado la instancia de GitLab proporcionada por la Escuela Técnica superior de Informática de la Universidad de Valladolid, alojada en "`gitlab.inf.uva.es`çomo repositorio remoto para el desarrollo de la aplicación.

4.3. CSS3

CSS3 es ultima gran revisión del lenguaje de diseño gráfico CSS. CSS es un lenguaje de diseño gráfico que permite definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web e interfaces de usuario escritas en HTML. [50]

Las siglas de CSS son el acrónimo de "Cascading Style Sheets"(Hojas de estilo en cascada). El significado de este nombre se entiende diferenciando a qué se refiere cada uno de los términos que lo componen:

- **Cascading**, significa que los estilos que aplicamos a un elemento de una web HTML son propagados a los elementos que este contiene, se propagan en cascada.
- **Style**, indica que lo que creamos con este lenguaje son estilos que se aplican a distintos elementos de nuestra web.
- **Sheets**, que en inglés significa "hojasïndica que los estilos declarados para que se apliquen a nuestra web son declarados en ficheros a parte del HTML de nuestra web, ficheros que, normalmente, poseen extensión "*.css*".

Desde su origen CSS ha estado muy ligado a HTML, ya que el objetivo de CSS al nacer fue poner orden a la hora de aplicar los estilos a las paginas web.

4.4. TypeScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. [25]

TypeScript soporta ficheros de de definición de estructuras de datos y tipos, los cuales pueden ser utilizados por otras clases mediante importación.

TypeScript es una extensión del lenguaje ECMAScript 6 que añade las siguientes características a este:

- Anotaciones de tipo y comprobación de tipos en tiempo de compilación.
- Inferencia de tipos.

- Borrado de tipos.
- Interfaces.
- Tipos enumerados.
- Genéricos.
- Espacio de tipos.
- Tuplas.
- Async/await.

4.4.1. Compatibilidad con JavaScript

Al ser TypeScript un superset de ECMAScript6 un programa escrito en JavaScript es un programa TypeScript válido, por lo tanto se puede emplear código escrito en JS en programas TS.

4.5. Angular

Angular [27] es una plataforma de desarrollo construida en torno a TypeScript. Como plataforma, Angular incluye:

- Un marco de trabajo basado en componentes para la construcción de aplicaciones web fácilmente escalables.
- Una colección de librerías que cubren una amplia variedad de funcionalidades, incluyendo gestión de rutas, manejo de formularios, comunicación cliente-servidor y muchas más.
- Un abanico de herramientas de desarrollo para ayudar en esta tarea a los desarrolladores, con herramientas para construir, probar y actualizar el código.

Los elementos principales de la arquitectura de Angular (Figura 4.1 sonson los componentes, módulos y servicios:

- **Módulos:** un módulo es una clase TypeScript con un decorador `@NgModule` al principio de esta que contiene los metadatos que identifican al módulo. Los módulos de Angular pueden contener Componentes, Servicios, otros módulos y otros archivos cuyo alcance esté definido por el módulo que los contiene.
- **Componentes:** Un componente se compone de un fichero HTML con la estructura de los elementos, una hoja de estilos en CSS y un fichero TypeScript que contiene la defienición del componente y los metadatos que identifican al componente. La clase definida dentro del fichero de TypeScript interactúa con la vista HTML mediante una API de propiedades y métodos.
- **Servicios:** un componente se centra en la experiencia de usuario y en nada más técnicamente, es por ello que para otras tareas como la carga de datos, la comunicación con el servidor, la comunicación entre componentes, se emplean los servicios de Angular. Los servicios son clases que van precedidas del decorador `@Injectable`, el cuál indica a Angular que la clase escrita a continuación puede ser inyectada a otras clases como dependencia. Angular distingue componentes de servicios para incrementar la modularidad y la reusabilidad.

- **Guards**: Un *guard* es un tipo especial se servicio que sirve para activar una serie concreta de rutas de la aplicación. Son inyectados a través del objeto de composición del módulo de *routing* y poseen dentro lógica que permite acceder a una ruta en concreto de la aplicación en función de si se cumple una condición o no. Permiten redirigir al usuario a otra página en caso de que no se permita el acceso. Por ejemplo, se podría implementar un *guard* que restrinja el acceso a las páginas de el perfil de un usuario si no hay ningún usuario con sesión iniciada.

- **Directivas:** aunque no son parte de los 3 elementos principales de Angular, las directivas son una parte muy importante del funcionamiento de este framework. Las directivas aplican una transformación sobre un elemento del template, por ejemplo, mostrarlo un numero x de veces, mostrarlo de forma condicional si se cumple una condición...

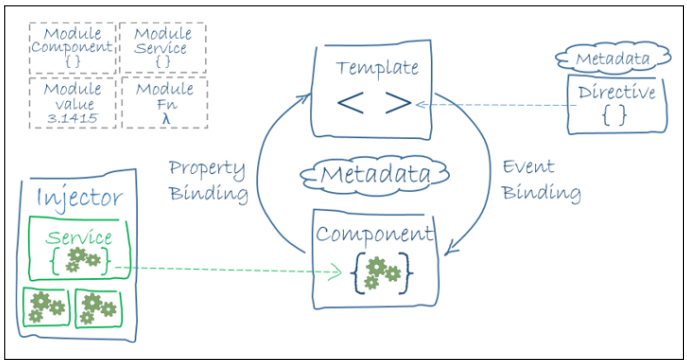


Figura 4.1: Relación entre elementos de Angular, Arquitectura Angular [17]

4.6. Firebase

Firebase es un servicio de Backend as a Service proporcionado por Google. [41] Propociona herramientas para el desarrollo de aplicaciones de alta calidad. De todas las herramientas proporcionadas por Firebase para este proyecto se han usado las siguientes:

4.6.1. Firebase Authentication

La autenticación es una necesidad indispensable de nuestra aplicación a desarrollar en este proyecto, es por ello que se hace necesaria la creación o el uso de un servicio de autenticación de usuarios.
Firebase Authentication [10] proporciona los servicios de back-end necesarios para autenticar usuarios en nuestra aplicación. Admite la autenticación mediante contraseñas, numeros de teléfono, o proveedores de identidad federada populares como Google, Facebook o Twitter entre otros muchos.

4.6.2. Storage

Cloud Storage [11] es un servicio de almacenamiento de objetos potente creado para el escalamiento de Google. Se pueden almacenar grandes objetos y organizarlos a nuestro gusto en una estructura simple de carpetas que pueden ser accedidos más tarde mediante una API que proporciona referencias remotas a los objetos proporcionada por Firebase. Los objetos que introducimos pueden ser direcamente referenciados mediante URI en nuestra base de datos de Firebase.

4.6.3. Firestore Database

Firestore Database [12] es una base de datos NoSQL alojada en la nube y que, por tanto, sigue un modelo de base de datos no relacional. En Firestore los datos se almacenan dentro de documentos, los cuales están agrupados dentro de colecciones de documentos. Los documentos contienen campos que se asignan a valores. Los documentos admiten varios tipos de datos, desde strings y números simples, hasta BLOBs y mapas de datos. Dentro de un documento se pueden crear sub-colecciones para crear así una estructura de datos jerárquica.

Al ser una herramienta de Firebase, Firestore es compatible con Firebase Authentication para la protección de los datos de los usuarios mediante reglas de seguridad que se evalúan en Firebase antes de cada consulta a la base de datos.

4.6.4. Hosting

Firebase Hosting [13] es un servicio de hosting de contenido web con nivel de producción orientado a desarrolladores. Permite entregar contenido a nivel global mediante una red de distribución de contenidos. Incluye SSL para que los contenidos se entreguen de forma segura sin la necesidad de configurar este servicio, lo realiza de forma automática. Permite implementar nuevas versiones de forma simple, instalando el `Firebase CLI` puedes publicar nuevas versiones con un solo comando de terminal.

4.7. npm

npm o "Node Package Manager" [2] es el gestor de paquetes de Node.js. Fue creado en 2009 como un proyecto de código abierto para ayudar a los desarrolladores de JavaScript a compartir paquetes entre ellos de forma simple. El registro de paquetes que maneja está considerado uno de los más grandes del mundo. [**?**]. El Registro de npm es una colección de paquetes de codigo abierto para ser empleados en Node.js, proyectos de front-end, back-end, y otros muchos usos que dependan de JavaScript. npm es también la línea de comandos que permite a los desarrolladores compartir y buscar estos parquetes en el Registro.

A continuación se detallan los paquetes npm utilizados:

4.7.1. @angular/cli

Angular CLI [8] es la interfaz de línea de comandos del framework Angular. Sirve para crear proyectos angular, generar nuevos componentes, nuevos servicios, compilar y desplegar aplicaciones Angular en local...

4.7.2. firebase

El paquete npm firebase contiene las herramientas e infraestructura necesaria para desarrollar aplicaciones que utilicen los servicios de Firebase. La herramienta más utilizada de este paquete ha sido "`deploy`"la cuál permite desplegar nuestra web-app con un solo comando en un dominio Firebase, el cuál es accesible desde cualquier parte del mundo.

4.7.3. @angular/fire

Angular Fire [4] es la librería oficial de angular para incluir Firebase en un proyecto. Fue creada y es mantenida por empleados de Google, auqneu no es un producto de Google como tal.

Contiene las herramientas necesarias para trabajar con Firebase en Angular de una forma sencilla y escribiendo código TypeScript, ya que las librerías de Firebase vienen escritas en JavaScript.

Basa sus operaciones en Observables, lo cuál hace que se pueda trabajar de forma simple con operaciones asíncronas.

4.7.4. bootstrap

Bootstrap [28] es un framework basado en HTML, CSS y JavaScript para desarrollar aplicaciones web responsivas. Una aplicación responsiva [29] es aquella que adapta su contenido para verse de forma correcta dependiendo del tipo y tamaño del dispositivo en el que sea mostrada.

Bootstrap provee al desarrollador con inumerables clases CSS que sirve para adaptar el contenido de su web de forma rápida y sencilla.

Posee también elementos dinámicos y animaciones para ellos, las cuales viene pre-programadas y no es necesario implementarlas.

4.7.5. jQuery

jQuery [18] es una librería pequeña pero potente de JavaScript. Se utiliza para la manipulación de documentos HTML de forma transversal, manejar eventos de navegador, crear y manejar animaciones y mucho más por medio de una API que funciona en multitud de navegadores modernos.

Esta librería es necesaria porque Bootstrap hace uso de ella para algunos aspectos como gestionar algunas animaciones.

4.7.6. rxjs

rxjs [24] es un conjunto de librerías empleadas incorporar funcionalidad asíncrona y basada en eventos empleando colecciones de Observables (objetos que siguen un patrón observador como forma de funcionamiento).

En angular se emplea normalmente para la comunicación mediante Observables entre componentes y servicios, a parte de para saber cuando tenemos disponible un dato cargado de nuestra base de datos, o cuando hemos terminado su carga o actualización.

4.7.7. primeng

PRIMENG [23] es un conjunto de componentes de interfaz Angular de código abierto. Proporciona múltiples componentes con animaciones y carga de datos dinámica escritos para poder trabajar con

Angular de forma directa y simple.

Para este proyecto se ha empleado principalmente para la inclusión de componentes con animaciones en nuestra aplicación como por ejemplo algunos carouseles de productos o imagenes.

4.7.8. primeicons

Los componentes de `primeng` utilizan de forma interna los iconos proporcionados por `primeicons`, la cuál es una librería de iconos para poder incluir dentro de nuestro template HTML.

A parte de para los componentes de `primeng` se utiliza en este proyecto para poder incorporar iconos en nuestra aplicación, lo cuál la hace más vistosa para los usuarios.

4.7.9. ngbootstrap

NgBootrstrap [20] es una librería que incorporpora a Angular algunos widgets programados para funcionar directamente con Angular. Se vio que poseia algunos widgets interesantes, como los carouseles de imagenes, y se decide incorporarlo a este proyecto.

4.7.10. Overleaf

Overleaf [21] es un editor de LaTeX en la nube el cual permite escribir grandes documentos dándoles forma importando paquetes y librerías de LaTeX de manera automática y compilarlos en la nave, para después poder descargarlos en formato pdf sin necesidad de hacer nada en local. Incorpora dentro de sí git para controlar las versiones de los documentos, pudiendo volver a versiones anteriores sin perder datos.

4.8. Visual Studio Code

Visual Studio Code [26] es un editor de texto ligero pero potente que permite aumentar sus capabilidades por medio de extensiones que la comunidad publica en él. Es desarrollado por Microsoft, los cuales también son creadores y mantenedores de TypeScript, es por esto que Visual Studio es una de los editores de código más populares entre los desarrolladores de webs basadas en TypeScript. Posee linea de comandos integrada, la cual es muy útil durante el desarrollo al poder ver los errores de las compilaciones de forma directa, sin cambiar de ventana.

4.9. Diagrams.net

Diagrams.net [1] es una tecnología de código abierto para la creación de diagramas. Es el software de creación de diagramas basado en navegador más usado del mundo. En él se pueden crear múltiples tipos de diagramas y posee formas ya creadas que son re-escalables y conectables entre sí. Se tomó como opción para la creación de los diagramas UML, pero su funcionalidad era escasa para este cometido. Sí fue empleado en la realización de otros diagramas, como el expuesto en 7.6.2.

4.10. Astah Profesional

Astah Profesional [6] es un software de edición de diagramas. Permite en una sola herramienta crear múltiples tipos de diagramas UML, entidad-relación, diagramas de flujo... Con Astah Profesional se puede crear código a partir de las definiciones creadas en los diagramas, aunque esta funcionalidad no es soportada para todos los lenguajes.

En este proyecto se emplea para la creación de los diagramas UML dado que Diagrams.net, explicado en 4.9, no poseía la funcionalidad necesaria para crear los diagramas UML necesarios para este proyecto de una forma rápida.

4.11. WireFrame Pro

Wireframe Pro [19] es un software en la nube de diseño de interfaces web creado por la empresa MockFlow. Permite el diseño de interfaces web por medio de componentes pre-creados los cuales van desde una representación de los elementos finales a componentes dibujados en forma de bocetos a mano alzada. También posee una tienda de compra de diseños creados por la comunidad. En este proyecto ha sido empleado para la creación de los bocetos de las vistas de la aplicación Angular.

Capítulo 5

Plan de riesgos y presupuestos del proyecto

5.1. Plan de riesgos

Los proyectos de desarrollo software, cómo cualquier otro proyecto de desarrollo pueden verse expuestos a riesgos durante su desarrollo que alteren los plazos de entrega del proyecto en el caso de que estos existan, o deriven en casuísticas y problemáticas mayores que la del propio riesgo.
En este capitulo se expondrá una definición de riesgo aplicado a proyectos de desarrollo y ser desarrollará un plan de riesgos y acciones relacionadas a estos para minimizar su impacto.

5.1.1. Definición de riesgo

Un riesgo, cuando nos referimos a ello en el ámbito de un proyecto software, se define como "algo que puede convertirse o bien en un problema, o en una oportunidad" [22]. Otra definición puede ser "un evento o condición incierta que, si ocurre, tiene un efecto positivo o negativo en los objetivos de un proyecto" [44].

5.1.2. Gestión de los riesgos

El objetivo de la gestión de riesgos es la de evitar o minimizar las adversidades de eventos imprevistos por medio de evitar esos riesgos o elaborando planes de contingencia para lidiar con ellos [43].

Un desglose de las tareas más comunes a realizar para lidiar con los riesgos de una manera correcta son las siguientes: [43]

- **Identificación de riesgos:** consiste en enumerar y pensar todos los riesgos que pueden aparecer en el transcurso de un proyecto.
- **Estimación de riesgos:** se refiere a evaluar la posibilidad de que el riesgo se materialice y el impacto que puede tener si esto llega a pasar.
- **Análisis de riesgos:** consiste en crear un ranking de riesgos para saber en cuales hay que poner un mayor esfuerzo y cuales merecen menos.
- **Planificación de riesgos:** consiste en crear planes de contingencia, y si es preciso, añadir estas medidas a las tareas del proyecto.

- **Control de riesgos:** se refiere a las funciones principales del gestor de riesgos, minimizar los riesgos y reaccionar a los problemas a lo largo de proyecto.
- **Monitorización de riesgos:** se refiere a la tarea de vigilar si un riesgo se está materializando o se va a materializar y poder detectarlo antes de tiempo.
- **Dirección de riesgo y personal de riesgo:** hace referencia a la gestión diaria de los riesgos. La resolución de problemas y la aversión de riesgos normalmente precisan de personal extra que debe ser planeado y dirigido.

5.1.3. Cómo crear un plan de riesgos

Sabiendo cuales son las principales tareas a llevar a cabo en la gestión de riesgos, podemos acotar cuales son las fases a llevar a cabo para la creación de un plan de riesgos y las partes que debe poseer el mismo:

1 Identificación de los riesgos.

2 Análisis de los riesgos.

3 Crear planes de contingencia para los riesgos.

4 Monitorizar y controlar los riesgos.

Para esta tarea se va a crear un registro de riesgo para cada uno de los riesgos identificados en nuestro proyecto siguiendo el modelo propuesto en "Software Project Management" [43], con una pequeña variación en la sección de probabilidad e impacto de los riesgos, en la cuál se simplificarán los valores asociados reduciéndolos a "Alto", "Medioz "Bajo". La sección de incidentes y acciones será eliminada para simplificar las tablas descriptivas del registro del riesgo.

5.1.4. Plan de riesgos

A continuación se describen los riesgos detectados en este proyecto, por medio de tablas que representan los registros de riesgo.

Riesgo	RG 01
Nombre	Falta cualificación técnica
Probabilidad	Media
Impacto	Medio
Descripción	Puede darse la casuística de no tener conocimientos suficientes en las tecnologías y metodologías a emplear en el desarrollo del proyecto.
Plan de mitigación	Adquirir cualificación previa a el comienzo de la implementación del proyecto.
Plan de contingencia	Consultar o pedir ayuda a alguien que sea experto en la tecnología que crea problemas

Tabla 5.1: RG 01 - Falta de cualificación

Riesgo	RG 02
Nombre	Estimación de tiempos no realista
Probabilidad	Baja
Impacto	Medio
Descripción	Los tiempos estimados para la realización de las tareas pueden no ser realistas y provocar un retraso en el desarrollo del proyecto.
Plan de mitigación	Estimar las tareas con más tiempo del necesario para proporcionar un colchón de tiempo
Plan de contingencia	Introducir más sprints al proyecto para poder completarlo, y dedicar horas extra no planificadas

Tabla 5.2: RG 02 - Mala estimación de tiempos

Riesgo	RG 03
Nombre	Desarrollo de la interfaz de usuario incorrecta
Probabilidad	Baja
Impacto	Medio
Descripción	En el desarrollo del proyecto puede que la interfaz desarrollada no sea apropiada para el proyecto que se está realizando y sea necesaria su modificación a mitad de estes, lo que conllevaría retrasos y un aumento de los costes
Plan de mitigación	Crear bocetos a modo de prototipo y presentárselos al cliente antes de pasar a su implementación
Plan de contingencia	Modificar la interfaz para adaptarse a los deseos del cliente.

Tabla 5.3: RG 03 - Diseño incorrecto de interfaz

Riesgo	RG 04
Nombre	Fallo en los equipos
Probabilidad	Media
Impacto	Media
Descripción	Puede ocurrir que alguno de los equipos de desarrollo, los ordenadores, se vea dañado y no pueda ser utilizado para este cometido.
Plan de mitigación	Poseer un equipo de respaldo por si el ordenador falla
Plan de contingencia	Adquirir un nuevo equipo

Tabla 5.4: RG 04 - Fallo de los equipos

Riesgo	RG 05
Nombre	Caída de los servicios del proveedor de back-end
Probabilidad	Baja
Impacto	Alta
Descripción	Puede darse la casuística de que el servicio de back-end se vea afectado por algún problema y no sea posible trabajar con él
Plan de mitigación	Elegir un servicio que tenga un alto nivel de seguridad y sea distribuido de forma global para contar con respaldos
Plan de contingencia	Migrar todos los servicios implementados en el back-end a otro proveedor de servicios

Tabla 5.5: RG 05 - Caída del proveedor de BaaS

Riesgo	RG 06
Nombre	Cambios tardíos en los requerimientos
Probabilidad	Baja
Impacto	Alto
Descripción	Puede ser que en el transcurso del desarrollo del proyecto el cliente quiera optar por incluir nueva funcionalidad al producto o modificar alguna ya introducida
Plan de mitigación	Realizar varias reuniones previas al desarrollo con el cliente para entender bien los requerimientos que se deben implementar y emplear metodologías ágiles para permitir la inclusión de nuevos requerimientos
Plan de contingencia	Añadir los nuevos requerimientos al proyecto y añadir tareas para poder completarlos

Tabla 5.6: RG 06 - Cambios en los requerimientos

Riesgo	RG 07
Nombre	Enfermedad de algún miembro del equipo
Probabilidad	Baja
Impacto	Alto
Descripción	Teniendo en cuenta la situación de emergencia sanitaria en la que se enmarca este proyecto es posible que el desarrollador caiga enfermo y no pueda llevar a cabo sus tareas.
Plan de mitigación	Planificar las tareas con un colchón para poder realizarlas y presupuestar el proyecto de forma alcista para asumir el coste extra del retraso.
Plan de contingencia	Mover las tareas afectadas por el retraso a otro sprint posterior y añadir horas extra en caso de ser necesario.

Tabla 5.7: RG 07 - Enfermedad de algún miembro

Riesgo	RG 08
Nombre	Cambios en alguna tecnología empleada
Probabilidad	Baja
Impacto	Alto
Descripción	Puede ocurrir que alguna de las tecnologías empleadas cambie su sintaxis o su forma de utilización y sea necesaria la re-escritura del código para adaptarse a los cambios en la tecnología.
Plan de mitigación	Elegir tecnologías que no sean cambiantes, y estén ya muy probadas, para que no puedan aparecer fallos en ellas que hagan cambiar la tecnología a los desarrolladores de esta. Elegir tecnologías en las que las nuevas versiones sean retro-compatibles con las versiones anteriormente publicadas.
Plan de contingencia	Re-escribir el código para adaptarlo a la nueva forma de utilización.

Tabla 5.8: RG 08 - Cambio en alguna tecnología

5.2. Presupuestos

5.2.1. Presupuesto inicial del proyecto

Para calcular estos presupuestos se ha tenido en cuenta que el sueldo publicado en el Boletín oficial del Estado de Martes 6 de Marzo de 2018, Sec. III. Pág. 26978, [38] en el que se describen los salarios y los convenios colectivos de el Área de Consultoría, Desarrollo y Sistemas vigentes desde la publicación de el mismo hasta el día de hoy. En este documento se cifra el sueldo base de un "`Analista desarrollador; Diseñador página web`" en 21.555,66€ y el "plus del convenio"se cifra en 1.408,18€, para hacer un total de 22.993,74€ de sueldo bruto total.

A esta cifra hay que añadirle los costes derivados que tiene la empresa por las cotizaciones a la Seguridad Social por el contrato del trabajador, las cuales son obligatorias y suponen aproximada un 30 % a mayores del sueldo base bruto del trabajador [45]. Lo que contando el 30 % que hay que pagar a la seguridad social nos queda que un "`Analista desarrollador; Diseñador página web`"le cuesta a una empresa 29.891,86€ por año trabajado. Teniendo en cuenta que las horas aproximadas que realiza al año un trabajador son unas 1800h [7] nos queda que el salario bruto por hora de un diseñador y analista de páginas web es de 16,60€ brutos por hora trabajada.

Como ya se ha descrito en la sección 2.3.1 la duración total del proyecto es de 300 horas planificadas inicialmente, con esto podemos calcular el coste de la mano de obra asociada al proyecto. Este proyecto contará únicamente con un trabajador el cuál realizará todas las tareas. Con esto nos queda que el coste de mano de obra se cifra en 4.982€ ; (16,60€/h * 300h)

En cuanto a coste de equipos para el desarrollo no existe ningún gasto de compra de equipos a priori dado que el trabajador posee un equipo sobremesa solvente para la realización del mismo, equipado con procesador Ryzen 5600X, 32Gb de RAM, GPU Nvidia RTX 3070 y 1TB de SSD NVMe, valorado en 2.350€. Aunque no haya que comprar ningún equipo sí es necesario recompensar al trabajador por la amortización del uso de su equipo. Esto se hará acorde a la tabla publicada por la Agencia Estatal de Administración Tributaria, que puede verse resumida en esta web [3]. En esta se cifra que la amortización máxima en un año

de un equipo informático es del 25 %, y que el periodo máximo para su amortización es de 8 años. Con esto nos queda que la amortización que debe percibir el trabajador por el uso de su equipo es de 293,75€ (2.350€ / 8 años), la cuál no sobrepasa el 25 % del coste del equipo y por tanto es lícita su adjudicación al trabajador.

En cuanto a herramientas y software para el desarrollo, los programas que se emplean en la realización del proyecto son completamente gratuitos en su uso, por tanto no suponen un coste para el devenir del proyecto. Firebase, el proveedor elegido como back-end, es gratuito hasta un cierto límite de uso, el cuál se estima que no va a ser sobrepasado en el desarrollo del proyecto, aunque se deja especificado que puede suponer un coste para total transparencia con el cliente.

En cuanto a formación, nuestro equipo de desarrollo no posee la experiencia necesaria para un proyecto de este calado en la tecnología empleada para el Front-End, es por ello que se presupuesta también la formación del mismo, la cuál consistirá en un curso de Udemy.

En cuanto a recursos gráficos para el sitio web, no se encuentra ningún coste asociado dado que se emplearan iconos de la librería de PrimeIcons, expuesta en 4.7.8. Para imágenes se emplearan las aportadas por el cliente, dado que ya las posee en su actual sitio web.

Se decide aumentar el presupuesto inicial en un 25 % para poder tener un fondo de maniobra, que, en caso de ser necesario, será empleado para sufragar los costes derivados de los riesgos que se hayan podido materializar en el transcurso del proyecto.

Habiendo explicado los costes podemos hacer una tabla resumen (Tabla 5.9 que nos aglutine todos estos costes asociados al proyecto:

Coste	**Montante del coste**
Programas y herramientas	0€
Firebase	0€ (puede variar)
Equipo de trabajo	4.982€
Amortización de equipos	293,75€
Imágenes y grafismos	0 €
Formación Angular	13.5 €
Total costes proyecto	5289,25€
Total + Fondo maniobra 25 %	**6611,50€**

Tabla 5.9: Presupuesto inicial de costes del proyecto

5.2.2. Presupuesto real del proyecto

Tras la realización del proyecto, los costes reales de la realización del mismo son los expuestos en la Tabla 5.10.

Coste	Montante del coste
Programas y herramientas	0€
Firebase	0€ (puede variar)
Equipo de trabajo	(304h * 16,60€/h) -¿5.046,4€
Amortización de equipos	293,75€
Imágenes y grafismos	0 €
Formación Angular	13.5 €
Total costes proyecto	**5.353,65€**
Variación final-inicial	**+1.257,85€**

Tabla 5.10: Presupuesto final de costes del proyecto

Al no haber habido una gran desviación de las horas planificadas en el proyecto y las horas empleadas en la realización del mismo, se ha conseguido completar el proyecto con un coste menor al planificado inicialmente, gracias a el fondo maniobra que se presupuestó.

Concretamente se han invertido en la realización del proyecto 64,4€ tomados del fondo de maniobra. Estos costes son asociados a horas de trabajo extra realizadas por el equipo de desarrollo.

Sabiendo esto, el coste final se ha visto reducido en 1.257,85€ frente a los costes presupuestados inicialmente (Tabla 5.10).

Capítulo 6

Desarrollo del proyecto

Durante el desarrollo se han ido realizando diversas tareas por sprint, las cuales han sido planificadas al principio de cada uno tal y como marca el marco de Scrum. En estas tareas se han incluido tareas de diseño, desarrollo, las reuniones con el cliente, la documentación, etc.

Cada tarea posee un tipo los cuales pueden ser :

- **DES**: reciben este tipo las tareas relacionadas con la parte de desarrollo de alguna funcionalidad del proyecto. También se incluye en este tipo la formación y la investigación, ya que son tareas que afectan al desarrollo.
- **DOC**: reciben este tipo las tareas que tienen relación con la labor de documentar.
- **TEST**: reciben este tipo las tareas que tienen relación con las pruebas realizadas en el proyecto.
- **BUG**: reciben este tipo las tareas que tienen relación con la solución de algún error detectado en la aplicación.
- **REU**: reciben este tipo las reuniones mantenidas con el cliente.

Las tareas tienen asociado un estado, el cuál indica como se encuentra la tarea en ese momento, los estados que pueden tener las tareas son:

- **En curso**: indica que la tarea al final del sprint no ha sido completada, pero que se ha dedicado alguna hora de trabajo a ella.
- **No iniciada**: indica que la tarea ha sido planificada para el sprint en el que está incluida pero que, al final del mismo no han comenzado los trabajos asociados a ella, y por tanto no se han invertido horas en ella.
- **Completada**: indica que las tareas planificadas para la tarea han terminado de forma satisfactoria.

En cada una de las tareas se especifica el tiempo invertido en esta tarea.

6.1. Desarrollo del proyecto sprint a sprint

6.1.1. Sprint 1

Este es el Sprint inicial del proyecto, en el ha tenido lugar la reunión con el cliente para entender sus deseos en lo que respecta al producto resultante. A partir de esa reunión se han planificado una serie de

tareas para las dos semanas siguientes, las cuales se basan principalmente en planificar el proyecto.

Tarea	**Tipo**	**Descripción**	**Tiempo invertido**	**Estado**
T001	REU	Reunión inicial del proyecto para concretar los deseos del cliente	3 horas	Completada
T002	DES DOC	Redacción de los requisitos funcionales	3 horas	Completada
T003	DOC	Crear proyecto en Overleaf, con los paquetes necesarios	2 horas	Completada
T004	DES	Crear el repositorio en GitLab	0.5 horas	Completada
T005	DOC	Redacción de la introducción	1h	En curso
T006	DOC	Documentar los objetivos	4 horas	Completada
T007	DES DOC	Documentar la planificación del proyecto	8 horas	En curso
T008	DES	Investigar tecnologías para el desarrollo del proyecto	10 horas	Completada
T009	DES	Investigar tiendas ya creadas	4 horas	Completada
T010	DOC	Documentar estado del arte	0.5 horas	En Curso
Total Sprint			36 horas	

Tabla 6.1: Sprint 1

Al final del sprint (Tabla 6.1) se habían invertido en el un total de 36 horas, con algunas tareas que no estaban completas del todo, las cuales pasaron al siguiente sprint para ser completadas. No se ha llegado a las 40 horas que estaban planificadas para el sprint por motivos personales del desarrollador, pero al ser el primer sprint no se considera problemático el haber invertido menos horas dado que pueden ser recuperadas o necesarias en otros sprints.

6.1.2. Sprint 2

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T005	DOC	Redacción de la introducción	1h	Completada
T007	DES DOC	Documentar la planificación del proyecto	6 horas	Completada
T010	DOC	Documentar estado del arte	2 horas	completada
T011	DES	Buscar cursos para la formación en las tecnologías a emplear	2 horas	Completada
T012	DES	Formación en Angular	8 horas	En curso
T013	DES	Documentar las tecnologías investigadas	8 horas	Completada
T014	DES	Investigar herramientas para el diseño de interfaces web	2 horas	Completada
T015	DES	Bocetos de la interfaz web	12 horas	En curso
Total Sprint			40 horas	

Tabla 6.2: Sprint 2

En este sprint (6.2 se han llevado a cabo las tareas que tienen relación con la investigación de las tecnologías a emplear, se ha creado la planificación del proyecto, se ha comenzado con la formación en Angular y el diseño en bocetos de la interfaz web.

Se intentó encontrar algún software que permita la creación de interfaces web de una forma simple, pudiendo más tarde crear el HTML y CSS de la web de manera automática. Se desechó esta idea al encontrar las herramientas bastante complejas y con curvas de aprendizaje similares a la del creación con HTML y CSS directamente. Por lo tanto se toma la decisión de crear la web empleando HTML y CSS.

Se ha iniciado también la formación en Angular, la cuál será una tarea continua durante todo el proyecto con casi toda certeza.

Se han iniciado los bocetos de la interfaz, los cuales se realizan con un software en la nube. La tarea necesita más tiempo del esperado, ya que no están terminados todos los bocetos, por ello se pasa la tarea al siguiente sprint.

6.1.3. Sprint 3

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T012	DES	Formación en Angular	8 horas	En curso
T015	DES	Bocetos de la interfaz web	12 horas	Completada
T016	DES	Creación y configuración del back-end en Firebase	1 hora	Completada
T017	DES	Creación del proyecto Angular e instalación de los paquetes necesarios para el desarrollo del proyecto	3 horas	Completada
T018	DES	Configuración de Git	1 hora	Completada
T019	DOC	Documentar tecnologías empleadas	8 horas	En curso
T019	DOC	Presupuesto inicial del proyecto	6 horas	Completada
Total Sprint			39 horas	

Tabla 6.3: Sprint 3

Durante este sprint (Tabla 6.3) se ha seguido con la formación en Angular, se han completado los bocetos de la interfaz y se ha configurado el Back-End en Firebase. Se ha investigado sobre los costes de un diseñador web y se ha elaborado un presupuesto inicial para el cliente.

Se ha empezado a documentar las tecnologías que van a ser empleadas en el proyecto, pero como esto es susceptible a cambios en cosas como, por ejemplo, los paquetes npm empleados, la tarea sigue en curso para ir completando.

También se ha creado el repositorio de Git en GitLab y se ha configurado Git en local para poder tener las credenciales de GitLab guardadas en local para poder realizar tareas contra el repositorio remoto sin tener que introducir las credenciales de forma continua.

6.1.4. Sprint 4

En este sprint (Tabla 6.4) se ha continuado con la formación en Angular, se han investigado frameworks de desarrollo de webs como Bootstrap y NgBootstrap para incluirlos al proyecto, también se ha creado el proyecto Angular en local y se han incorporado los paquetes Bootstrap, NgBootstrap y PrimeNG para poder hacer uso de ellos en el desarrollo de las vistas web.

Se ha comenzado a crear las vistas en HTML desde los bocetos para poder luego darles funcionalidad mediante Angular.

A mitad del sprint se ha tenido una pequeña reunión en forma de llamada telefónica con el cliente para conocer su satisfacción con los diseños de la web, la cuál fue satisfactoria ya que al cliente le pareció correcta la interfaz web.

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T012	DES	Formación en Angular	8 horas	En curso
T019	DOC	Documentar tecnologías empleadas	4 horas	En curso
T020	DES	Investigar frameworks de desarrollo de HTML y CSS para Angular	4 horas	Completada
T021	DES	Creación del proyecto Angular y sincronización con repositorio remoto	1 hora	Completada
T022	DES	Añadir los paquetes npm al proyecto Angular y comprobar su correcto funcionamiento	1 hora	Completada
T023	DES	Creación de la header	2 horas	Completada
T024	DES	Creación de la vista de Homepage o Página Principal	3 horas	Completada
T025	DOC	Diagrama de navegación del sitio web	3 horas	Completada
T026	DOC	Documentar la interfaz web y paleta de colores	4 horas	Completada
T027	DES	Creación de la vista sobre nosotros	1 hora	Completada
T028	REU	Llamada telefónica con el cliente	1 hora	Completada
T029	DES	Creación del a template del componente card producto	2 horas	Completada
Total Sprint			35 horas	

Tabla 6.4: Sprint 4

6.1.5. Sprint 5

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T012	DES	Formación en Angular	8 horas	En curso
T019	DOC	Documentar tecnologías empleadas	0 horas	En curso
T030	DES	Creación de la vista de inicio de sesión y registro	1 hora	Completada
T031	DES	Creación del servicio de autenticación	1 horas	En curso
T032	DES	Dar funcionalidad a la vista de inicio de sesión y registro para poder probar la autenticación	0.5 horas	Completada
T033	DES	Añadir e inicializar el módulo de Firebase Auth en la aplicación y probar la autenticación sin estado	4 horas	Completada
T034	DES	Creación de la vista de Producto	1 hora	Completada
T035	DES	Creación de la vista de completar registro 1	1 horas	Completada
T036	DES	Creación de la vista de completar registro 2	0.5 horas	Completada
T037	DES	Creación de la vista de perfil cuenta y sus sub-vistas	6 horas	Completada
T038	DES	Creación de la vista sobre nosotros	0.25 horas	Completada
T039	DES	Creación de la vista del formulario de contacto	0.5 horas	Completada
T040	DES	Creación de la vista de condiciones del servicio	2 horas	Completada
T041	DES	Creación de la vista Cesta de la compra	3 horas	Completada
T042	DES	Creación de la vista de pedido	2 horas	Completada
T043	DES	Creación de las vistas de pago y de completar pago	2 horas	Completada
T044	DES	Creación del módulo de routing	3 horas	En Curso
T045	DES	Creación de la vista principal de la tienda	2 horas	Completada
T046	DES	Recopilación de productos de la antigua tienda del cliente e inclusión en base de datos	1 hora	Completada
T047	DES	Formación sobre Storage y la conexión con Firebase	2 horas	Completada
Total Sprint			41 horas	

Tabla 6.5: Sprint 5

En este sprint(Tabla 6.5) se ha seguido con la creación de los archivos HTML para las vistas de el sitio web y se ha creado algún componente que va dentro como componente de esas vistas como el card-producto. Se ha dado una funcionalidad limitada a la vista de inicio de sesión para poder comprobar si el módulo de Firebase para angular funcionaba correctamente. Se ha incluido el routing inicial de la aplicación para poder acceder a los componentes mediante la ruta web.

Se han buscado ejemplos de como introducir las imágenes dentro del módulo Storage de Firebase y de cómo poder hacerlas referencia en Firestore para poder recuperarlas luego y poder mostrarlas en la página web. Se han recopilado dos productos y se han introducido sus datos en base de datos para poder probar el funcionamiento de el servicio de base de datos.

6.1.6. Sprint 6

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T012	DES	Formación en Angular	8 horas	Completada
T019	DOC	Documentar tecnologías empleadas	0.5 horas	En curso
T048	DES	Creación de las clases TypesCript para los tipos de datos	1 hora	Completada
T049	DES	Modificación de el proceso de registro de nuevos usuarios e implementación	6 horas	Completada
T050	DES	Dar funcionalidad al despliegue de los campos en la vista Condiciones del Servicio	2 horas	Completada
T051	DES	Cargar los productos en la vista de la tienda	4 horas	Completada
T052	DES	Implementar el envío de consultas desde la vista Contacto	2 horas	Completada
T053	DES	Añadir productos a la cesta de la compra	3 horas	Completada
T054	DES	Dar funcionalidad a la vista de producto	3 horas	Completada
T055	DES	Hacer la header colapsable y funcional al ser colapsarse	2 horas	Completada
T056	DES	Categorías de productos	3 horas	Completada
T057	DES	Carousel de productos en la página principal	2 horas	En curso
T058	DES	Cambiar cantidad de un producto en la cesta	1 hora	Completada
T059	DES	Implementado el realizar un pedido	6 horas	Completada
T060	DES	Mostrar pedidos de cliente	4 horas	Completada
T061	DES	Volcar datos en base de datos de la cesta en usuario cada vez que la modifica	2 horas	Completada
Total Sprint			40.5 horas	

Tabla 6.6: Sprint 6

Durante este sprint (Tabla 6.6 se ha dado funcionalidad a las características principales de la tienda, como mostrar los productos, añadirlos a la cesta, realizar un pedido... También se han implementado volcados de datos en base de datos para las acciones que realiza un usuario.

Se ha dado por completada la formación en Angular, dado que se entiende que ya se poseen conocimientos suficientes para llevar a cabo los desarrollos pendientes del proyecto.

Se han invertido 0.5 horas en documentar PrimeNG y NgBootstrap, los cuales han sido empleados para el carousel de la página principal y para el carousel de imágenes de la vista de un producto detallado.

Se da por completada la tarea de documentar tecnologías empleadas al no preveer la inclusión de nuevas en el proyecto.

6.1.7. Sprint 7

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T062	DES	Implementar auto-login para mantener abierta la sesión del usuario	6 horas	Completada
T063	BUG	Arreglar un error que hace que se dupliquen los productos al iniciar sesión	2 horas	Completada
T064	DES	Implementar modificación de datos personales	4 horas	Completada
T065	DES	Implementar cambio de direcciones	3 horas	Completada
T066	TEST	Planificar las pruebas con usuarios	2 horas	Completada
T067	DOC	Diagramas de la arquitectura de la aplicación y redacción en la memoria	15 horas	Completada
T068	DES	Cancelar un pedido	1 hora	Completada
T069	DES	Reestructuración del código para solo tener un NgModule	2 horas	Completada
T070	BUG	Arreglar un error en el cuál el stock de un producto al realizar un pedido no se modificaba	2 horas	Completada
T071	DES	Añadir iconos y spinners a la aplicación	2 horas	Completada
Total Sprint			39 horas	

Tabla 6.7: Sprint 7

Durante este sprint se ha implementado un auto-inicio de sesión para si un usuario cierra el navegador, no pierda su estado en la aplicación y cuando vuelva a acceder a la web conserve su cesta y su sesión iniciada. Esto generó un error, ya que al iniciar sesión se duplicaban los productos que un usuario tuviera en su cesta de la compra, ya que el inicio de sesión contaba a efectos prácticos como auto-login y la lógica de este era disparada al iniciar sesión.

Se ha implementado la funcionalidad de que usuario pueda cambiar sus datos personales y sus direcciones de facturación.

Se han creado los diagramas de componentes y de dependencias de la aplicación y se han incluido en la memoria del proyecto.

6.1.8. Sprint 8

Tarea	Tipo	Descripción	Tiempo invertido	Estado
T072	BUG	Reparar bugs visuales	3 horas	Completada
T073	DES	Despliegue de la aplicación en Firebase Hosting	2 horas	Completada
T074	TEST	Pruebas con usuarios	6 horas	Completada
T075	BUG	Error en la recuperación de contraseña	0.5 horas	Completada
T076	DES BUG	Cambiar comportamiento de la header en dispositivos móviles	0.5 horas	Completada
T077	BUG	Arreglar los enlaces en el registro hacia condiciones del servicio y política de protección de datos	0.25 horas	Completada
T078	DES	Modificar el comportamiento de las flechas del carousel en un producto	0.5 horas	Completada
T079	BUG	Reparar el formulario de contacto	0.5 horas	Completada
T080	BUG	Error en la suma de productos	0.5 horas	Completada
T081	DES BUG	Eliminar botones de inicio de sesión con proveedores externos	0.25 horas	Completada
T082	BUG	Error en el proceso de registro, en las direcciones	1 hora	Completada
T083	DOC	Documentar implementación	3 horas	Completada
T084	DES	Comentar código y eliminar logs innecesarios	4 horas	Completada
T085	DOC	Redactar manual de instalación	1 hora	Completada
T086	DOC	Documentar las pruebas con usuarios	3 horas	Completada
T087	DOC	Redactar abstract	0.5 horas	Completada
T088	DOC	Redactar conclusiones	1 hora	Completada
T089	DOC	Revisar y completar documentación	6 horas	Completada
Total Sprint			33.5 horas	

Tabla 6.8: Sprint 8

Este sprint (Tabla 6.8) ha sido el sprint final del proyecto y en el transcurso de él se ha completado la documentación y se ha desplegado y probado la aplicación en Firebase Hosting. También se han realizado pruebas con usuarios y se han documentado. Se han corregido, en la medida de lo posible, los errores detectados en la aplicación, y se han implementado las sugerencias, viables de implementar, recibidas por los usuarios de prueba,. Se ha intentado tener una reunión con el cliente para informarle del devenir del proyecto pero no se ha podido realizar por que estaba de viaje nuevamente. Se realizará posteriormente pero no se incluirá en este seguimiento.

Con respecto a lo planificado se han invertido 13.5 horas más de las planificadas en este sprint, pero al afectar a tareas consideradas de alta importancia sabiendo que algunos sprints han sido más cortos de lo planificado, se alargó este sprint sin afectar casi de forma muy notoria a las horas totales del proyecto.

6.1.9. Fin del seguimiento

Se han empleado en la realización del proyecto un total de 304 horas, lo cuál supone 4 horas de desviación de la planificación original expuesta en la tabla 2.1.
La fecha en la que se ha finalizado el proyecto el día 20 de Junio de 2022, la cual es anterior a la fecha de entrega del proyecto (24 de Junio de 2022, recogida en el apartado 2.3.1).

Capítulo 7

Diseño

7.1. Modelo Vista Controlador

El ModeloVistaControlador (MVC por sus siglas) es un patrón del diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Se basa en la separación entre la lógica de negocio, los datos de esta y su visualización. Esta separación hace que exista una mejor división del trabajo y que sea mucho más simple y rápido el mantenimiento de sistemas y productos que sigan este patrón.

- **Modelo:** es la parte que se encarga de gestionar los datos y la información de la aplicación.
- **Vista:** es la parte encargada de generar y mostrar la interfaz de usuario.
- **Controlador:** es el intermediario entre vista y modelo, se encarga de ordenar al modelo y la vista qué deben mostrar y almacenar.

Un diagrama del funcionamiento de este patrón se puede ver en la Figura 7.1

7.1.1. Modelo

El modelo [42] define los datos que debe contener la aplicación. Es el encargado de gestionar la información y los datos con los que la aplicación trabaja. Si el estado de estos datos cambia, el modelo notificará a la vista de estos cambio para que pueda mostrar los datos correctos, y si es necesario también notifica a el controlador de estos cambios por si necesita realizar alguna acción lógica derivada de estos cambios.

7.1.2. Vista

La vista presenta o muestra la información necesario en un formato apropiado con el que el usuario puede interactuar. También es la encargada de recoger las interacciones de usuario con la aplicación y notificar a el controlador de estas acciones para que pueda realizar las tareas lógicas precisas.

7.1.3. Controlador

El controlador gestiona los eventos y las acciones del usuario, en base a lo cuál realiza peticiones al modelo para que modifique sus datos. También se encarga de ordenar a la vista cómo debe actuar en base a un evento generado por el usuario, si necesita actualizar los datos provenientes del modelo, si debe navegar a una nueva ventana...

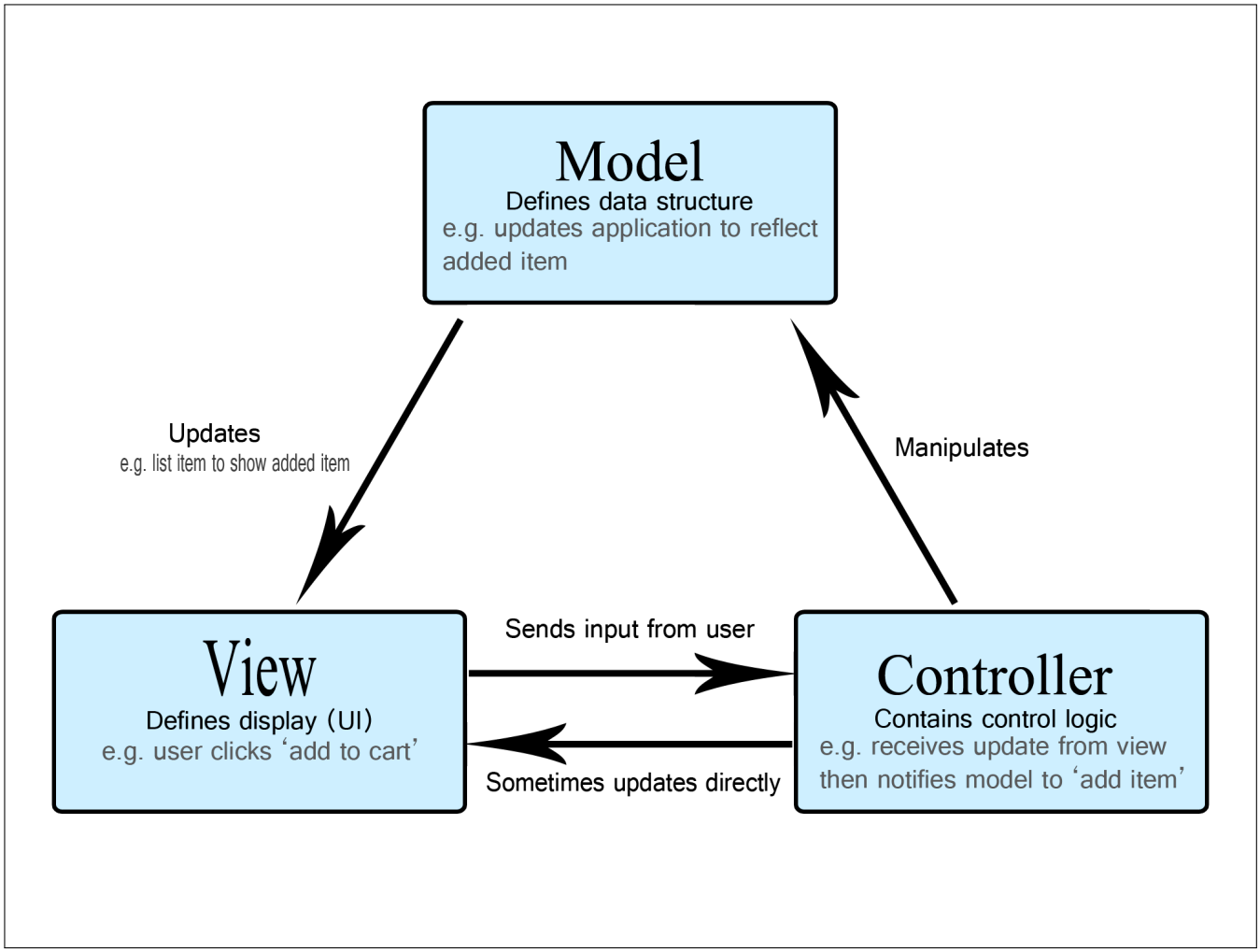


Figura 7.1: Diagrama funcionamiento MVC

Haciendo una analogía con una orquesta, podríamos decir que el controlador es el director, el modelo son las partituras y la vista es la música generada por los instrumentos y músicos, los eventos serían, por ejemplo, la reacción del público (aplausos o abucheos, lo cuál haría a la orquesta hacer un bis, o ordenar al director cambiar de partitura.

7.1.4. MVC en aplicaciones web

Este patrón es comúnmente empleado en el diseño de aplicaciones web, debido a que las partes que componen una web son normalmente paralelas al MVC. Las aplicaciones web, normalmente, poseen bases de datos locales o remotas que almacenan los datos referentes a la aplicación, esta parte encaja con la definición de Modelo expresada anteriormente. Las webs modernas poseen interfaces creadas con HTML5 y CSS, las cuales permiten mostrar información al usuario final de una forma ordenada, es parte encaja con la definición de Vista anterior. También suelen poseer código para el control de esta vista y modelo, el cuál suele estar escrito en HTML y JavaScript (u otros lenguajes de scripting), esta parte encaja con la definición de Controlador.

7.1.5. Ventajas del patrón MVC

- Es un modelo de arquitectura muy estandarizado, lo que permite que el día de mañana otros desarrolladores comprendan la separación del código de la aplicación de forma rápida y eficaz.

- Este tipo de arquitectura tiene una base bien definida, lo que permite ahorro de tiempo en el desarrollo, además su estructura permite la reutilización de código de forma simple. Por ejemplo, un mismo modelo puede tener varias vistas que dependan de él.
- Sigue un flujo de interacción sencillo, lo cuál permite entender el código de una manera simple y esto, a su vez, permite que el mantenimiento y corrección de errores sea mucho más sencillo.
- Crear o añadir nueva funcionalidad al sistema es más sencillo que en otro tipo de arquitecturas.

7.1.6. Flujo de control habitual

El software desarrollado con el patrón MVC sigue, por norma general, el siguiente flujo de interacción: [42]

1 El usuario interactúa con la interfaz (vista) de alguna manera, por ejemplo pulsando un botón, un enlace, haciendo scroll...

2 El controlador recibe por parte de los eventos de la interfaz la acción que el usuario ha realizado y gestiona dichos eventos.

3 El controlador efectúa una petición al modelo, el cual consultará nueva información, la actualizará o la borrará. En ciertas ocasiones el controlador no verá esto necesario y le enviará una respuesta directa a la vista.

4 El modelo enviará la nueva información o petición al controlador el cual a su vez delegará en la vista la tarea de desplegar la interfaz de usuario con los datos actualizados.

5 Finalmente la interfaz queda a la espera de nuevas interacciones por parte del usuario, comenzando el ciclo de nuevo.

En algunas implementaciones, la vista y el modelo pueden llegar a interactuar directamente, aunque no es lo habitual.

7.1.7. MVC en este proyecto

Angular, la tecnología que empleamos en el desarrollo de la parte de front-end de este proyecto sigue un modelo de arquitectura que se asimila bastante al MVC. Cada componente de forma independiente poseería las tres partes de esta arquitectura, aunque, como explicamos anteriormente, el modelo puede ser compartido entre múltiples componentes. En este framework, la parte de la vista sería la asociada al *template* la cuál contiene el código HTML y la lógica de interfaz de Angular. La parte del controlador sería el fichero TypeScript asociado a el componente. El modelo serían los servicios encargados de almacenar datos, realizar consultas a base de datos y almacenar información sobre estas consultas.

7.2. Desarrollo software basado en componentes

El desarrollo software basado en componentes es un modelo que describe construye y emplea técnicas software para elaborar sistemas abierto y distribuidos, mediante el ensamblaje de partes software reutilizables. [9]

7.2.1. Características

Este modelo es usado para reducir los costes, tiempo y esfuerzos del desarrollo de software. La modularidad, reusabilidad, y componibilidad son características nucleares de este modelo de desarrollo. El diseño basado en componentes se engloba dentro del paradigma de la programación de sistemas abiertos, los cuales son extensibles y tienen interacción con componentes heterogéneos que se integran o abandonan el sistema de forma dinámica, esto es, que los componentes pueden ser sustituidos por otros componentes, independientemente de su arquitectura subyacente, sin crear errores por esta sustitución.

7.2.2. Beneficios del diseño software basado en componentes

- **Simplifica las pruebas:** permite que las funcionalidades concretas de cada componente puedan ser probadas antes de probar el sistema completo con todos los componentes ensamblados.
- **Simplifica el mantenimiento:** cuando no existe apenas acoplamiento entre los componentes, se puede únicamente solucionar los problemas del componente que los está creando, sin necesidad de modificar el resto para arreglar el error.
- **Mayor calidad:** los componentes desarrollados pueden ser optimizados de forma continua sin afectar al resto de componentes, lo cual radica en una mayor calidad de la aplicación final y una mejora a lo largo del tiempo.
- **Reutilización de código:** permite ahorrar tiempo de desarrollo debido a la reutilización de código, ya que una vista compuesta de diversos componentes ya programados no necesita apenas lógica para hacerla funcionar correctamente.
- **Funcionalidad mejorada:** al usar un componente que contenga funcionalidad, solo se necesita entender qué hace, y no el cómo lo hace, lo cuál reduce los tiempos de desarrollo al existir una fachada entre lo que el componente hace y el cómo lo hace. Al añadir un componente a nuestro sistema solo debemos preocuparnos de conectarlo de forma correcta, no de la lógica subyacente que hace que el componente funcione.

7.2.3. Componentes en Angular

Angular fue diseñado con el desarrollo basado en componentes en mente. En la Figura 7.2 se puede ver un diagrama explicativo de como está formado un componente Angular, el cuál se compone de una clase de TypeScript y un *template* HTML los cuales se relacionan entre si por medio del "`data binding`çomo se aprecia en la Figura 7.2.

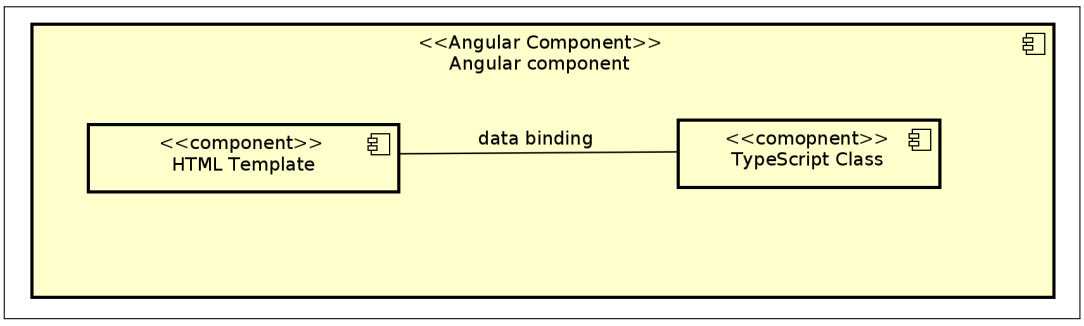


Figura 7.2: Diagrama de componentes: componente Angular básico

Un componente Angular puede contener dentro de sí otros componentes. En la Figura 7.3 se puede ver cómo un componente puede ser hijo de otro componente. El componente hijo es parte del *template* HTML, y padre e hijo pueden comunicarse entre sí.

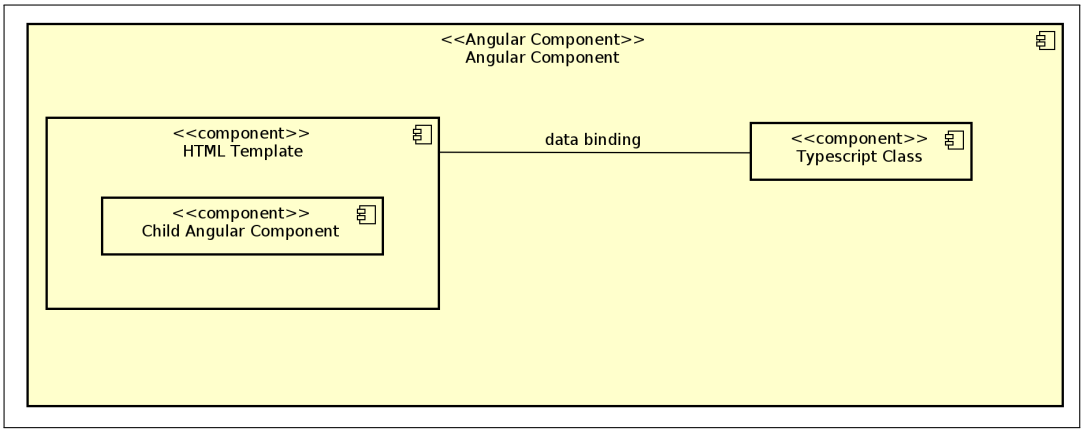


Figura 7.3: Diagrama de componentes: componente Angular con componente hijo

Los servicios en Angular son inyectados mediante el inyector de dependencias, el cuál actúa como interfaz para el componente y permite hacer uso de los métodos proporcionados por el servicio de una forma simple. En la Figura 7.4 se detalla esta interacción entre componentes y servicios.

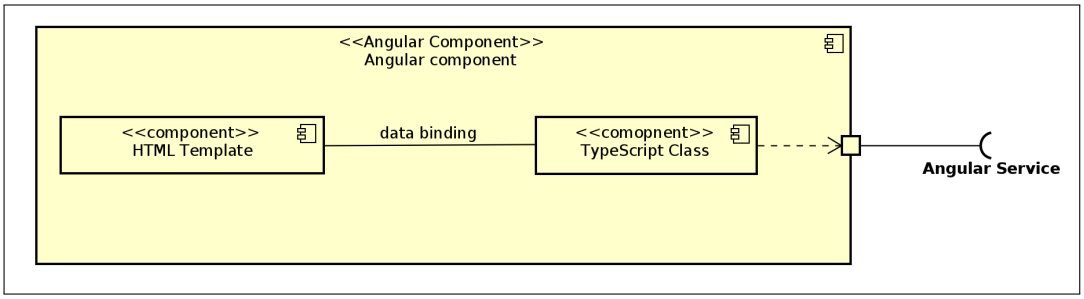


Figura 7.4: Diagrama de componentes: componente con servicio

Las directivas de angular pueden ser parte del componente angular también. Las directivas actúan como un componente reutilizable que añade funcionalidad al *template* HTML. En la Figura 7.5 se puede ver esta relación.

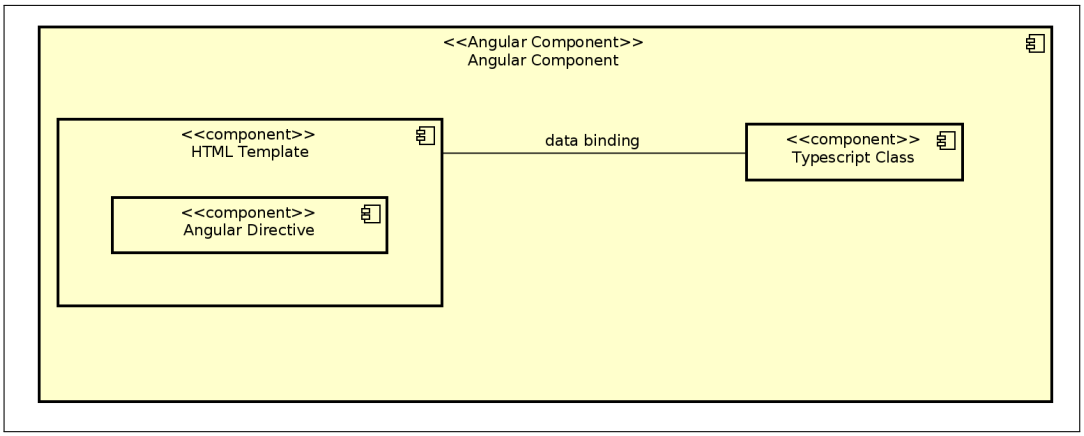


Figura 7.5: Diagrama de componentes: componente con directiva

Los servicios también son componentes por definición en lo que respecta al diseño software basado en componentes, pero no son componentes de la forma en que Angular entiende los componentes. Son piezas de código reusable, pero no componentes Angular al uso.

7.2.4. Componentes creados para este proyecto

A continuación se detallan los componentes y servicios Angular creados para este proyecto. La distinción entre componente Angular y servicio se hace visible en los estereotipos de los diagramas.

Componentes Angular Creados:

- `App:` es el encargado de contener la aplicación completa. Contiene la header y un router-outlet. (Figura 7.6)

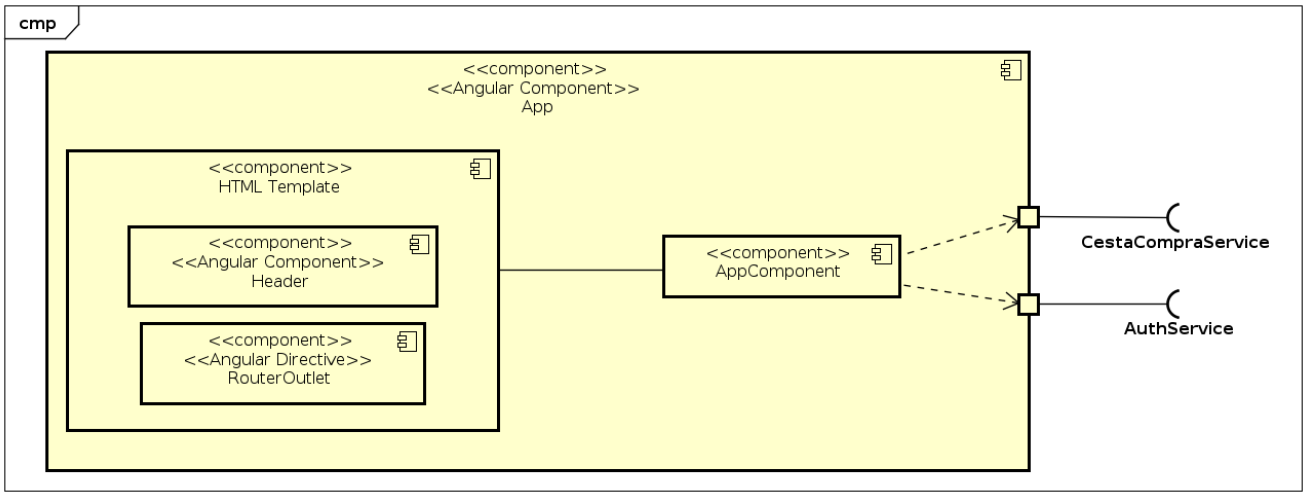


Figura 7.6: Componente App

- `Tienda:` contiene un router-outlet que muestra los distintos componentes de la tienda. Figura 7.7)

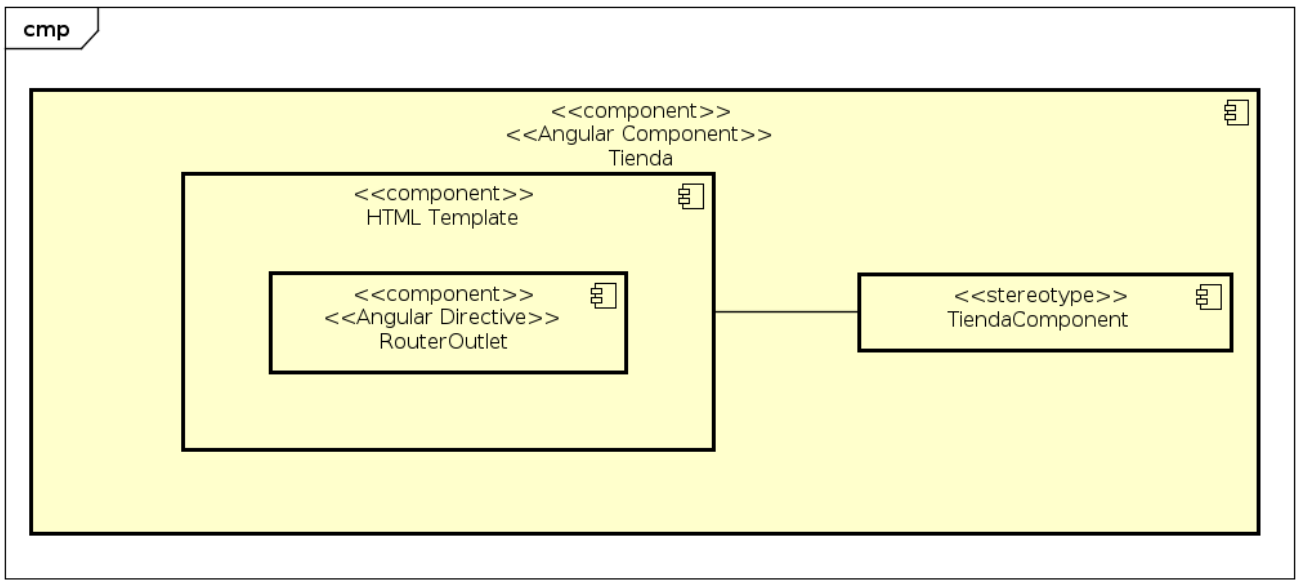


Figura 7.7: Componente Tienda

- `TiendaTienda:` contiene la pagina principal de la tienda. (Figura 7.8)

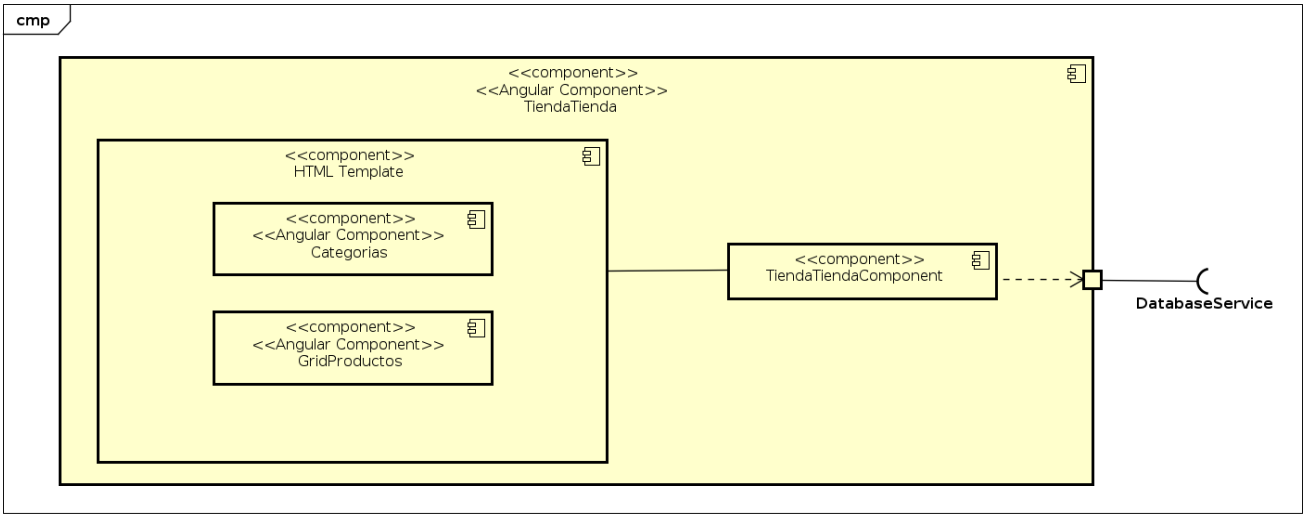


Figura 7.8: Componente TiendaTienda

- `CardProducto:` contiene un producto de la tienda que se muestra de una forma abreviada. (Figura 7.9)

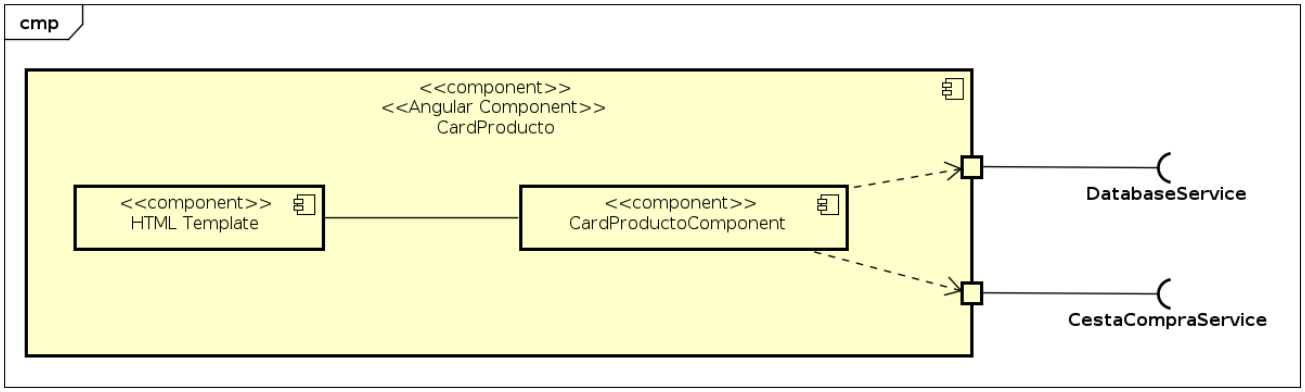


Figura 7.9: Componente CardProducto

- `CarouselProductos:` contiene un numero de *cards de producto* en la pagina principal de la aplicación de forma cíclica. (Figura 7.10)
- `Categorias:` contiene nombres de categorías de la tienda. (Figura 7.11)
- `GridProductos:` contiene productos en forma de *card producto*. (Figura 7.12)
- `Producto:` se encarga de contener un producto de forma detallada. (Figura 7.13)
- `SobreNosotros:` contiene la vista "Sobre nosotros", la cuál se compone de texto e imágenes que contarán la historia de la empresa. (Figura 7.14)
- `InicioSesionRegistro:` contiene la vista de inicio de sesión y registro. (Figura 7.15)
- `CompletarRegistro:` contiene el formulario completo de registro. (Figura 7.16)
- `ContrasenaOlvidada:` contiene un formulario para enviar un correo de recuperación de contraseña al usuario. (Figura 7.17)

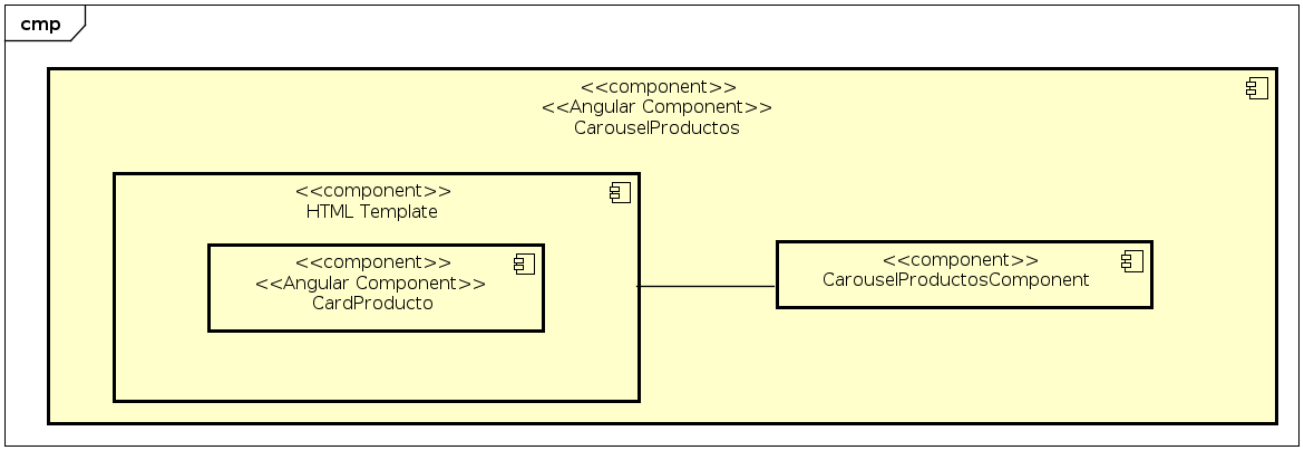


Figura 7.10: Componente CarouselProductos

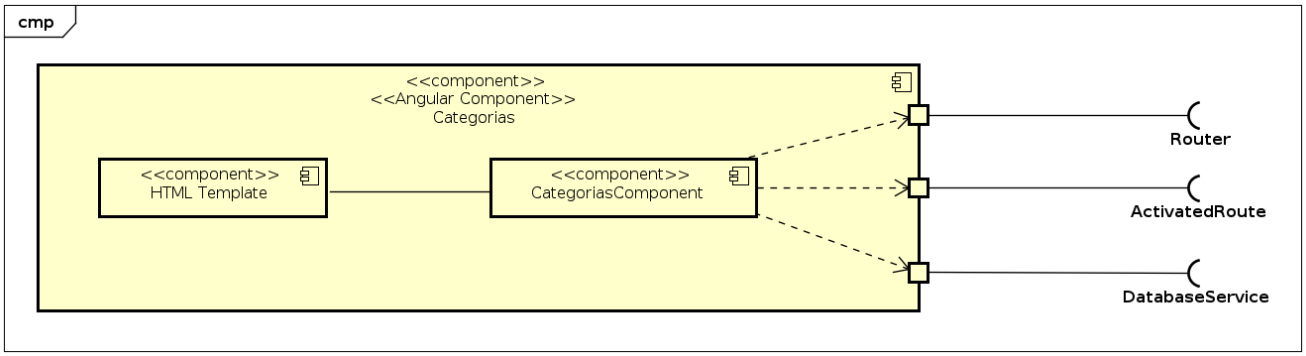


Figura 7.11: Componente Categorias

- `RestablecerPassword:` contiene un formulario para que el usuario pueda introducir una nueva contraseña para la cuenta después de olvidar la contraseña. (Figura 7.18)
- `PerfilCuenta:` contiene una columna de navegación en la parte izquierda y un router-outlet (Figura 7.19)
- `CerrarSesion:` se encarga de contener un mensaje de que se está cerrando la sesión. (Figura 7.20)
- `DatosPago:` contiene los datos de pago que posee un usuario. (Figura 7.21)
- `DatosPersonales:` contiene los datos personales de un usuario. (Figura 7.22)
- `Direcciones:` contiene las direcciones de un usuario. (Figura 7.23)
- `Pedidos:` contiene todos los pedidos de un usuario. (Figura 7.24)
- `CardPedidoResumenPerfil:` contiene de forma resumida un pedido de un usuario. (Figura 7.25)
- `Pedido:` se encarga de contener una descripción detallada de un pedido de un usuario. (Figura 7.26)
- `PedidoItem:` se encarga de contener uno de los productos que se ha incluido dentro del pedido y las unidades y costes asociados a ese producto. (Figura 7.27)
- `Homepage:` contiene la página de llegada a la aplicación. (Figura 7.28)
- `Header:` contiene la barra de navegación, que siempre está presente en la aplicación. (Figura 7.29)

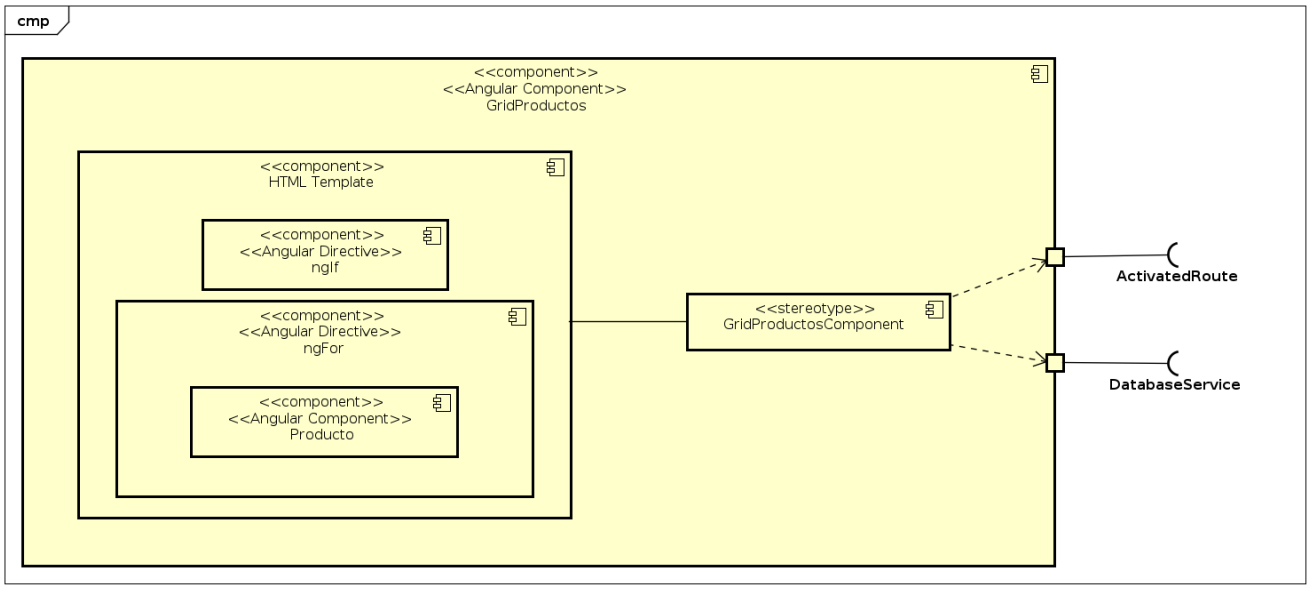


Figura 7.12: Componente GridProductos

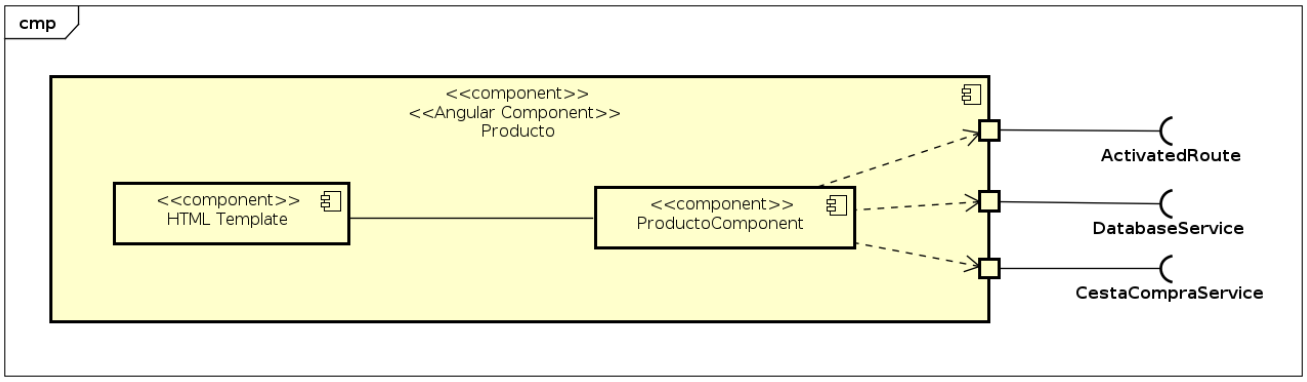


Figura 7.13: Componente Producto

- `Contacto:` contiene el formulario de contacto. (Figura 7.30)
- `Cesta:` contiene los productos que el usuario ha añadido a la cesta, en forma de ItemCesta, y los costes asociados al pedido que puede realizar el usuario. (Figura 7.31)
- `ItemCesta:` se encarga de contener uno de los productos que ha añadido el usuario a la cesta. (Figura 7.32)
- `CompletarPedido:` contiene el formulario de realizar pedido. (Figura 7.33)
- `CompletarPedidoPago:` contiene el formulario de completar el pedido y pagar, y de crear el pedido en la base de datos. (Figura 7.34)

Los servicios Angular creados son los siguientes:

- `AuthService:` este servicio se encarga de todo lo referente a autenticación de usuarios, creación de usuarios, actualización de credenciales... Se comunica con diversos componentes para que reciban datos del usuario que está con sesión iniciada en la aplicación y para recibir datos y realizar operaciones contra el módulo *Authentication* de Fireabse. También se comunica con el *DatabaseService* para recuperar los datos del usuario que ha iniciado sesión o para incluir en base de datos los datos

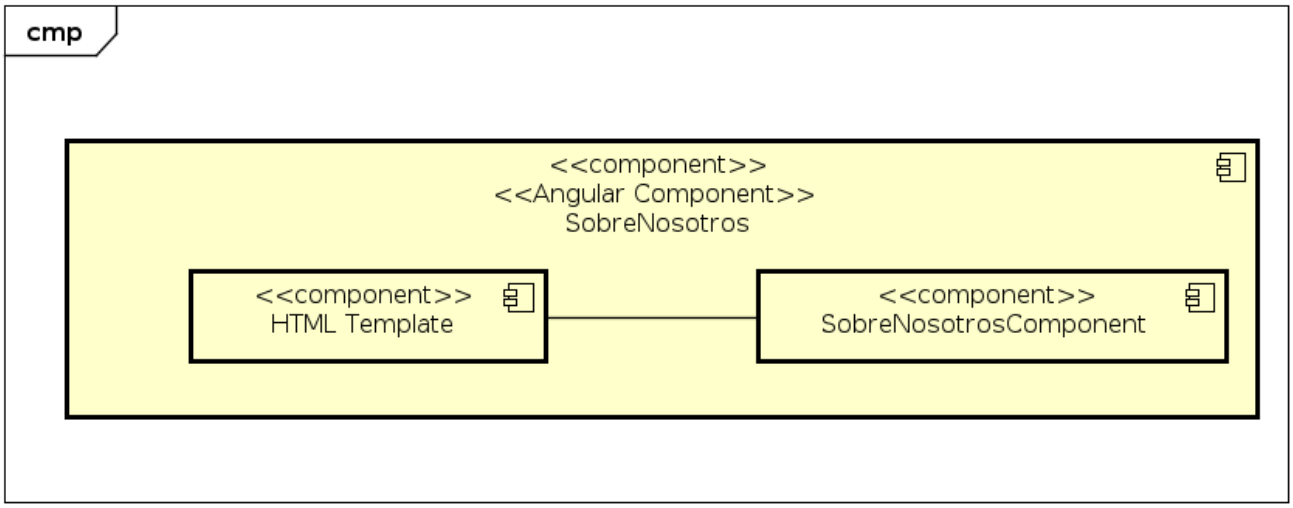


Figura 7.14: Componente SobreNosotros

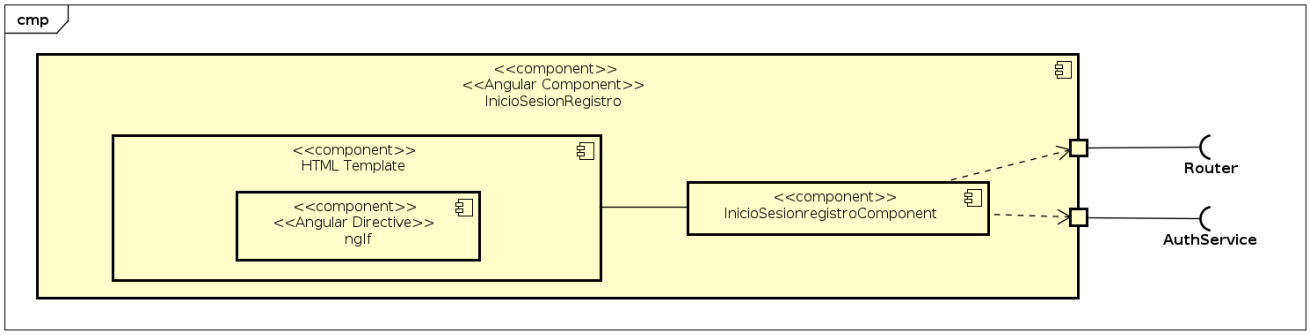


Figura 7.15: Componente InicioSesionRegistro

de nuevos usuarios creados. (Figura 7.35)

- `DatabaseService:` este servicio se encarga de la comunicación con la base de datos Firestore de Firebase. Posee las operaciones CRUD mínimas necesarias para el funcionamiento de la aplicación. (Figura 7.36)
- `CestaCompraService :` este servicio es el encargado de almacenar los productos que el usuario ha introducido en la cesta de la compra y de notificar a los componentes que lo necesiten de cuando realiza estos cambios. También es el encargado de recuperar por medio del *DatabaseService* la cesta que reside en Firestore de un usuario que ha iniciado sesión y de ir actualizándola a medida que el usuario va añadiendo o eliminando productos. También se encarga de la creación de pedidos y por medio del *DatabaseService* los crea en la base de datos.
  (Figura 7.37)

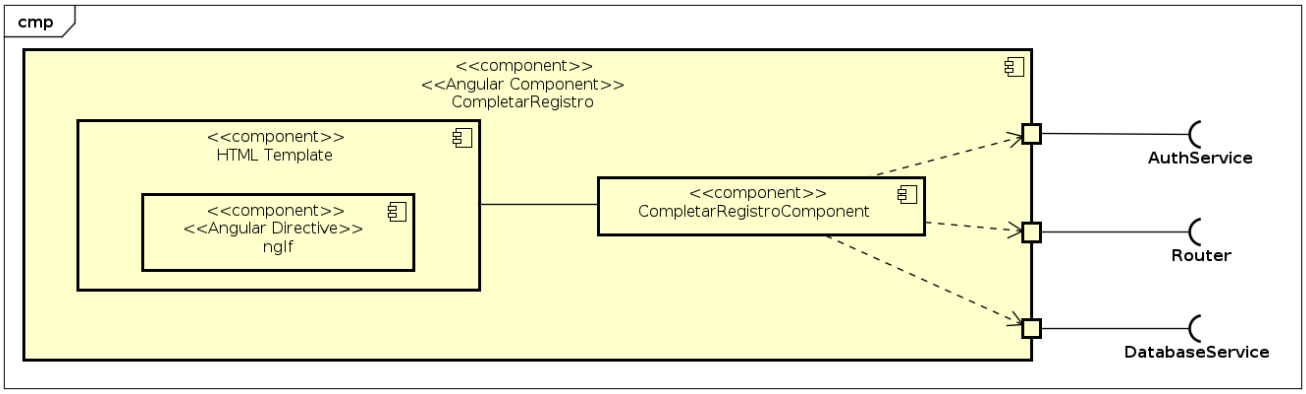


Figura 7.16: Componente CompletarRegistro

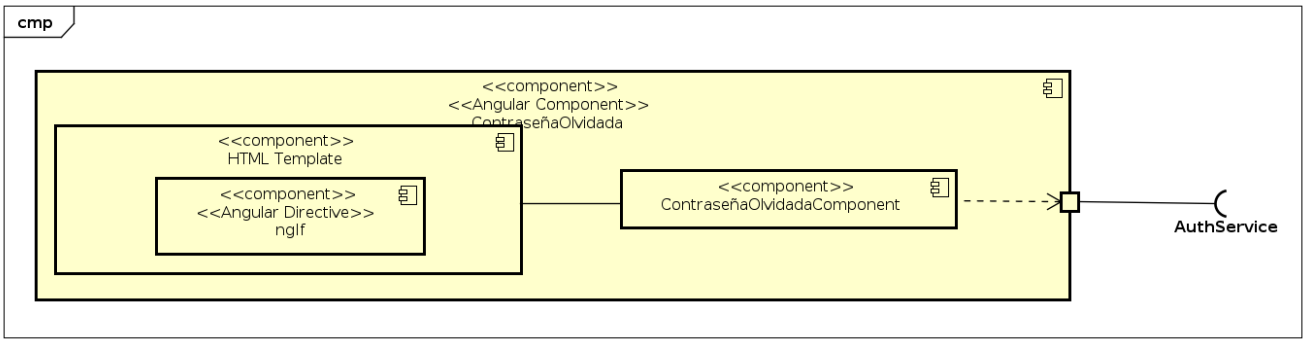


Figura 7.17: Componente ContrasenaOlvidada

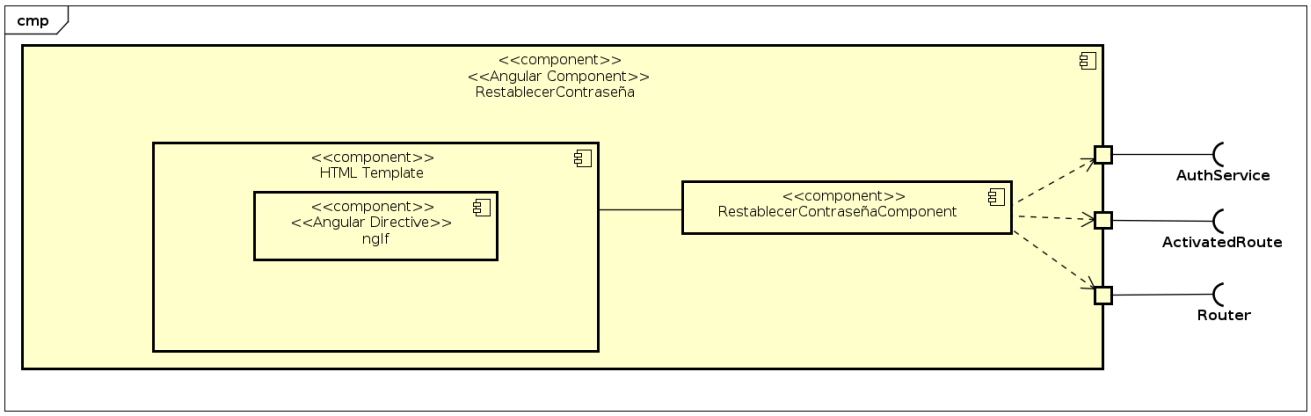


Figura 7.18: Componente RestablecerPassword

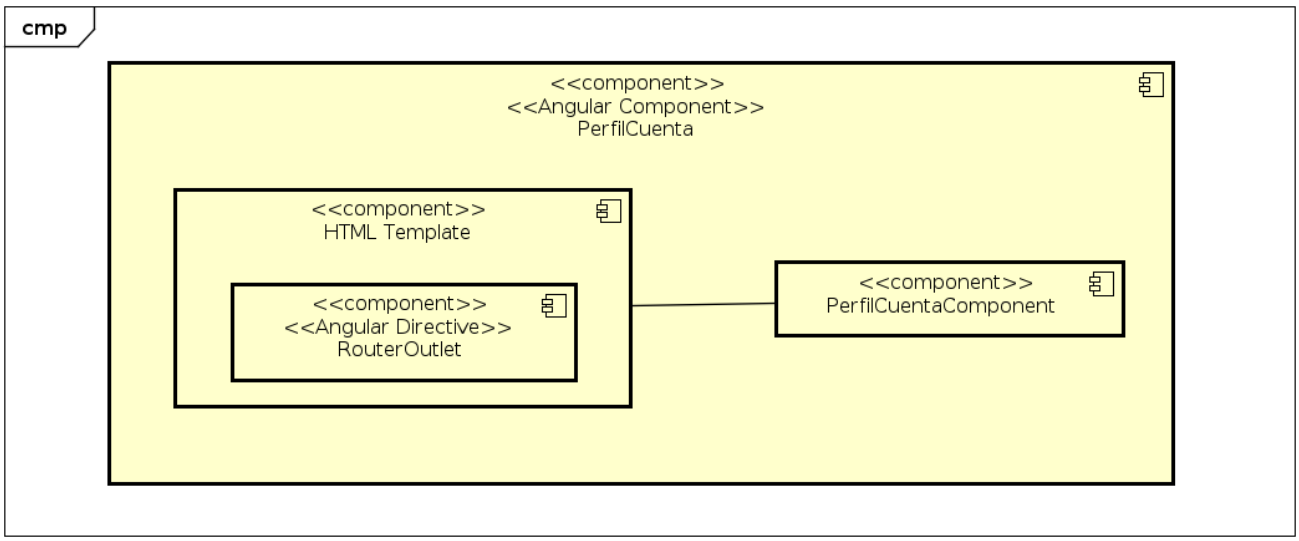


Figura 7.19: Componente PerfilCuenta

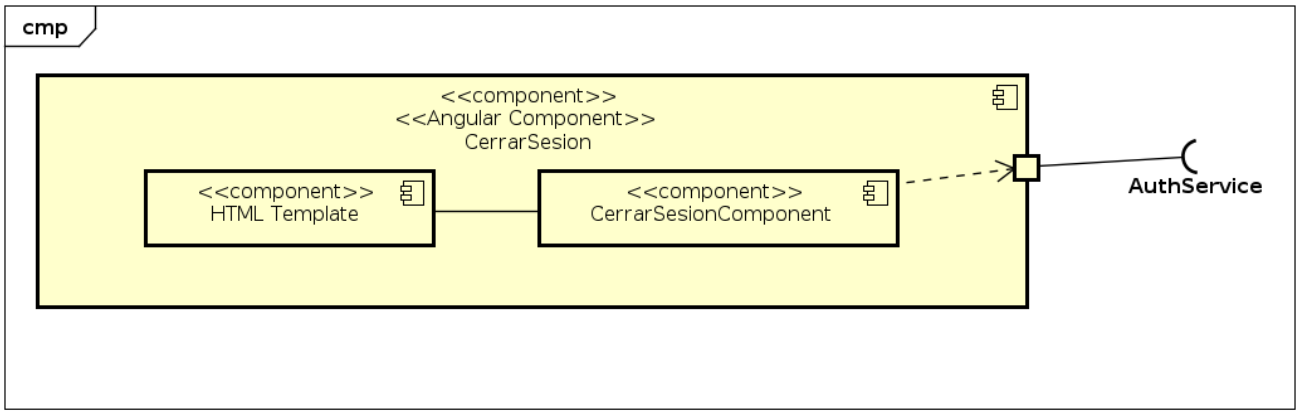


Figura 7.20: Componente CerrarSesion

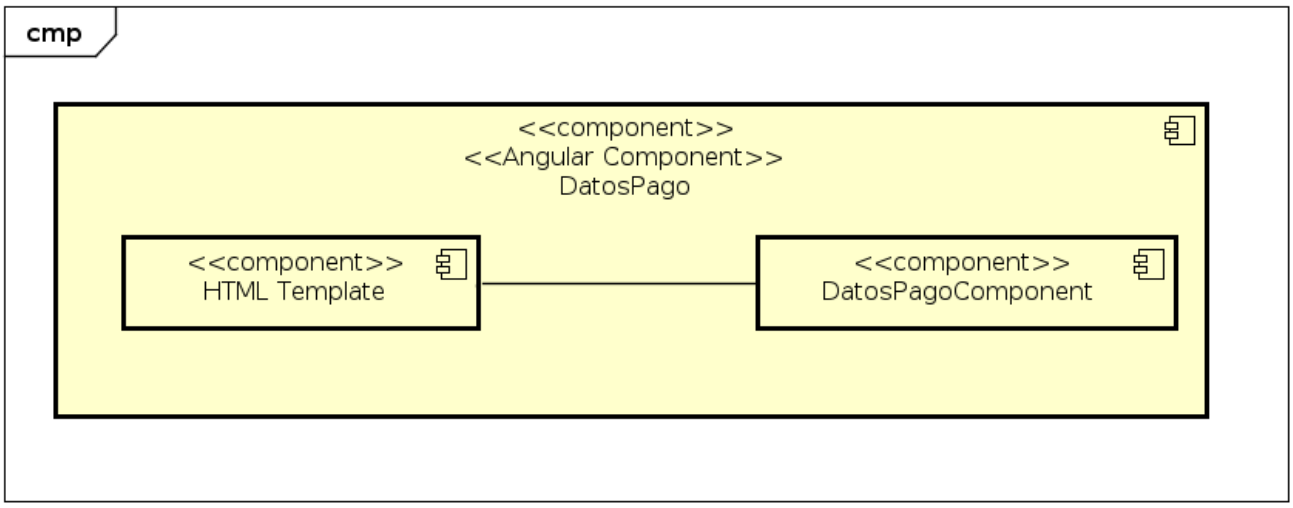


Figura 7.21: Componente DatosPago

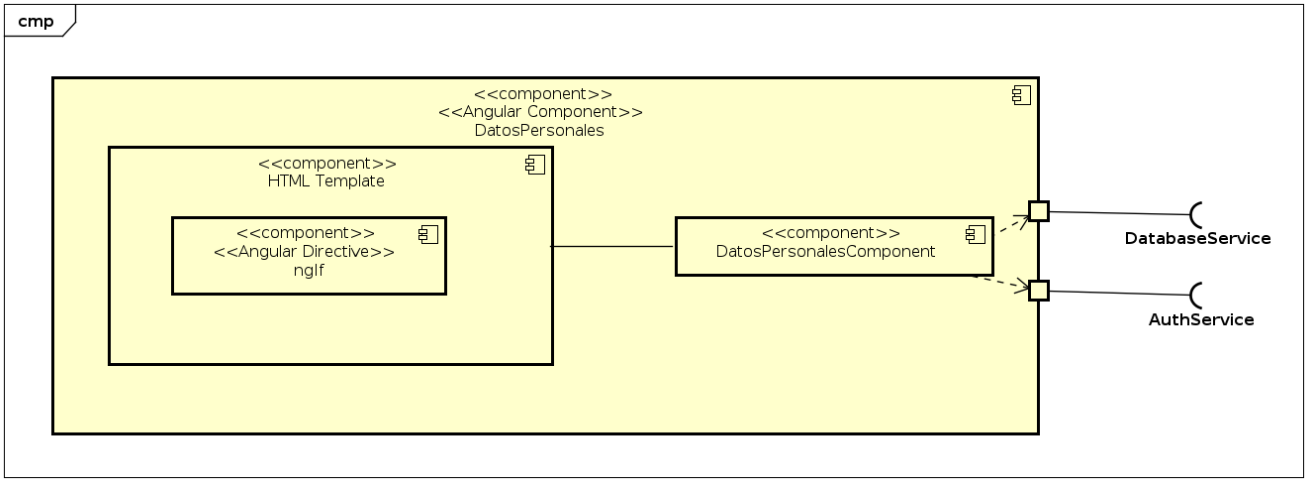


Figura 7.22: Componente DatosPersonales

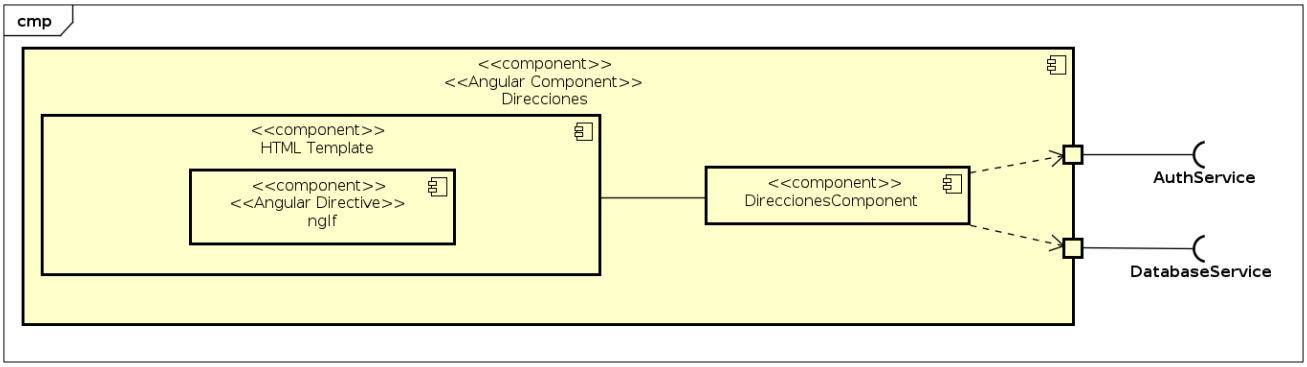


Figura 7.23: Componente Direcciones

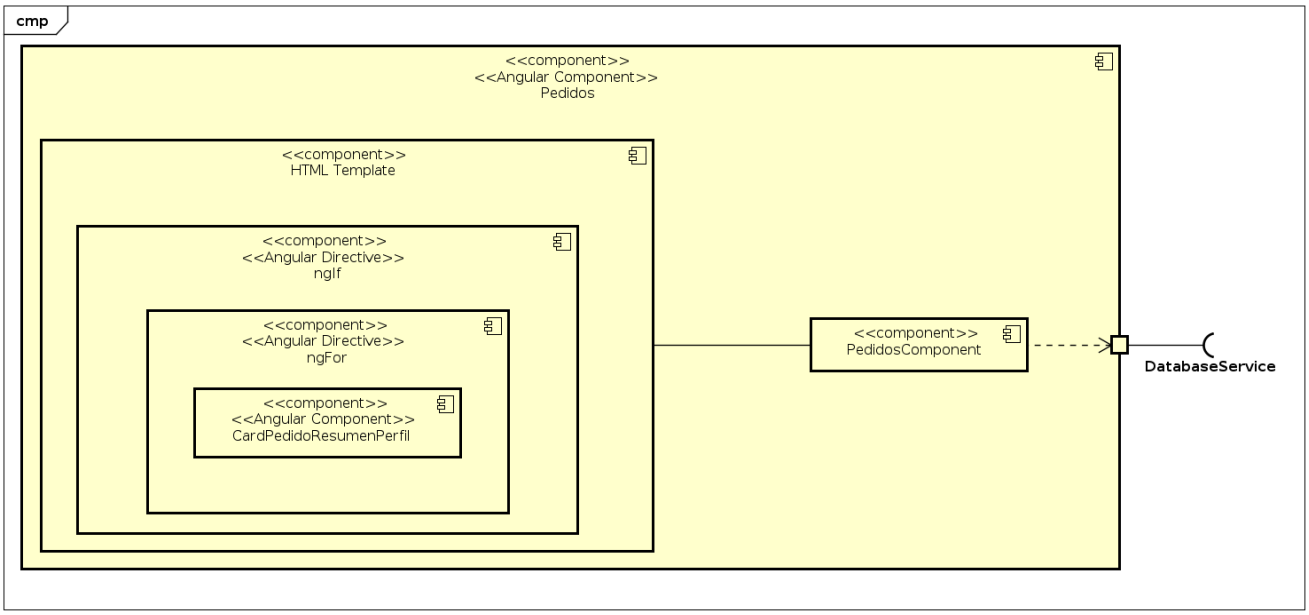


Figura 7.24: Componente Pedidos

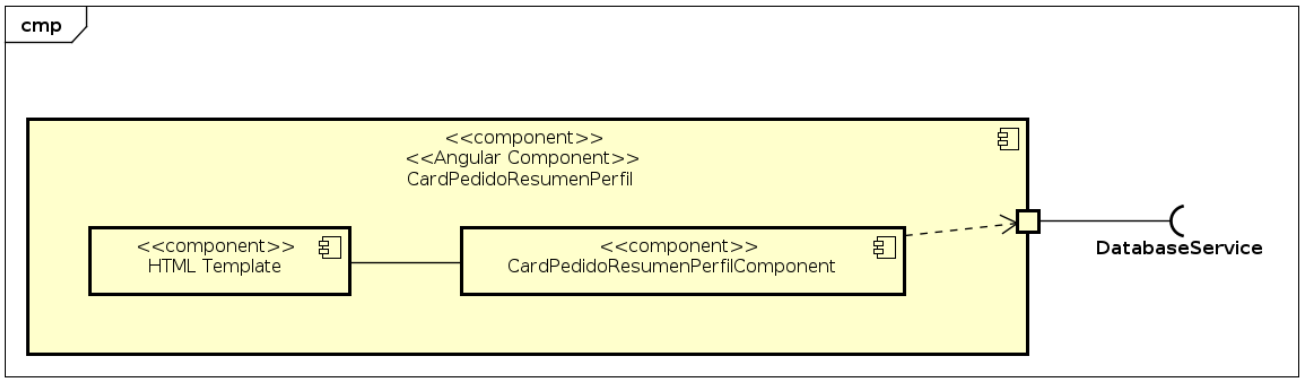


Figura 7.25: Componente CardPedidoResumenPerfil

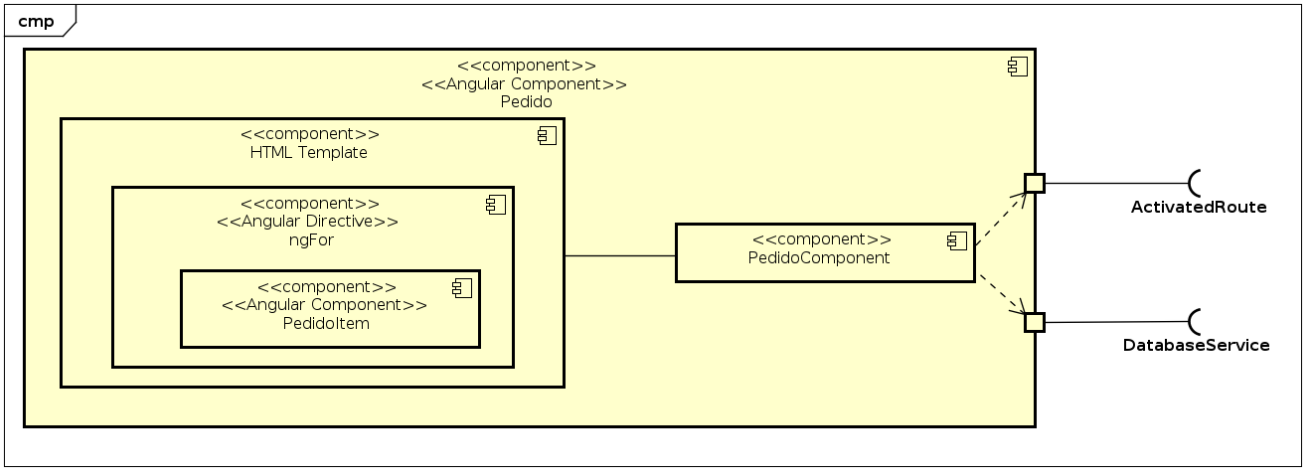


Figura 7.26: Componente Pedido

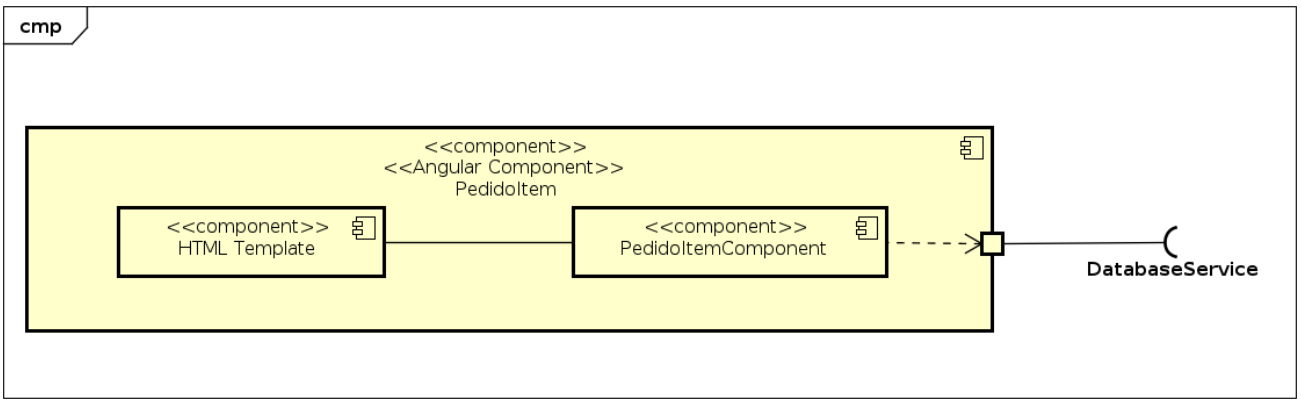


Figura 7.27: Componente PedidoItem

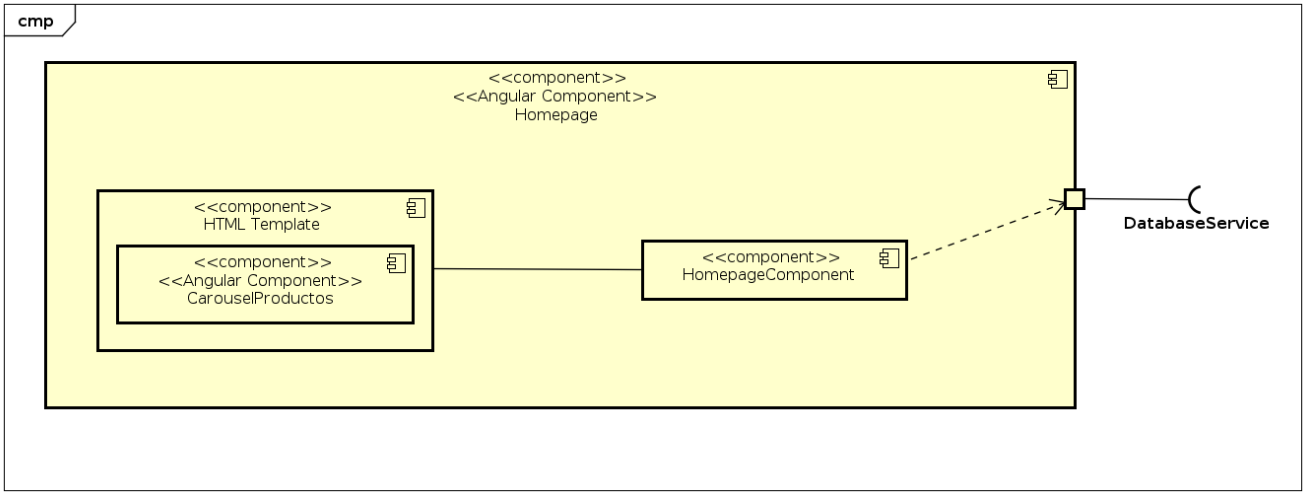


Figura 7.28: Componente Homepage

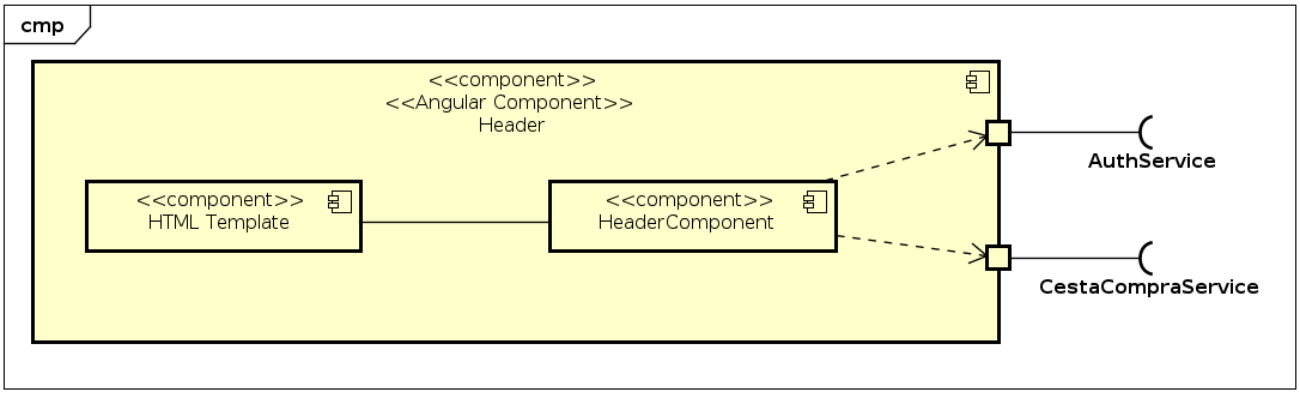


Figura 7.29: Componente Header

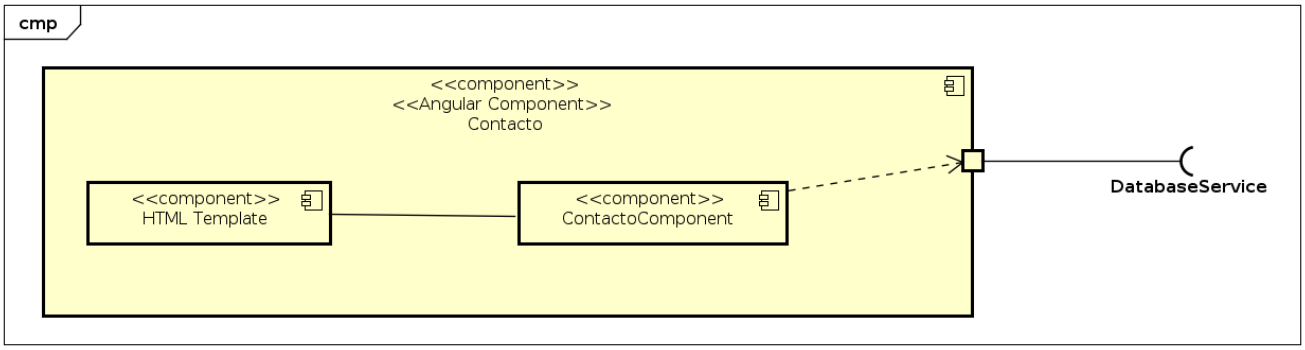


Figura 7.30: Componente Contacto

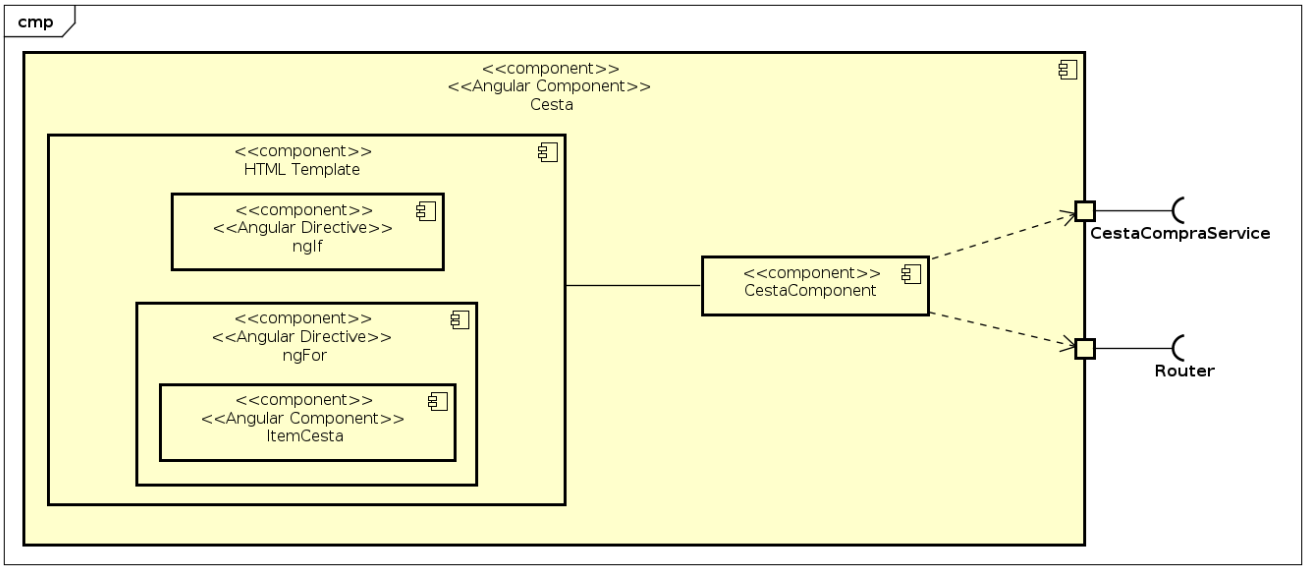


Figura 7.31: Componente Cesta

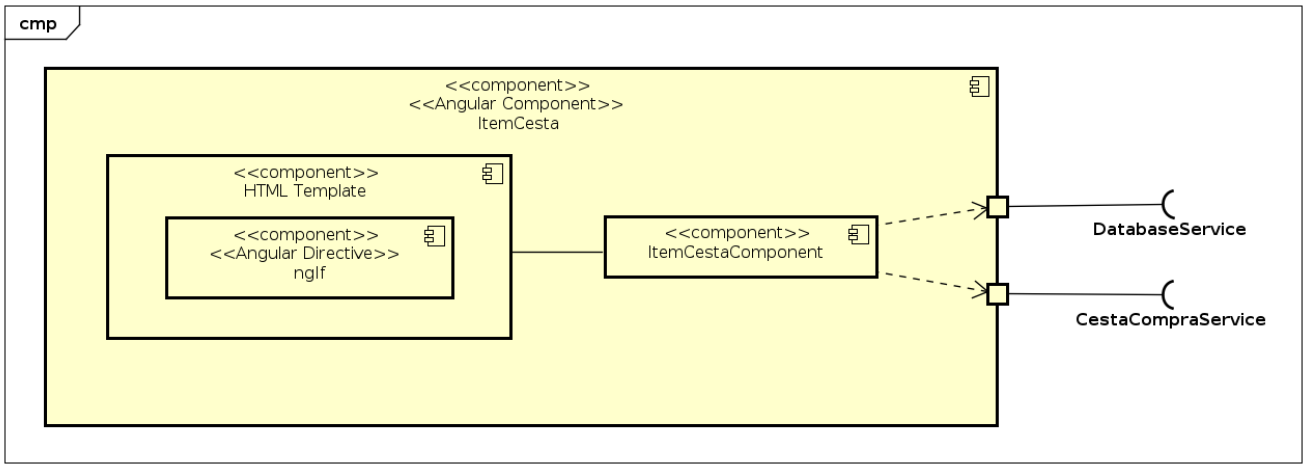


Figura 7.32: Componente ItemCesta

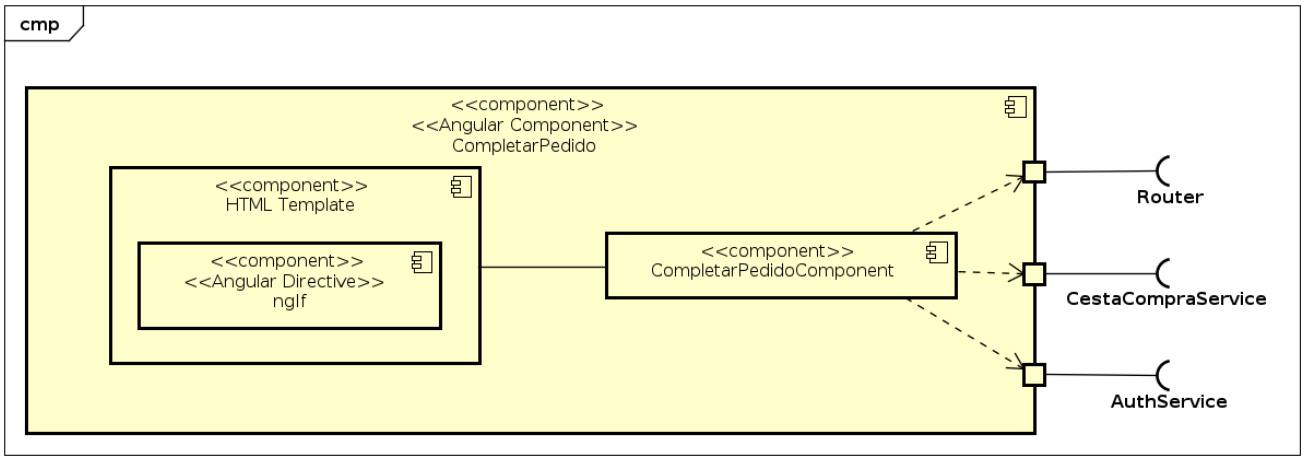


Figura 7.33: Componente CompletarPedido

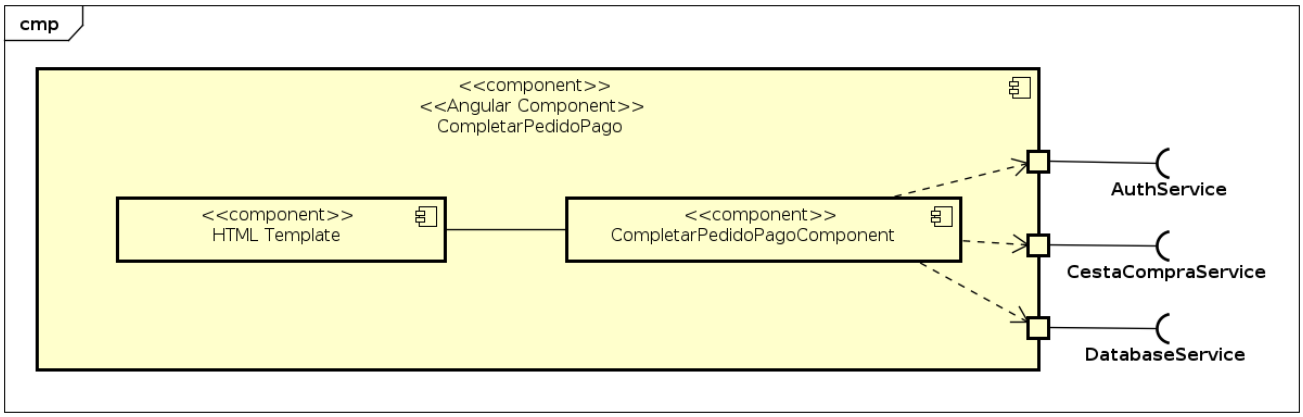


Figura 7.34: Componente CompletarPedidoPago

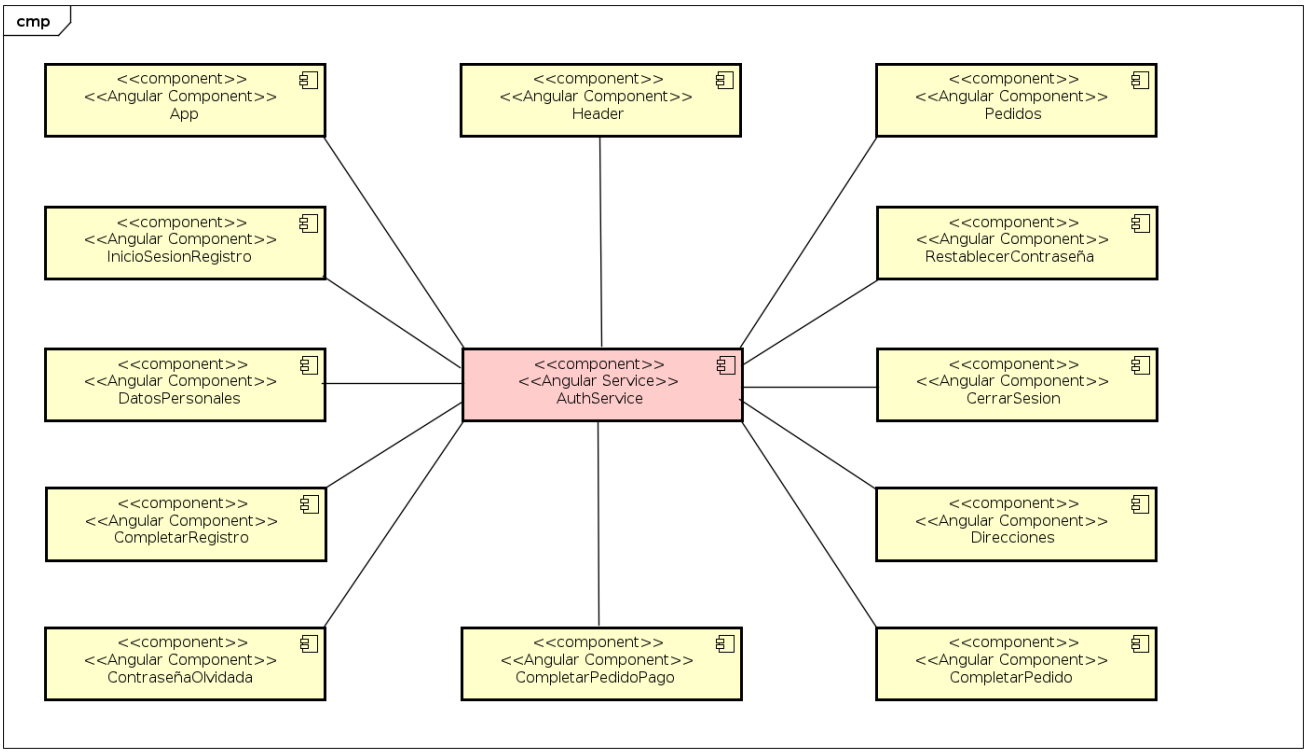


Figura 7.35: Servicio AuthService

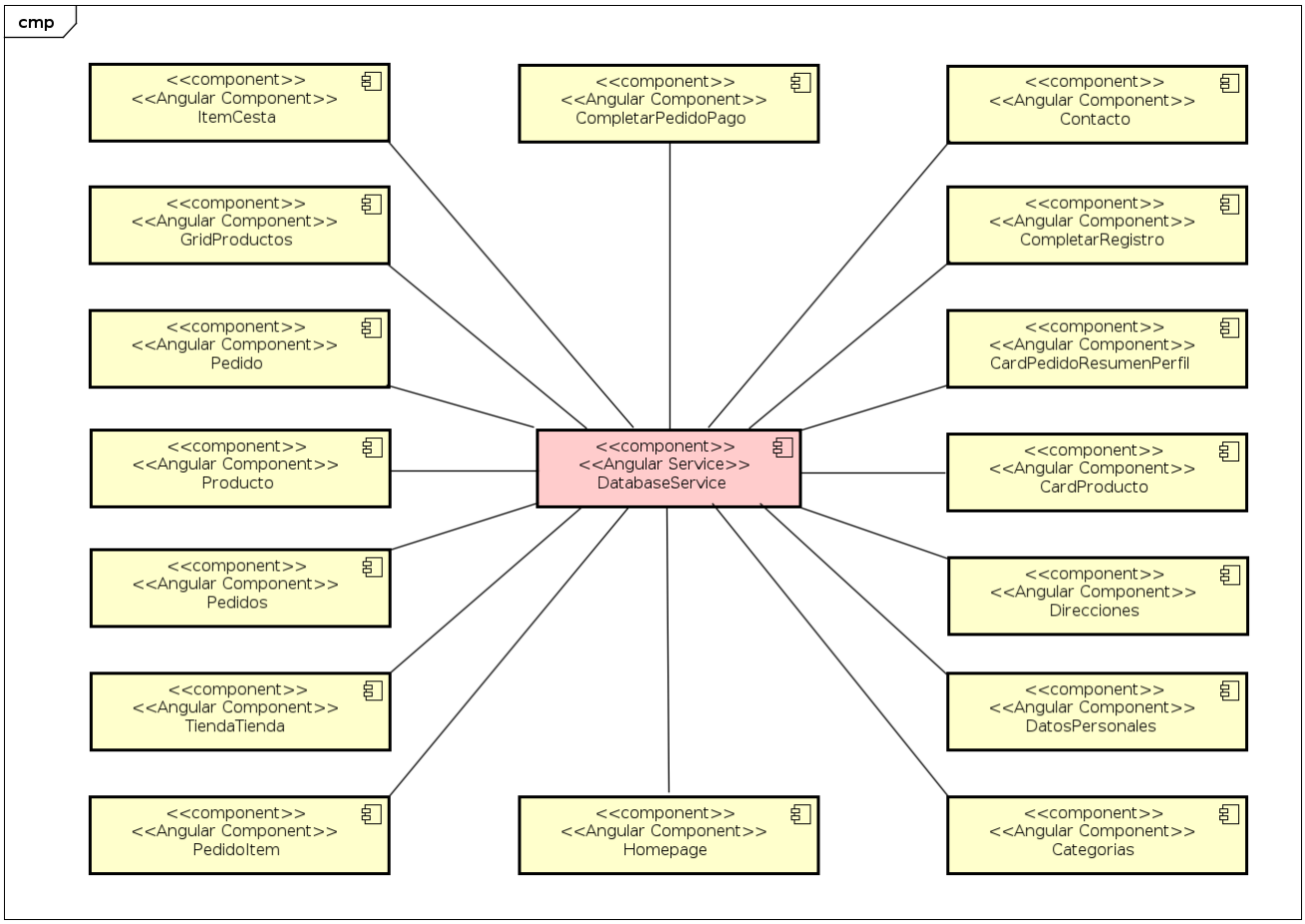


Figura 7.36: Servicio DatabaseService

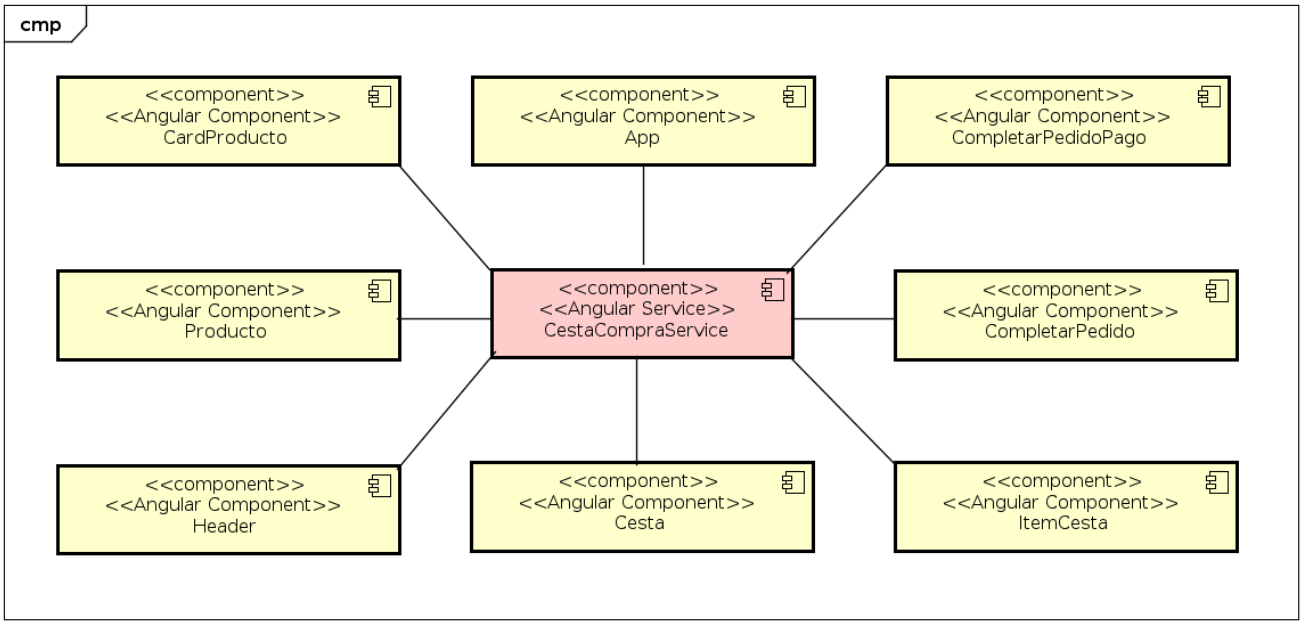


Figura 7.37: Servicio CestaCompraService

7.3. Diagrama de clases

Habiendo visto la estructura de la aplicación en tiempo de ejecución mediante componentes y la comunicación entre los mismos, queda detallar las clases que realizan en tiempo de compilación las interfaces y funcionalidad de los componentes representados.

No se representa la clase del componente "`app.module`" ya que su único cometido es de declarar todos los componentes de la aplicación en sus metadatos y de realizar las importaciones de módulos necesarias para que la aplicación funcione correctamente.

Las clases creadas son:

- `AppComponent`: contiene el título de la aplicación y realiza llamadas en su inicio a el *AuthService* y el *DatabaseService* para intentar recuperar el estado de inicio de sesión y la cesta que poseía el usuario en memoria local. (Figura 7.38)

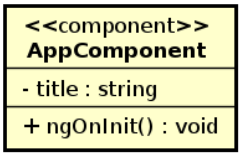


Figura 7.38: AppComponent class

- `TiendaComponent`: no posee lógica interna dado que su función es la de incluir en su *template* un router-outlet para incorporar los distintos componentes que componen la tienda. (Figura 7.39)

Figura 7.39: TiendaComponent class

- `TiendaTiendaComponent`: se encarga de recuperar todos los productos de base de datos por medio del *DatabaseService* para enviarlos a el grid de productos que contiene como componente. (Figura 7.40)

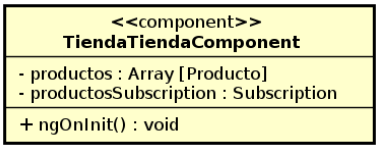


Figura 7.40: TiendaTiendaComponent class

- `CardProductoComponent`: encargada de la lógica de mostrar un producto de forma abreviada. Posee un atributo de tipo Producto, que es el producto que va a ser mostrado. El atributo imagen es un observable que devuelve una referencia en forma de `string` a la imagen que se obtiene del módulo `Storage` por medio del *DatabaseService*. Posee un método para añadir el producto a la cesta cuando se pulsa el botón que contiene este componente. (Figura 7.41)

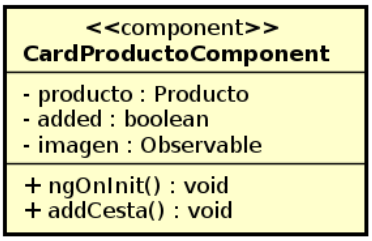


Figura 7.41: CardProductoComponent class

- `CarouselProductosComponent`: posee un array de productos para poder crear desde el los múltiples componentes de tipo CardProdcucto que contendrá el carousel. Posee un atributo `responsiveOptions` que son las opciones que recibe el Carousel de PrimeNG para saber cómo comportarse en diferentes tamaños de pantalla. (Figura 7.42)

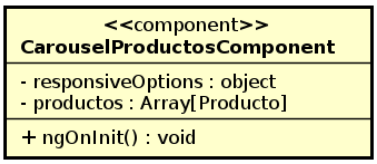


Figura 7.42: CarouselProductosComponent class

- `CategoriasComponent`: se encarga de cargar los nombres de las categorías presentes en la tienda a través del *DatabaseService* y de mostrar de manera dinámica botones para volver o mostrar todas las categorías dependiendo de en qué componente se encuentre añadida. (Figura 7.43)

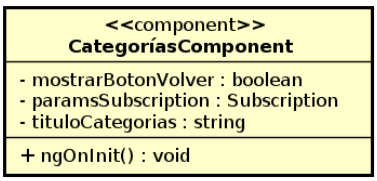


Figura 7.43: CategoriasComponent class

- `GridProductosComponent`: se encarga de enviar al *template* los productos que ha cargado de una categoría o de enviarle los productos que recibe como entrada. Interactúa de forma activa con el *DatabaseService* para llevar a cabo sus tareas. (Figura 7.44)
- `ProductoComponent`: se encarga de cargar un producto desde base de datos, por medio de una llamada al *DatabaseService*, y mostrar sus propiedades en el *template* mediante un binding de datos. También posee la lógica para añadir la cantidad de producto deseada a la cesta por medio del CestaCompraService. (Figura 7.45)
- `SobreNosotrosComponent`: no posee lógica ni atributos, dado que su único cometido es mostrar la vista `Sobre nosotros`, la cual solo muestra texto e imágenes. (Figura 7.46)
- `InicioSesionRegistroComponent`: posee la lógica par que el usuario pueda iniciar sesión o iniciar el proceso de registro. Posee también lógica para saber cuando se está procesando una solicitud de inicio de sesión y poder mostrar de forma dinámica un "`spinner`"para indicar al usuario que está cargando. Se comunica activamente con el *AuthService* para llevar a cabo esta acción. (Figura 7.47)

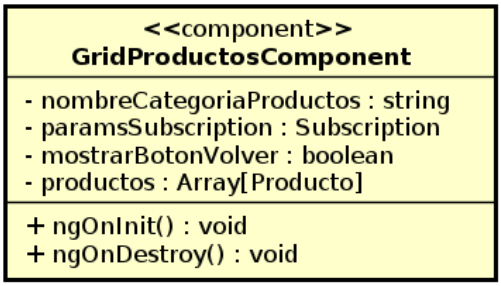


Figura 7.44: GridProductosComponent class

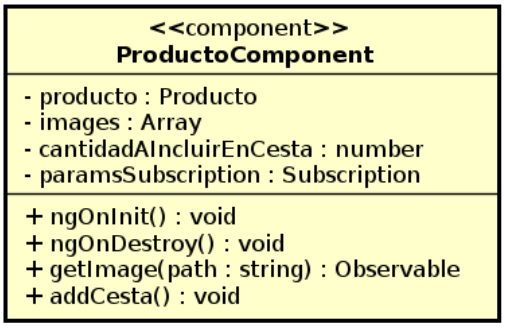


Figura 7.45: ProductoComponent class

- `CompletarRegistroComponent`: contiene la lógica para recuperar los datos introducidos en el formulario de registro por el usuario y hacer una llamada a los servicios *AuthService* y *DatabaseService* para que den de alta al usuario en la aplicación. (Figura 7.48)

- `ContrasenaOlvidadaComponent`: posee la lógica para hacer una llamada al *AuthService* con los datos introducidos por el usuario para recuperar su contraseña. (Figura 7.49)

- `RestablecerPasswordComponent`: Posee la lógica para recuperar los datos introducidos por el usuario en el formulario presente en su *template* asociado. Posee lógica para recuperar el parámetro `oobCode` de la url con la que se accede a este componente, el cuál es necesario para poder hacer la llamada al módulo Auth de Firebase a través del *AuthService* para cambiar la contraseña. Si en los parámetros no existe este código, se redirige al usuario a la página de inicio, ya que no se podría realizar la acción sin este parámetro. (Figura 7.50)

- `PerfilCuentaComponent`: no posee lógica asociada, dado que su único objetivo es el de contener una directiva una columna de navegación y un router-outlet en su *template* para poder mostrar los distintos componentes que se muestran en el perfil. (Figura 7.51)

- `CerrarSesionComponent`: posee una cuenta atrás que redirige al usuario a la página de inicio de sesión tras haber cerrado la sesión. Fue necesario incluir este componente para poder dar una retro-alimentación al usuario de que se estaba cerrando la sesión, ya que el cierre de esta era instantáneo. Interactúa activamente con el *AuthService* para llevar a cabo estas acciones. (Figura 7.52)

- `DatosPagoComponent`: no posee lógica debido a que esta funcionalidad no ha sido implementada finalmente en el proyecto, pero se deja incluido el componente vacío para así no tener que incluirlo en un futuro y que sea más rápido el desarrollo de esta funcionalidad. (Figura 7.53)

- `DireccionesComponent`: posee la lógica para mostrar en el *template* los datos asociados a las

Figura 7.46: SobreNosotrosComponent class

Figura 7.47: InicioSesionRegistroComponent class

direcciones de el usuario que ha iniciado sesión en la tienda. También posee la lógica asociada al cambio de las direcciones y retroalimentación para el usuario. Interactúa activamente con su *DatabaseService* y con el *AuthService* para llevar a cabo estas acciones. (Figura 7.54)

- `DatosPersonalesComponent`: posee la lógica necesaria para mostrar el correo electrónico y el teléfono del usuario que está actualmente con sesión iniciada en la tienda y la lógica que permite cambiarlos.Interactúa activamente con su *DatabaseService* para llevar a cabo estas acciones, también interactúa con el *AuthService*. (Figura 7.55)

- `PedidosComponent`: posee la lógica para recuperar todos los pedidos asociados a un usuario, ordenarlos por fecha y mostrarlos en el *template*. Interactúa activamente con su *DatabaseService* y el *AuthService* para llevar a cabo estas acciones. (Figura 7.56)

- `CardPedidoResumenComponent`: posee la lógica para recuperar un pedido de base de datos y de enviar los datos de este al *template* para mostrarlos al usuario de una forma resumida. También calcula la fecha del pedido a partir del `timestamp` almacenado en base de datos. Interactúa activamente con su *DatabaseService* para llevar a cabo estas acciones. (Figura 7.57)

- `PedidoComponent`: posee la lógica para recuperar un pedido de base de datos a partir de su número de pedido y mostrarlo en el *template*. También permite cancelar un pedido que se encuentre en un estado distinto a "Cancelado.º "Enviado". Interactúa con su *DatabaseService* para esta tarea. (Figura 7.58)

- `PedidoItemComponent`: se encarga de recuperar un producto de base de datos por medio del *DatabaseService* y de mostrarlo en el *template* junto con las unidades de ese producto que hay en el pedido. (Figura 7.59)

- `HomepageComponent`: contiene la lógica necesaria para recuperar de base de datos los productos que estén destacados, por medio de una llamada al *DatabaseService*, y enviárselos al Ccarousel que tiene dentro. (Figura 7.60)

- `HeaderComponent`: contiene la lógica para que la barra de navegación funcione y muestre la información de forma correcta. Interactúa con el *AuthService* para saber si hay usuario con sesión iniciada y con el CestaCompraSErvice para ir actualizando el número que muestra el total de productos en la cesta. (Figura 7.61)

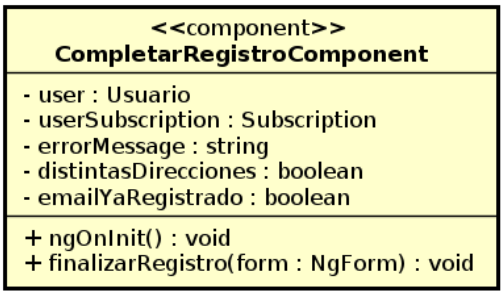


Figura 7.48: CompletarRegistroComponent class

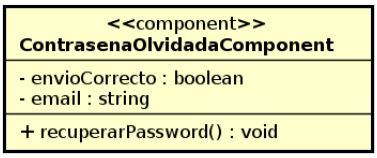


Figura 7.49: ContrasenaOlvidadaComponent class

- `ContactoComponent`: contiene la lógica para enviar la solicitud de contacto a la base de datos por medio de una llamada su *DatabaseService*. (Figura 7.62)

- `CestaComponent`: contiene la lógica necesaria para recuperar los productos que hay actualmente en la cesta de la compra y mostrarlos en el *template*. Interactúa con el CestaCompraService para poder llevar a cabo esta tarea. También permite iniciar la creación de un pedido. (Figura 7.63)

- `ItemCestaComponent`: se encarga de recuperar los datos de un producto en base de datos por medio de una llamada al *DatabaseService* y de mostrar los datos de este producto, junto con las unidades que hay en la cesta, botones para eliminarlo o modificar su cantidad y redirigir al producto al usuario. (Figura 7.64)

- `CompletarPedidoComponent`: se encarga de mostrar el contener la lógica asociada al formulario de realizar un pedido, el cual contiene la dirección en la que será entregado el pedido, la cuál es cargada de los datos del usuario que ha iniciado sesión. En caso de que no haya un usuario con sesión iniciada, su lógica se encarga de notificar al *template* para que muestre un mensaje de error y pida al usuario iniciar sesión. Se comunica con el *AuthService* y el CestaCompraService para llevar a cabo estas funciones. (Figura 7.65)

- `CompletarPedidoPagoComponent`: contiene la lógica necesaria para procesar los datos introducidos por el usuario en el formulario que pide la dirección a la que será facturado el pedido. Es el ultimo paso de la realización de un pedido. Una vez el usuario con sesión iniciada ha terminado de introducir los datos se encarga de mandar una petición al *DatabaseService* para crear un pedido para el usuario. Interactúa con el *AuthService* para saber quién ha iniciado sesión y con el CestaCompraService para actualizar la dirección del pedido que se va a crear en base de datos y vaciar las cestas local y remota. (Figura 7.66)

- `AuthService class`: contiene la lógica del servicio que se encarga de todo lo referente a la autenticación del usuario. También se encarga de dar de alta nuevos usuarios y de recuperar sus datos de base de datos mediante una llamada al *DatabaseService* al iniciar sesión en la aplicación. Se encarga de mantener continuamente actualizados los datos del usuario que tiene la sesión iniciada. Todos los componentes que hacen uso de el AuthService se conectan a la misma instancia de este, pues solo

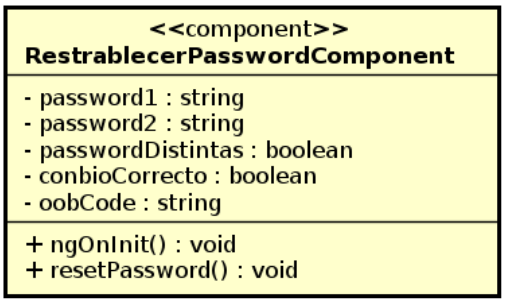


Figura 7.50: RestablecerPasswordComponent class

Figura 7.51: PerfilCuentaComponent class

existe uno en la aplicación, por tanto el AuthService sigue un patrón Singleton.
(Figura 7.67)

- `DatabaseService class`: contiene la lógica del servicio que se encarga de todo lo referente a recuperación y colocación de datos en base de datos. Posee métodos para llevar a cabo sus tareas de forma correcta. Este servicio no está pensado para seguir un patrón Singleton, como sí lo están el servicio Auth y CestaCompra, por ende, cuando este servicio es inyectado como dependencia dentro de algún componente lo hace con una instancia propia para el componente que ha solicitado la inyección. Esto se hace porque en el caso de carga múltiple de datos, al tener cada componente su propio servicio, no tendrá que esperar a que se finalice el envío de otro componente para hacer la suya, sino que se harán todas de forma paralela, cada componente en su propio servicio.(Figura 7.68)

- `CestaCompraService class`: clase que contiene la lógica de el servicio que se encarga de lo referente a mantener la cesta actualizada e informar a los componentes que necesitan saber los datos que hay en la cesta. Este servicio sigue un patrón Singleton, al igual que el *AuthService*. Cada vez que se realiza un cambio en la cesta, si hay un cliente con sesión iniciada se encarga de hacer una llamada al *DatabaseService* para actualizar en la base de datos la cesta del cliente que ha iniciado sesión Interactúa con el *AuthService* para tener actualizados en todo momento los datos del usuario autenticado y para poder recuperar su identificador para poder volcar los cambios en la cesta de forma correcta en la base de datos. (Figura 7.69)

- `AuthGuard class`: este guard se encarga de restringir el acceso a las vistas de el perfil de un usuario. Si hay un usuario autenticado en la aplicación, permitirá acceder a las vistas del perfil, en caso contrario redirigirá al usuario a la vista de inicio de sesión y registro para que el usuario inicie sesión o se registre, desbloqueando de esta forma las vistas a las que intentaba acceder.
(Figura 7.70)

- `PedidoGuard class`: este *guard* se encarga de restringir el acceso de un usuario a la página de pagar un pedido si no ha iniciado el proceso de hacer un pedido.
(Figura 7.71)

Figura 7.52: CerrarSesionComponent class

Figura 7.53: DatosPagoComponent class

7.3.1. Modelo de datos

Para facilitar la tarea de trabajar con la base de datos al recuperar y enviar datos se han creado clases TypeScript que representan los tipos de datos que eran necesarios para esta aplicación y que permiten mapear en el cliente los objetos que se alojan en la base de datos.

Las clases de tipos creadas son las siguientes:

- `Usuario`: esta clase representa a un usuario de nuestra aplicación. Solo posee atributos y un constructor que inicializa los valores a 0 o vacío. El usuario posee un atributo "`cesta`.[el] cuál contiene pares productoID-unidades los cuales representan un producto que el usuario tiene añadido a la cesta y las unidades de ese producto en esta. Se decide incluir únicamente el identificador del producto por no repetir datos en base de datos ni en la aplicación.
  (Figura 7.72)
- `Cesta`: esta clase representa una cesta de la compra para usar en la aplicación. La cesta contiene un array de pares unidades-Producto los cuales son las unidades de un producto que hay añadido en la cesta y el Producto en sí. (Figura 7.73)
- `Producto`: esta clase representa un producto, con todos los datos asociados a este y la ubicación dentro del módulo Storage de las imágenes. (Figura 7.74)
- `Pedido`: esta clase representa un pedido, posee los datos de donde entregarlo, donde facturarlo, a nombre de quién se realiza y los productos y unidades que contiene. (Figura 7.75)
- `MetodoPago`: clase que representará en un futuro uno de los métodos de pago de un usuario registrado en nuestra aplicación. Actualmente sin atributos, dado que no ha sido implementados los pagos en el aplicativo. (Figura 7.76)

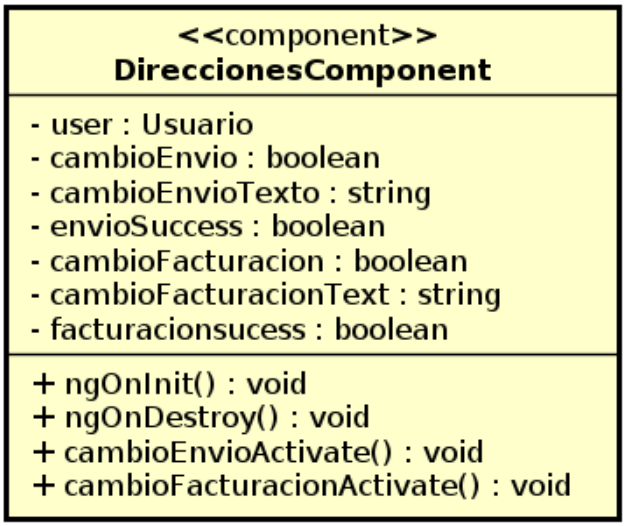


Figura 7.54: DireccionesComponent class

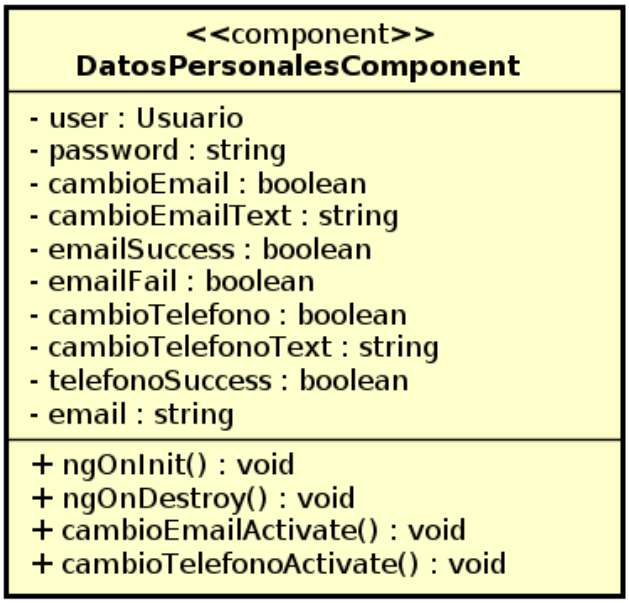


Figura 7.55: DatosPersonalesComponent class

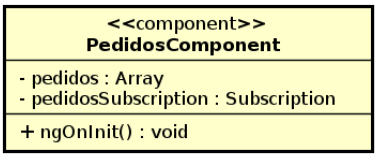


Figura 7.56: PedidosComponent class

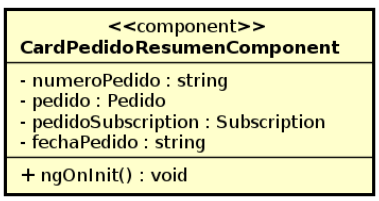


Figura 7.57: CardPedidoResumenComponent class

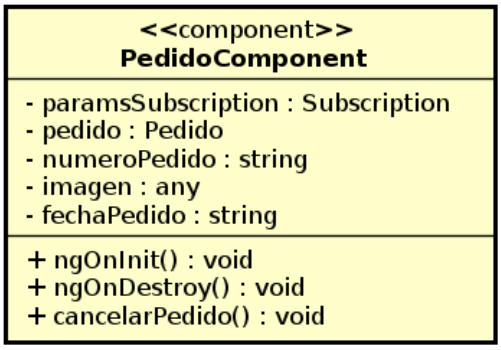


Figura 7.58: PedidoComponent class

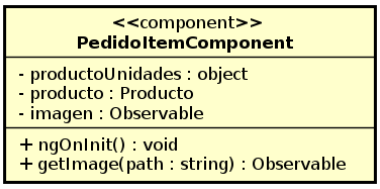


Figura 7.59: PedidoItemComponent class

Figura 7.60: HomepageComponent class

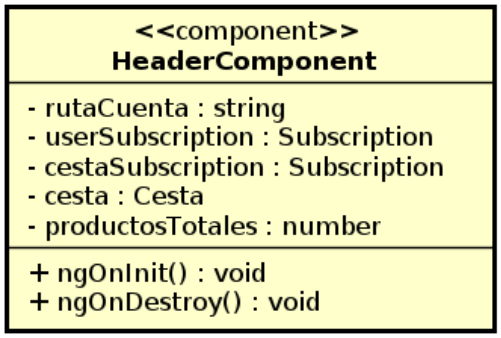


Figura 7.61: HeaderComponent class

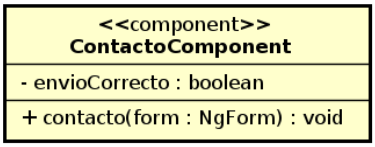


Figura 7.62: ContactoComponent class

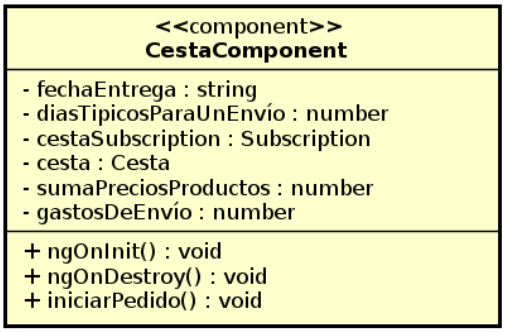


Figura 7.63: CestaComponent class

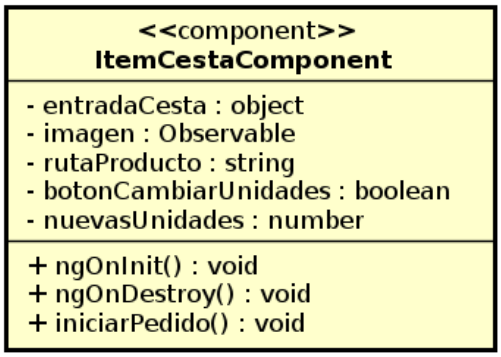
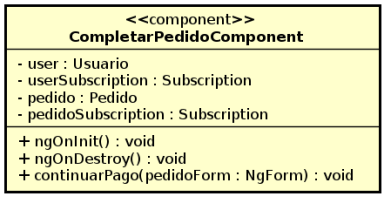


Figura 7.64: ItemCestaComponent class



5

Figura 7.65: CompletarPedidoComponent class

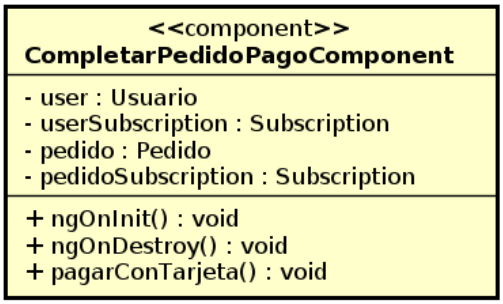


Figura 7.66: CompletarPedidoPagoComponent class

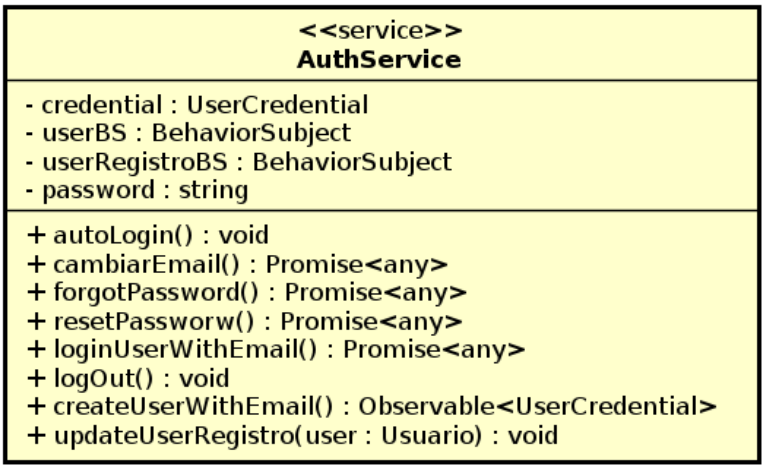


Figura 7.67: AuthService class

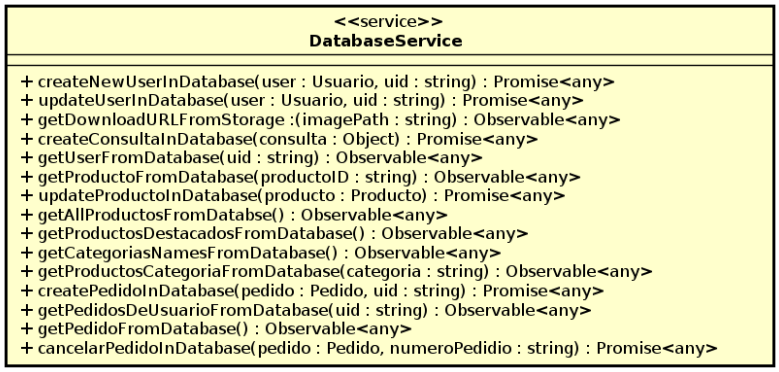


Figura 7.68: DatabseService class

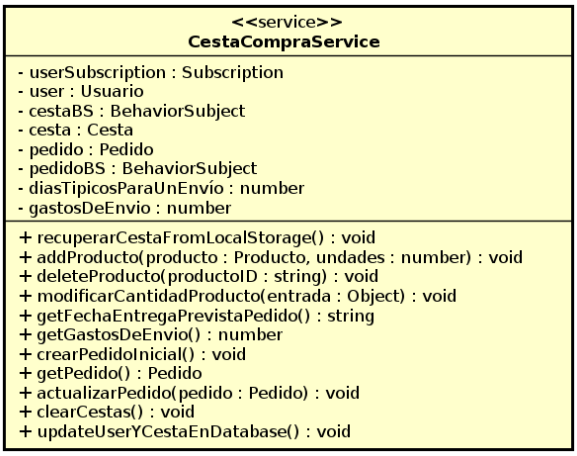


Figura 7.69: CestaCompraService class

Figura 7.70: AuthGuard class

Figura 7.71: PedidoGuard class

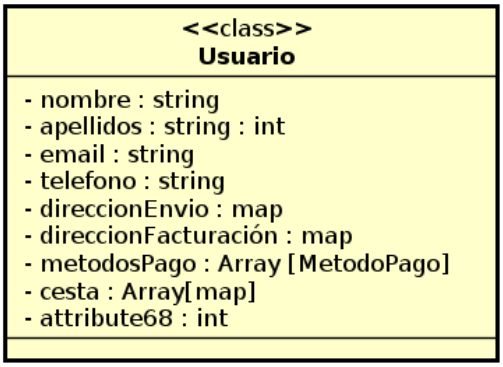


Figura 7.72: Usuario class

Figura 7.73: Cesta class

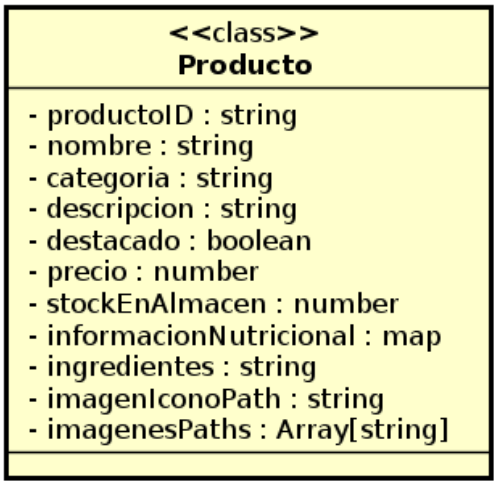


Figura 7.74: Producto class

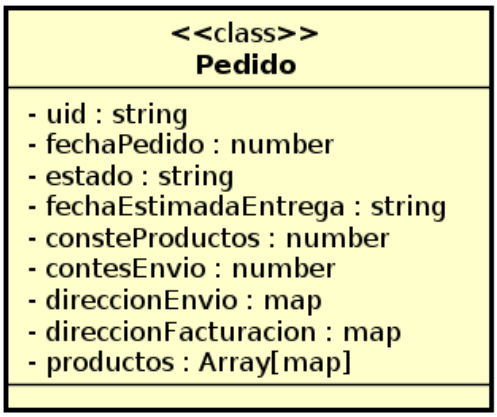


Figura 7.75: PedidoClass class

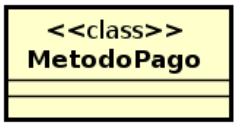


Figura 7.76: MetodoPago class

7.4. Diseño de la base de datos

7.4.1. Diseño conceptual

Para el proyecto se emplea la base de datos Firestore de Firebase, la cuál sigue un modelo de base de datos no relacional. Antes de realizar el diseño de la base de datos en Firebase se creó un diseño conceptual entidad-relación de como debía ser la base de datos que guardase los datos de nuestra aplicación.

Esta conceptualización se puede ver en la Figura 7.77.

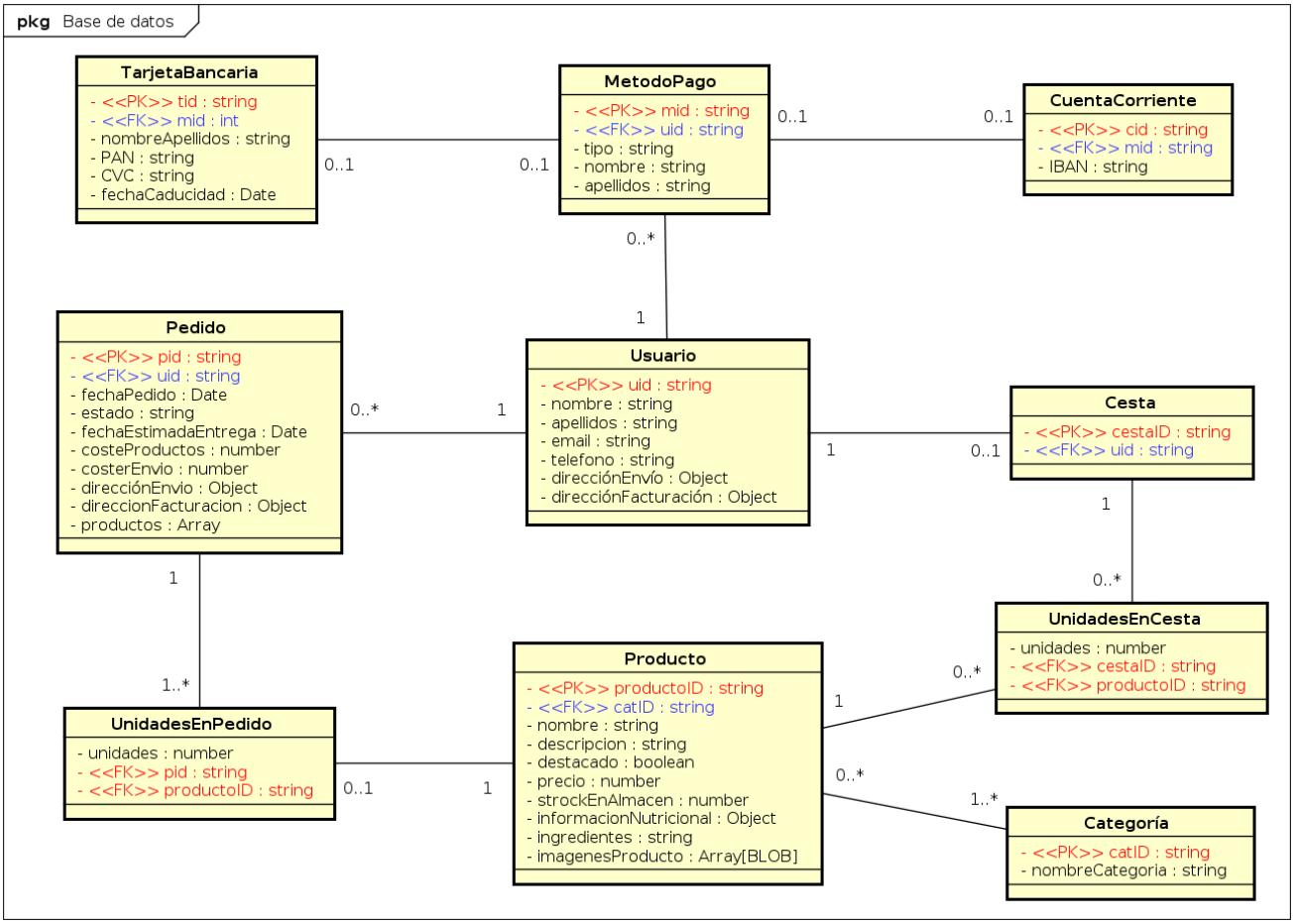


Figura 7.77: Diseño conceptual de base de datos

7.4.2. Diseño Firestore

Como ya hemos explicado, Firestore no sigue un modelo de base de datos relacional. Los datos en Firestore se introducen dentro de Documentos, los cuales son ficheros con formato JSON, que son agrupados en Colecciones. Un documento puede a su vez contener dentro una colección (sub-colección) lo cuál permite crear una estructura de datos jerárquica.

Al no seguir un modelo relacional, el diseño conceptual se vio modificado para poder adaptarlo a los requisitos del modelo NoSQL de Firestore.

Firestore asigna, si lo deseamos, un Identificador único y calculado de forma automática, a cada uno de los documentos que introducimos en una colección. Por lo tanto, la mayoría de claves primarias asociadas

a las entidades del diseño conceptual pudieron ser eliminadas. Al no seguir un modelo relacional, las claves foráneas de nuestro modelo conceptual pasan a ser meros identificadores de aquello a lo que hacían referencia, no exigiendo de esta forma la existencia de los elementos a los que se hace referencia.

Durante el transcurso del desarrollo de la implementación, se detectó la necesidad de almacenar las consultas enviadas por los usuarios en base de datos, por lo tanto se creó una colección de Firestore para ello.

La estructura de Firestore se puede ver en la Figura 7.78

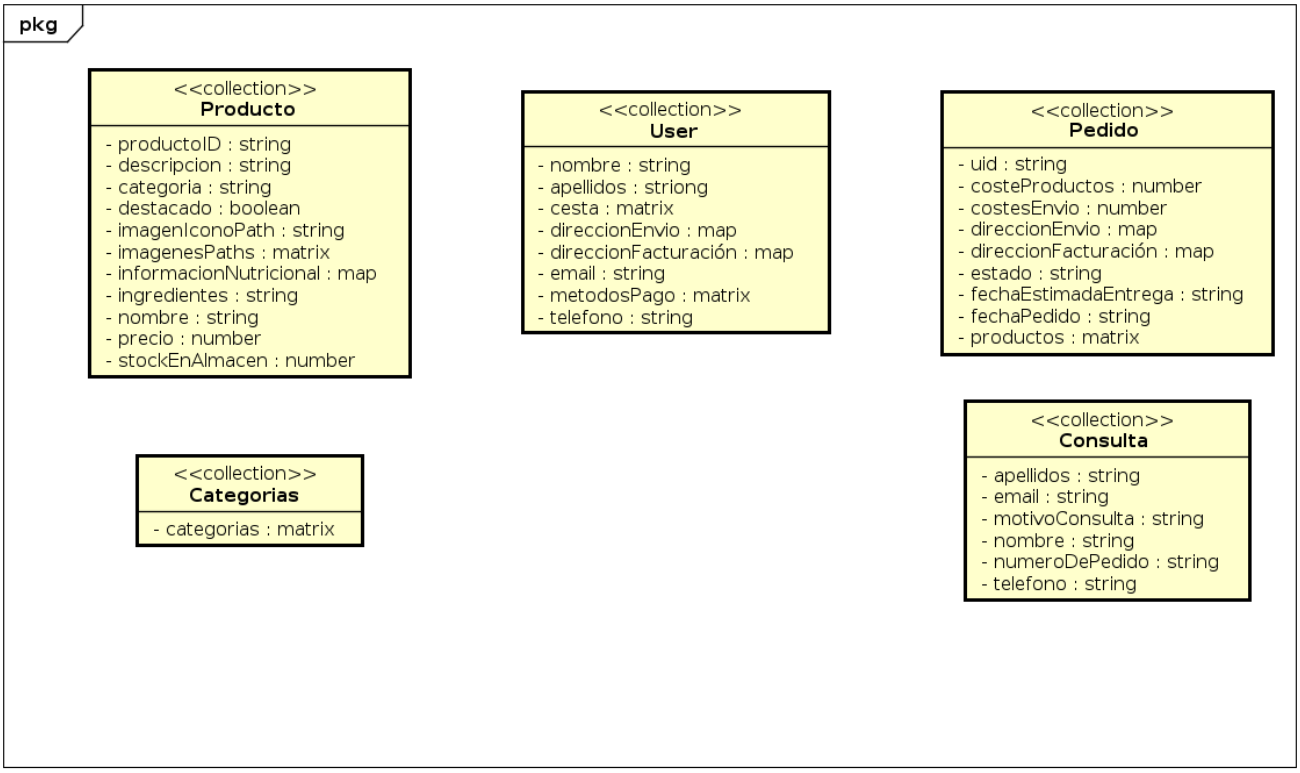


Figura 7.78: Diseño Firestore

7.5. Despliegue

La aplicación ha sido desplegada en el servicio de Hosting que posee Firebase. Por este motivo, al no haber diseñado desde cero el back-end, se proporciona un diagrama descriptivo (Figura 7.79) de como se despliegan los distintos módulos de Firebase y las relaciones diseñadas entre ellos que existirán cuando sean desplegados. .

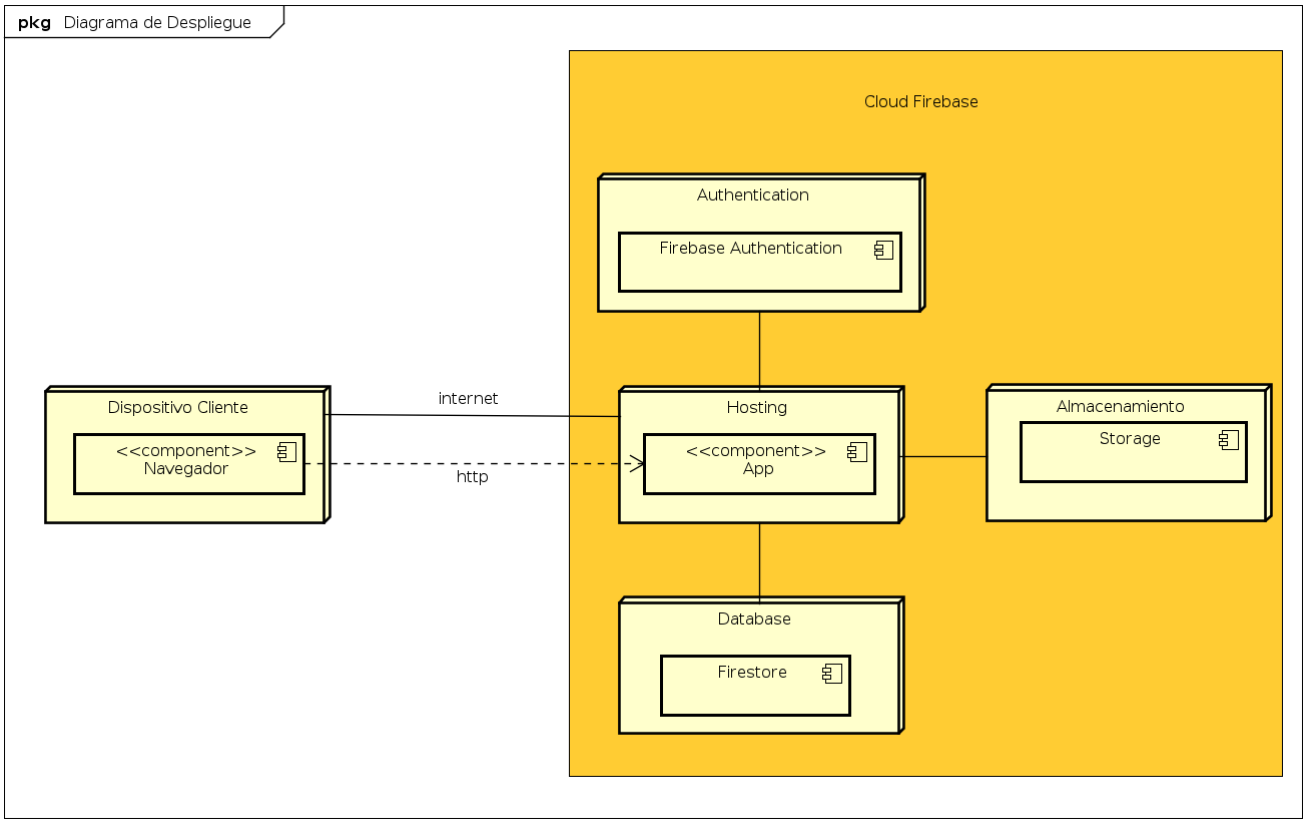


Figura 7.79: Diagrama de Despliegue

7.6. Interfaz

La interfaz de usuario de la aplicación es una parte muy importante de la aplicación, ya que debe reflejar la identidad de marca de la empresa, ser fácil de utilizar por los clientes y estar distribuida de una forma eficiente. Es por ello que en el diseño de esta interfaz se ha trabajado en estrecho contacto con los futuros propietarios del sistema para personalizar esta interfaz a su gusto, basándonos en alternativas ya presentes en el mercado pero dando la identidad propia de la marca y reflejando en la interfaz los deseos de los clientes del sistema.

7.6.1. Paleta de colores

Lo primero que se acordó con el cliente, fue la paleta de colores a emplear, dado que así se podría diseñar la interfaz final con los colores finales, asegurándonos así de que el resultado final sea como el cliente desea.

El cliente quería que, dado que es una tienda de productos veganos, el color principal fuera un verde, llamativo, pero no demasiado. También quería combinar ese color verde con un fucsia, dado que según su criterio eran dos colores que contrastaban y combinaban muy bien. Desde nuestro lado, le expresamos que era complicado introducir ese color contrastándolo con el verde, dado que se hacía muy complicada la lectura de los textos si estos eran en fucsia sobre verde. Tras varios intentos, se decidió no emplear el fucsia para los textos, pero sí que se emplearía como color secundario de acento para algunos elementos de la interfaz. De esta forma conseguimos integrar los colores que el cliente quería para su tienda y se pudo empezar el boceto de la interfaz.

La paleta de colores consta de 5 elementos en total, el verde y el fucsia elegidos por cliente, el blanco y negro como colores indispensables para textos y detalles, y un gris para marcar elementos de forma especial. La paleta está expuesta en la Figura 7.80

Se acuerda con los clientes emplear el color fucsia para todo lo referente al proceso de compra, como añadir un producto a la cesta, iniciar el pago, o ver el número de elementos en la cesta de la compra.

Figura 7.80: Paleta de colores del sitio web

7.6.2. Diagrama de navegación inicial del sitio web

Se crea un diagrama de navegación (Figura 7.81) web inicial para saber cómo se navegará a través del sitio web una vez esté en funcionamiento.

Para facilitar la navegación del sitio, se acuerda la instalación de una barra de navegación fija en la parte superior de la web para que el cliente sepa en todo momento en que sección la aplicación se encuentra. Esta barra también poseerá un icono que representará la cesta de la compra, e irá contabilizando los elementos añadidos por el cliente a esta.

Desde la barra de navegación se tendrá acceso a múltiples secciones y vistas de la web. También se podrá acceder a la *"landing page"* que es la primera página que se muestra al llegar al sitio web. Está página será el punto de entrada para nuevos usuarios a nuestra aplicación.

7.6.3. Bocetos de la interfaz cliente

En esta parte se detallan los bocetos creados de la tienda on-line. Los bocetos han sido creados basándose en otras tiendas online líderes de sus respectivos sectores. Estas tiendas han sido explicadas en la sección 1.4 .

Pagina principal

Esta página es la primera que se muestra al llegar al sitio web. Cuenta con varias franjas principales en las que se expondrán productos destacados (las franjas negras) y cada producto contará con un botón en fucsia para añadirlo a la cesta de la compra. Más abajo, incluirá un pequeño texto sobre la empresa y más productos. (Figura 7.82)

Vista principal de la tienda

Si el usuario selecciona en la barra superior la sección tienda, o pulsa en el botón que habrá en la página principal para direccionar a la tienda, se llegará a esta vista. (Figura 7.83)

En esta vista se presentarán primero una serie de categorías al usuario para que seleccione una de ellas.

No se presentarán todas las categorías en esta sección, ya que se habilitará uno de los los botones de esta zona para redirigir al usuario a una vista específica que contendrá todas las categorías de productos de la tienda.

Justo después de esta parte en la que se presentan las categorías se presentaran algunos productos destacados al cliente. Todos y cada uno de los productos tendrán la opción de ser añadidos a la cesta mediante un botón en fucsia específico para ello o de redirigir al usuario a la vista del producto en cuestión. Para esto ultimo habrá que hacer click en cualquier parte del elemento del producto que no sea el botón para añadirlo a la cesta.

Vista de categorías de productos

En esta página estarán expuestas todas las categorías de la tienda y el usuario podrá seleccionar una de las categorías para ver los productos relacionados con esa categoría en una vista específica para ello. (Figura 7.84)
El usuario será redirigido a la vista de los productos de la categoría si pulsa en el botón *"Productos"* de alguno de los elementos o si pulsa el elemento en cuestión.

Vista de producto

A esta página se llega desde varios sitios de la web, los cuales están expuestos en el diagrama de navegación de la sección 7.6.2. La vista se encuentra expuesta en la Figura 7.85

La vista contará con una barra de links tipo *"miga de pan"* para que el usuario sepa en que parte de la tienda se encuentra y pueda volver sobre sus pasos y a la categoría del producto de una forma rápida e intuitiva.

La vista tendrá como elemento principal un carrusel de imágenes del producto que se presenta en ella, siendo posible ver distintas fotos del producto haciendo click en las flechas laterales de este elemento. Tendrá el nombre del producto expuesto en grande, y justo debajo de él, el precio de venta al público del producto. Poseerá una pequeña descripción a modo resumen del producto. Existirá un elemento que muestre la disponibilidad o no de stock en los almacenes del producto. Debajo de estos elementos habrá un *"spinner"* que permitirá modificar la cantidad de producto que se desean añadir a la cesta de la compra. Justo al lado de este elemento se encontrará en botón para añadir la cantidad seleccionada de producto a la cesta de la compra.

En la parte inferior habrá una sección, cambiante mediante una barra de links o una *"wedgebar"*, que expondrá la información nutricional del producto, los componentes alérgenos de este, los ingredientes y las valoraciones del producto.

Por cuestión de tiempo no se ha podido incluir esta última parte en la aplicación realizada, pero se deja en los bocetos iniciales a modo de una vía futura de mejora del sistema.

Vista de inicio de sesión y registro

Al hacer click en la barra superior de navegación en "Mi cuenta"se mostrará esta vista de iniciar sesión (Figura 7.86), si el usuario no tiene una sesión inicada. También se mostrará esta vista cuando el usuario no haya iniciado sesión, pero desee hacer un pedido.

Esta vista cuenta son dos formularios, uno para iniciar sesión compuesto de usuario, contraseña y un botón para realizar la acción.Esta forma de inicio de sesión será la inicial, principal y preferida del sistema.

Adicionalmente, este formulario contará con dos botones para poder iniciar el proceso de inicio de sesión mediante proveedores de autenticación (como Google o Facebook en este boceto).

El formulario contará con un enlace para poder recuperar la contraseña..

El formulario de registro solo cuenta con campos para introducir nombre, apellidos y un correo electrónico.

Una vez el usuario haya introducido estos datos y haga click en *Registrarse* será redirigido a las vistas para completar el registro, en las cuales se le requerirán es resto de datos personales.

Vista de completar registro 1

A esta vista (Figura 7.87) el usuario llegará tras iniciar el proceso de registro en la vista de inicio de sesión y registro. Los campos de nombre, apellidos y correo electrónico serán auto-completados con los datos que el usuario introdujo en la anterior vista, pero serán editables para que tenga oportunidad de corregirlos en caso de haber cometido algún error.

La vista contará con dos campos para introducir una contraseña y repetirla para asegurarse de que es la misma, contará con un campo para introducir el número de teléfono y en la parte superior, debajo de la barra de navegación contará con una serie de puntos a modo de miga de pan para que el usuario sepa los pasos que le quedan para completar el registro.

En la barra de navegación quedará marcada la opción de *Mi cuenta* para indicar al usuario que está haciendo gestiones con su cuenta.

Vista de completar registro 2

Esta vista (Figura 7.88) será continuación de la *Vista de completar registro 1*, Figura 7.87 y se llegará a ella pulsando el botón *"Siguiente"* de la vista mencionada. Contará con campos para que el usuario introduzca su dirección de envío habitual. Tendrá también un *RadioButtonGroup* para poder seleccionar si la dirección de facturación va a ser la misma que la de envío habitual. En el caso de que el usuario decida *"Utilizar otra dirección para la facturación"* aparecerá otro formulario debajo para que el usuario pueda introducir su dirección de facturación.

Cuando el usuario pulse sobre *"Finalizar Registro"* se enviarán los datos de registro y se dará por finalizado el registro.

Tras esto el usuario será redirigido a la vista de *"Pagina principal"*.

Vista Mi Cuenta

A esta vista (Figura 7.89) se accederá iniciando sesión, tras el registro, haciendo click en la sección correspondiente en la barra de navegación o tras el resumen de un pedido realizado.

En la parte superior aparecerá en todo momento el nombre y los apellidos del usuario.

La vista cuenta con una columna lateral para navegar en la vista que muestra distintos elementos en la parte derecha de la vista dependiendo de la sección que tengamos seleccionada. En el boceto de ejemplo se

muestran los pedidos que ha realizado el usuario, pudiendo seleccionar alguno para acceder a los detalles de este.

En la parte superior izquierda cuenta con una *"Miga de Pan"* para que, a parte de por la columna de navegación izquierda, el usuario sepa en todo momento en que parte de la aplicación se encuentra y pueda volver sobre sus pasos de forma simple. Esta miga de pan se irá alargando conforme el usuario acceda a elementos anidados.

Vista de Pedido

A esta vista (Figura 7.90) se accederá desde la vista *"Mis pedidos"* de la sección *"Mi Cuenta"* cuando el usuario seleccione uno de los pedidos para ver sus detalles.

En la parte superior conservaremos la miga de pan, que ahora tendrá el numero de pedido. Se decide no mostrar la columna lateral en esta vista, dado que es una vista que va a contener mucha información, y, potencialmente muchos elementos.

En la parte izquierda tendremos una zona que nos mostrará, la fecha en que se realizó el pedido, el estado del pedido (solo aparecerá uno de los 4 expuestos) y los productos que se incluyeron en el pedido por parte del cliente junto con el precio por unidad y las unidades que compró el cliente. Si esta zona de productos se alargase, en la parte izquierda de esta zona aparecerá una barra para poder desplazar la lista y ver el resto de productos.

En la parte inferior, habrá un botón para poder contactar con los administradores de la tienda, el cuál redirigirá a la sección de *"Contacto"* para que el cliente pueda resolver sus dudas.

En la parte derecha habrá una zona que hará las veces de resumen del pedido, solo mostrará un resumen de los costes y totales de este.

Debajo, estará la dirección de envío, y un botón para cancelar el pedido que al pulsarlo pedirá confirmación y un motivo para cancelarlo. Solo se podrá cancelar el pedido si este está en estado *"Aceptado"* o en estado *"En Preparación"*.

Vista de Contacto

A esta vista (Figura 7.91) se llegará desde cualquier botón *"Contacto"*, los cuales estarán situados por diversas partes del aplicativo. El formato de esta vista consta de diversos campos a rellenar por el usuario en el que proporcionará sus datos personales, un motivo de consulta y el problema al que se está enfrentando para que los administradores puedan actuar en su ayuda con la mayor brevedad posible.

Vista Sobre Nosotros

Esta vista (Figura 7.92) contendrá una breve presentación de la empresa junto con una serie de imágenes a modo de página para la promoción y explicación de los servicios y productos que ofrece la empresa.

Vista Condiciones del servicio

En esta vista (Figura 7.93) estarán expuestas para el cliente los términos y condiciones de el servicio de compra en la tienda on-line. Contará con una columna con las distintas secciones, cada una de las cuales será un desplegable que abrirá un párrafo explicando esa sección.

Vista de Cesta de la Compra

Esta vista (Figura 7.94) será una de las más importantes del sitio web. A medida que el usuario vaya añadiendo productos a la cesta de la compra, serán reflejados en esta vista junto con la cantidad de producto añadido al pedido..

A esta vista se accederá pulsando el icono de la cesta situado en la parte derecha de la barra de navegación. Este icono cuenta con un circulo en fucsia que aparecerá si se añade algún producto a la cesta y contará con un numero dentro que indicará el numero de productos distintos que ha añadido el usuario a la cesta de la compra.

Se podrán eliminar productos de la cesta, así como modificar la cantidad a comprar de un producto, con un enlace y un *"Spinner"* respectivamente. Habrá un botón en la parte inferior para regresar a la tienda y otro para continuar con el pedido de los productos de la cesta. En la parte superior habrá una barra para indicar los pasos para realizar el pedido, y a medida que se vaya moviendo el usuario por los pasos se irá marcando el paso en el que se encuentra. En la parte derecha habrá una tabla con los costes totales de los productos, el envío y el precio final de todo. El botón *"Continuar"* llevará al usuario a la vista de completar pedido.

Vista completar el pedido

Cuando el usuario ya ha seleccionado los productos a adquirir, ha revisado la cesta de la compra y ha hecho click en *"Continuar"* se llega a esta vista de completar pedido (Figura 7.95), en la cual se terminan de introducir los datos para el pedido.

En la parte superior aparecerá marcado *"Envío"* para saber en que parte del proceso de realizar un pedido se encuentra el usuario.

En esta vista aparecerán los datos personales del usuario que introdujo cuando se registró, pero serán modificables, ya que estos datos estarán asociados únicamente al pedido, y no al usuario. Esto se hace, por si un usuario quiere hacer un pedido a nombre de otra persona y que le llegue a su casa, para lo cuál está la dirección de entrega. Los datos de esta zona también serán precargados con la dirección de envío habitual introducida por el usuario en el registro pero, una vez más, serán modificables. Debajo de esa zona, aunque no se vea, habrá una opción para hacer esa dirección la dirección de facturación o introducir una nueva. Tras esto el usuario pulsará en el botón *"Continuar"* para avanzar a la siguiente vista.

Vista de pago

Cuando el usuario haya introducido sus datos de envío llegará a esta vista de pago (Figura 7.96). La vista contará con un formulario para que el usuario introduzca su dirección de facturación, aunque el formulario estará auto-completado con la dirección de facturación introducida cuando el usuario se registró. En la parte inferior habrá diversos botones con las diversas opciones de pago disponibles. Es-

tos botones aumentarán o disminuirán en función de los métodos de pago que se implementen en el servicio.

Al hacer click en uno de estos botones, se comprobarán los datos del formulario y el usuario será redirigido al proveedor de pagos elegido.

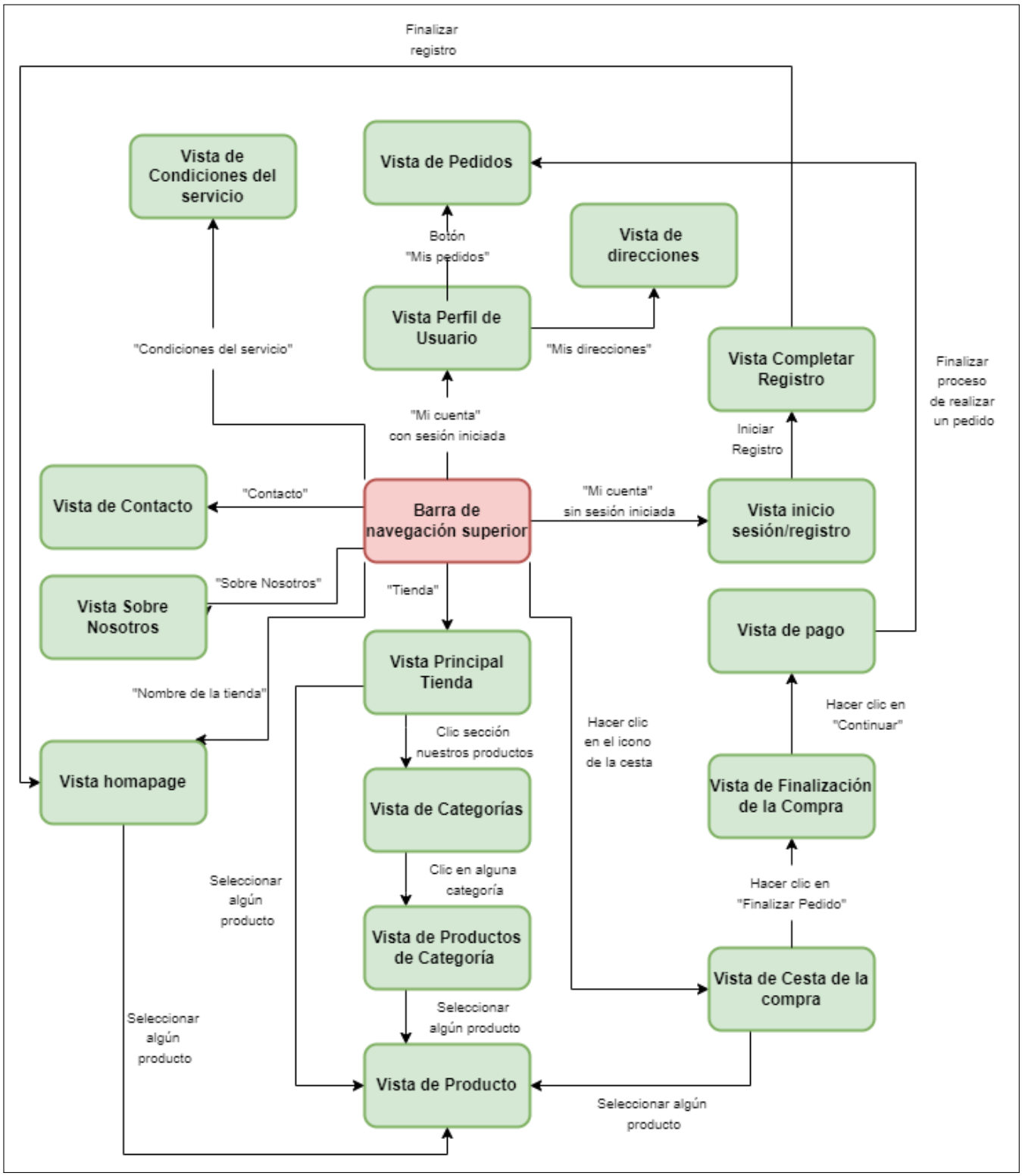


Figura 7.81: diagrama de navegación del sitio web

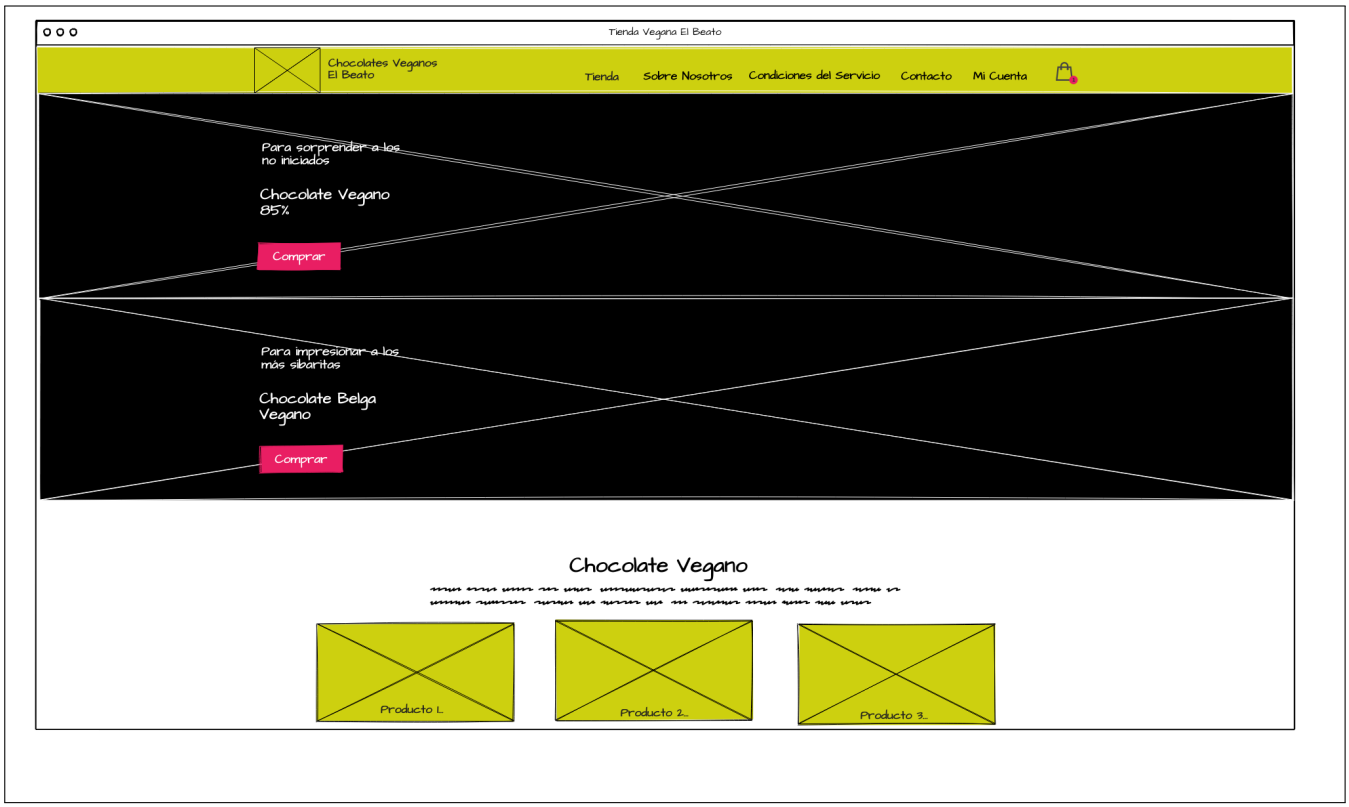


Figura 7.82: Página principal

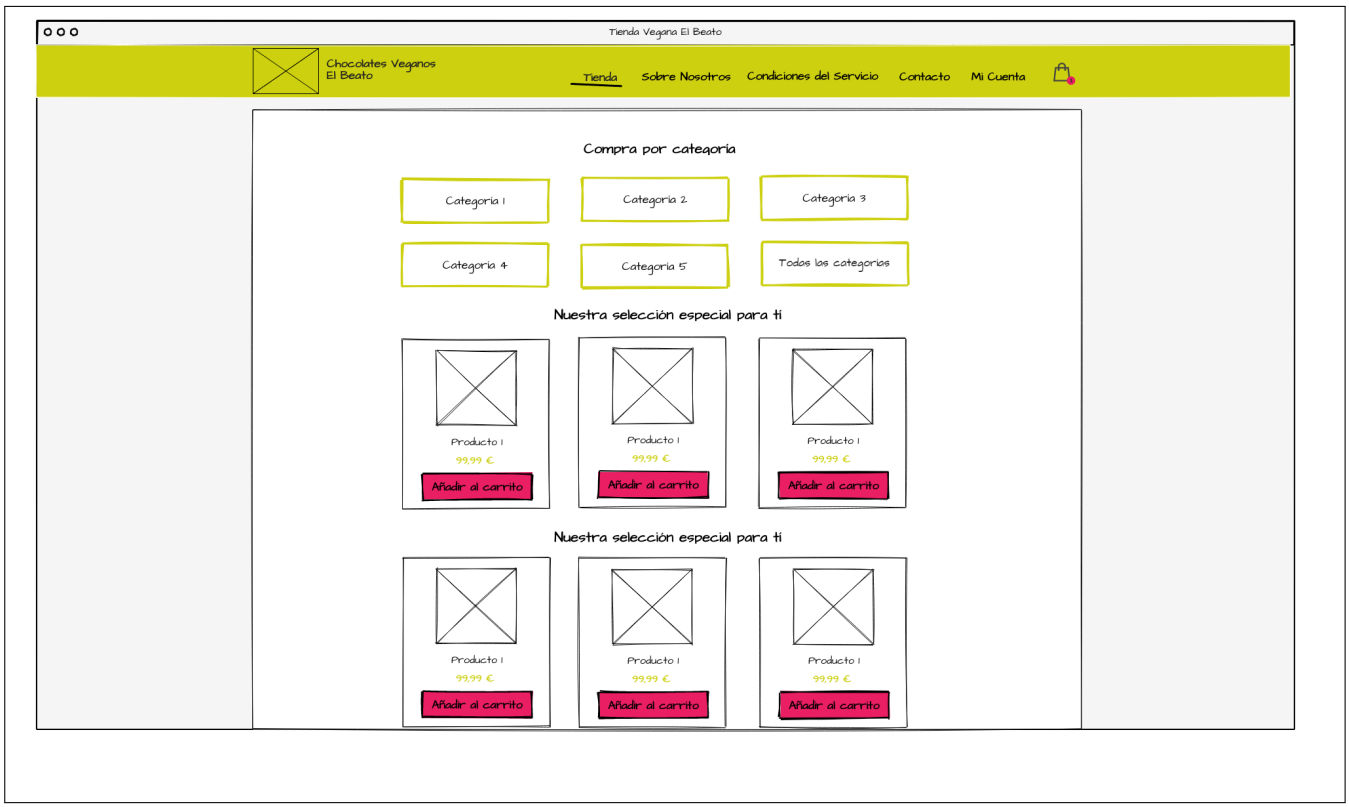


Figura 7.83: Vista principal de la tienda

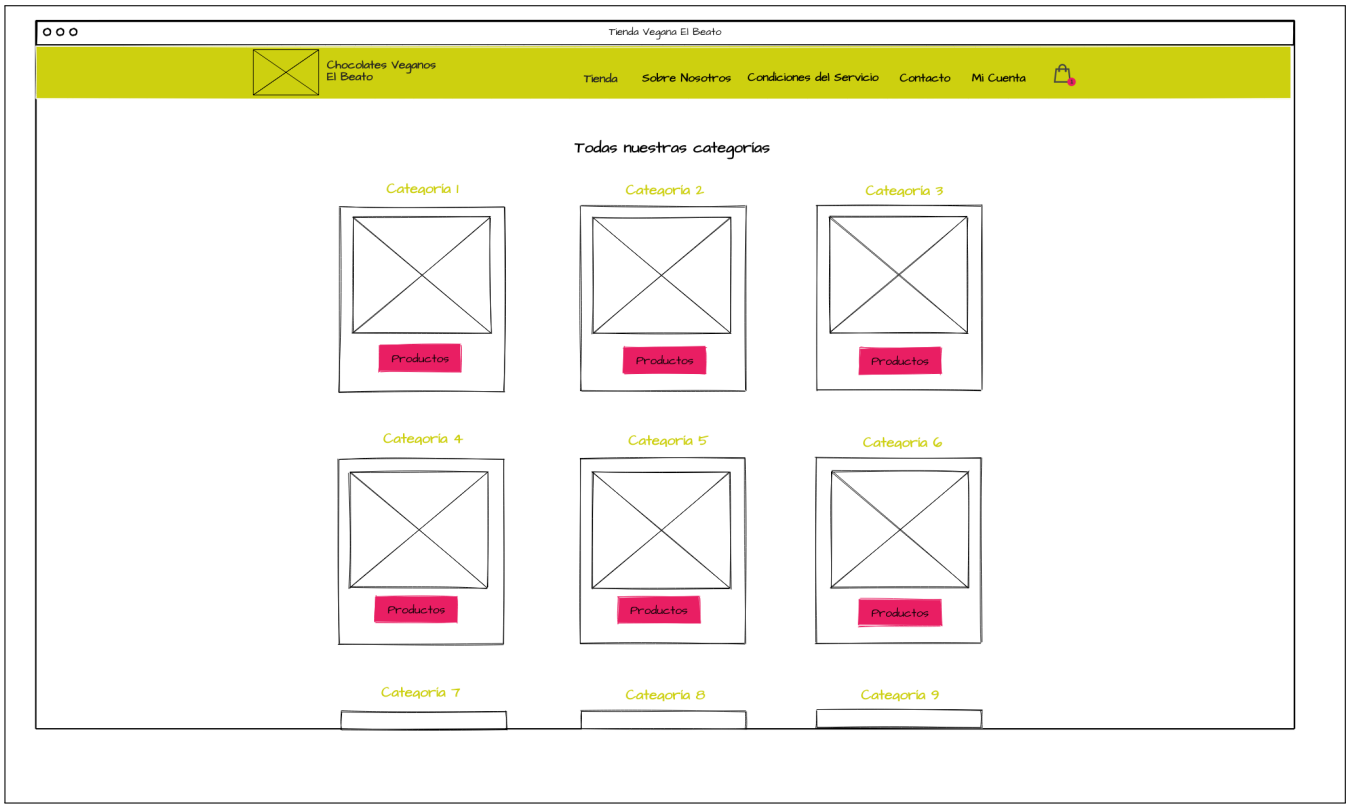


Figura 7.84: Vista de categorías de productos

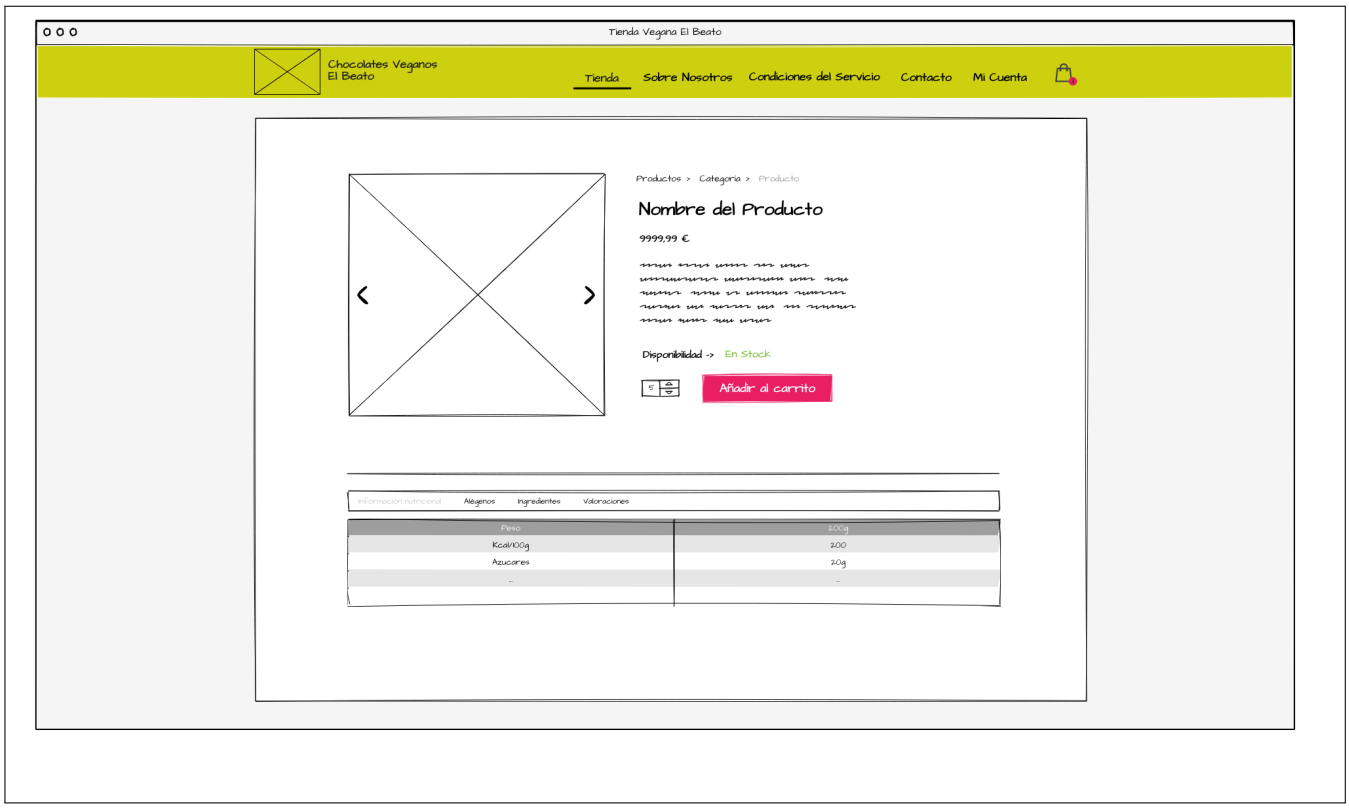


Figura 7.85: Vista de producto

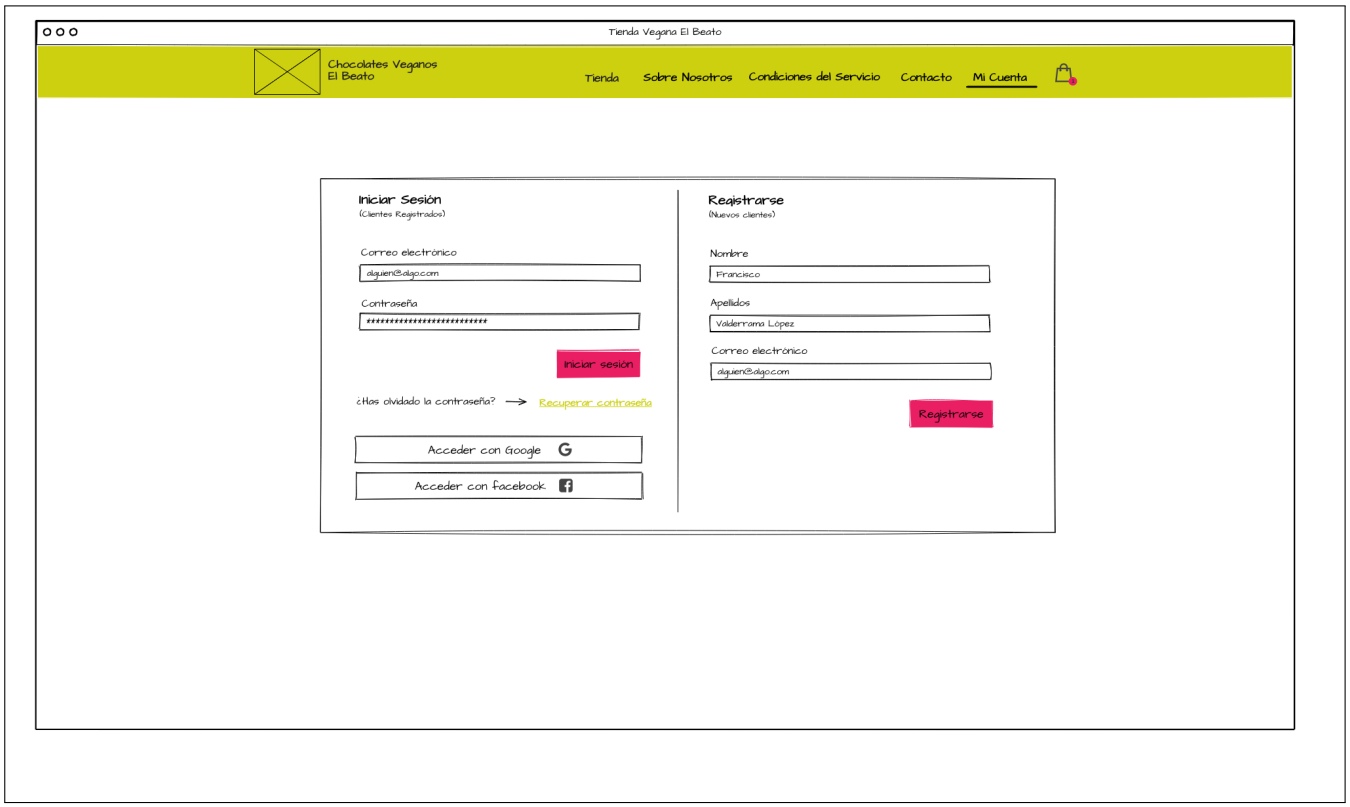


Figura 7.86: Vista de inicio de sesión y registro

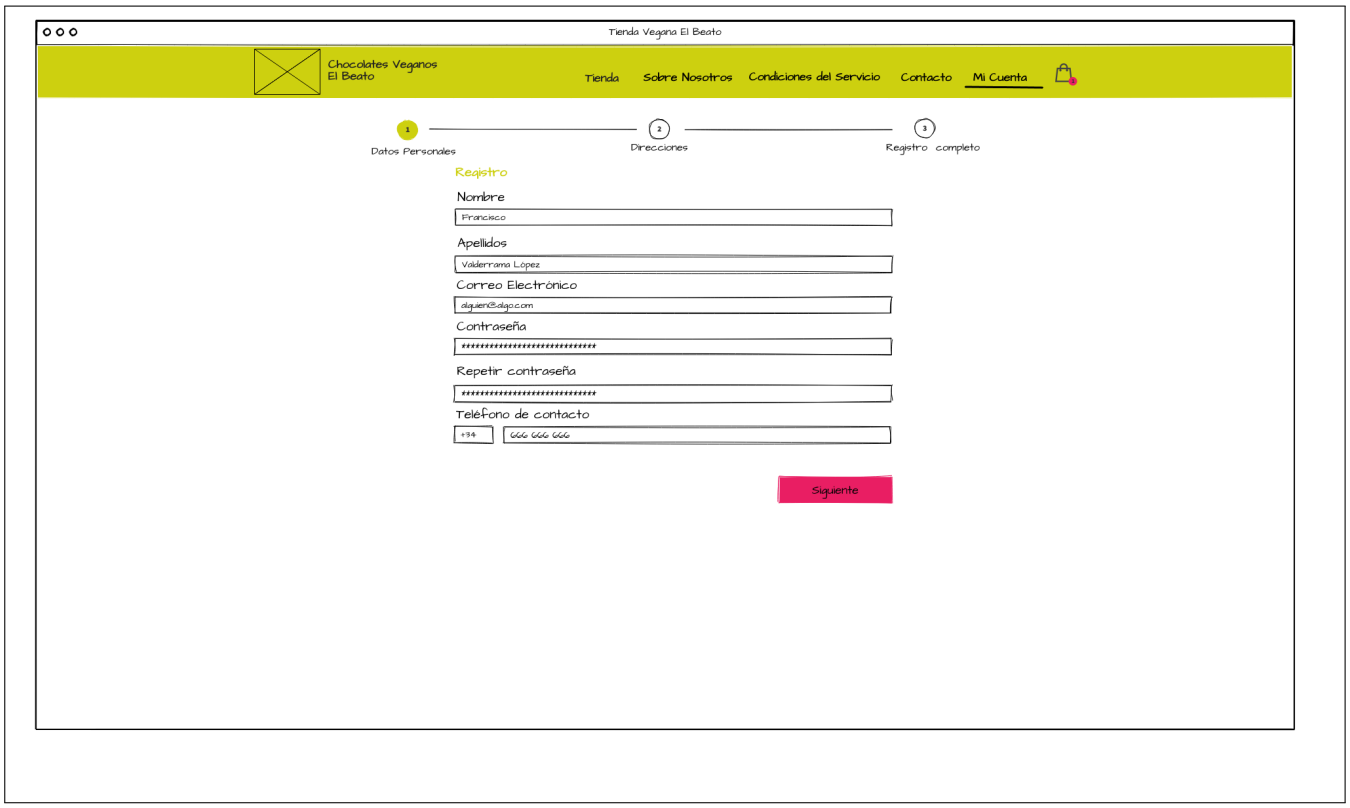


Figura 7.87: Vista de completar registro 1

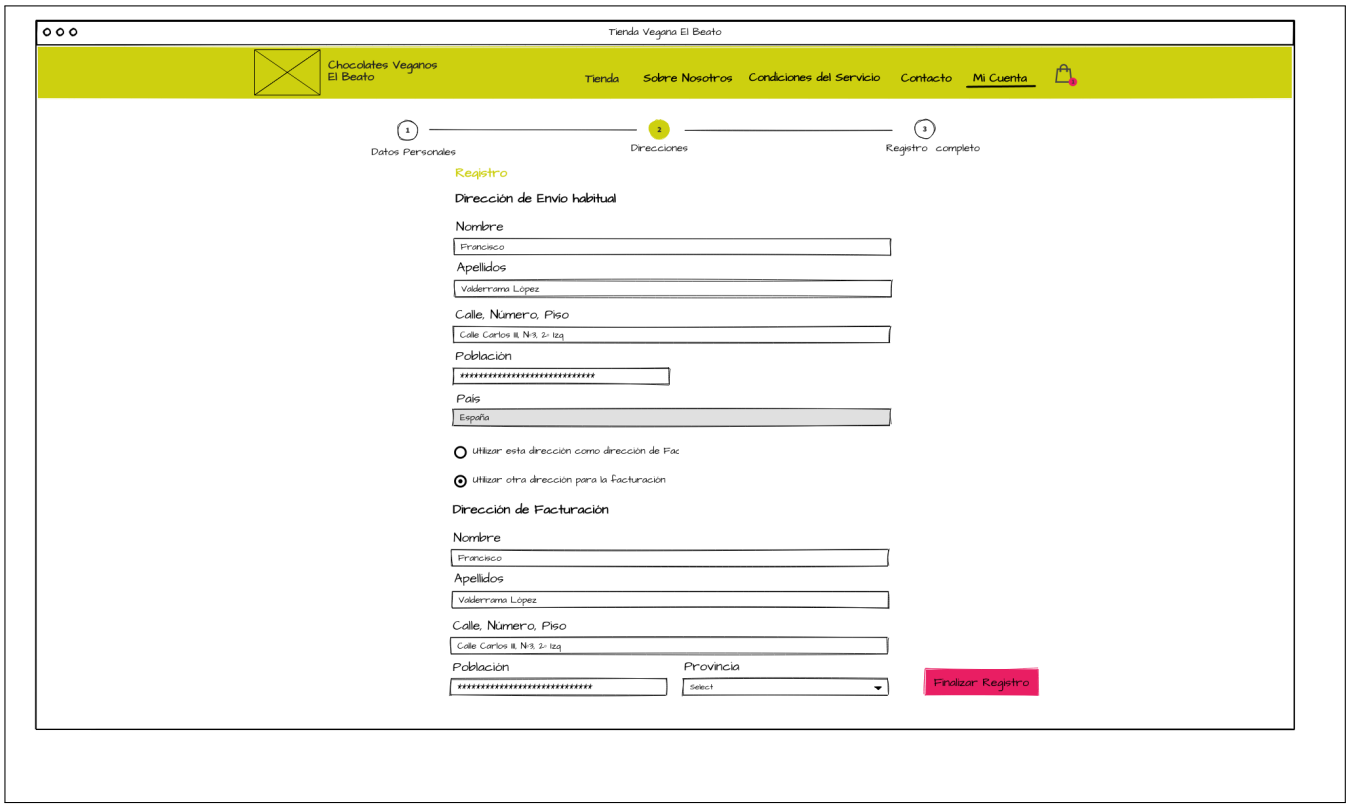


Figura 7.88: Vista de completar registro 2

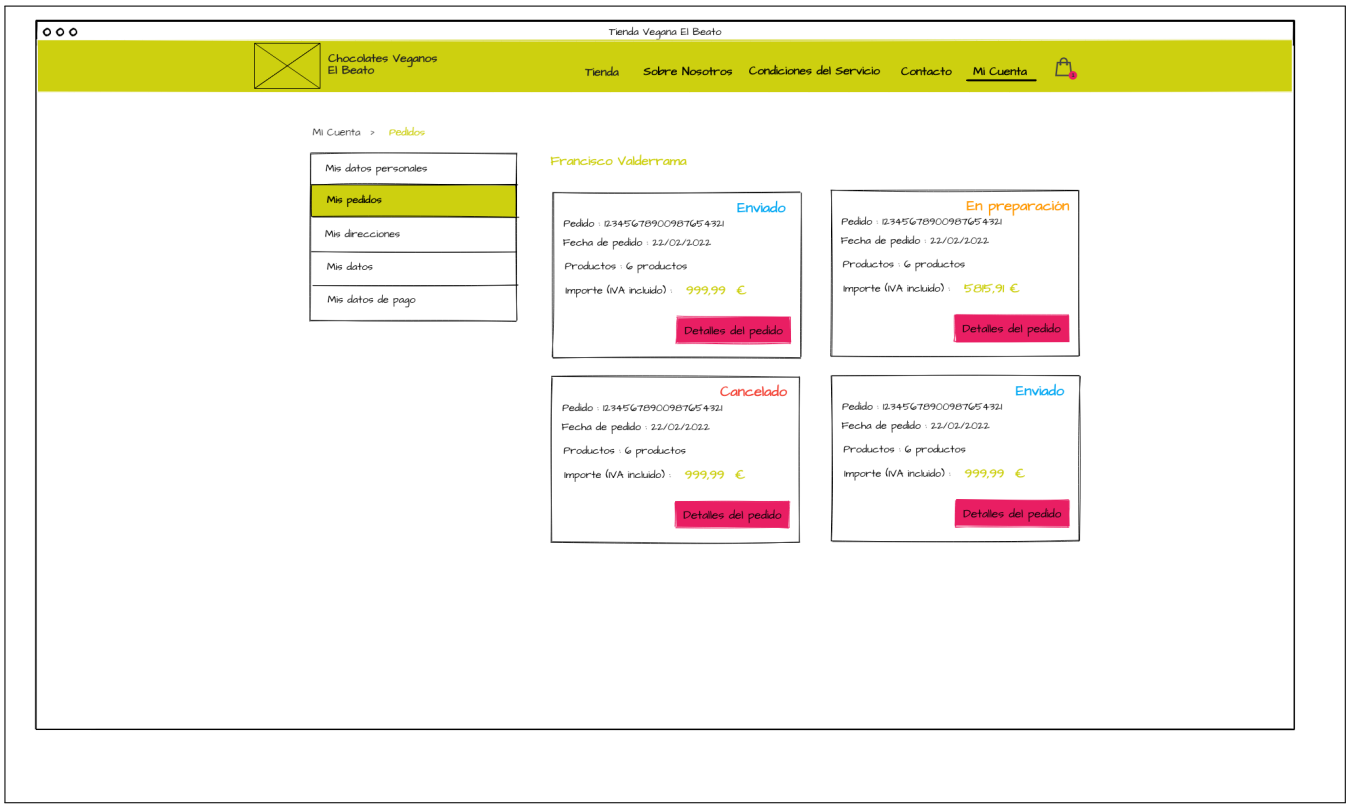


Figura 7.89: Vista de Mi cuenta

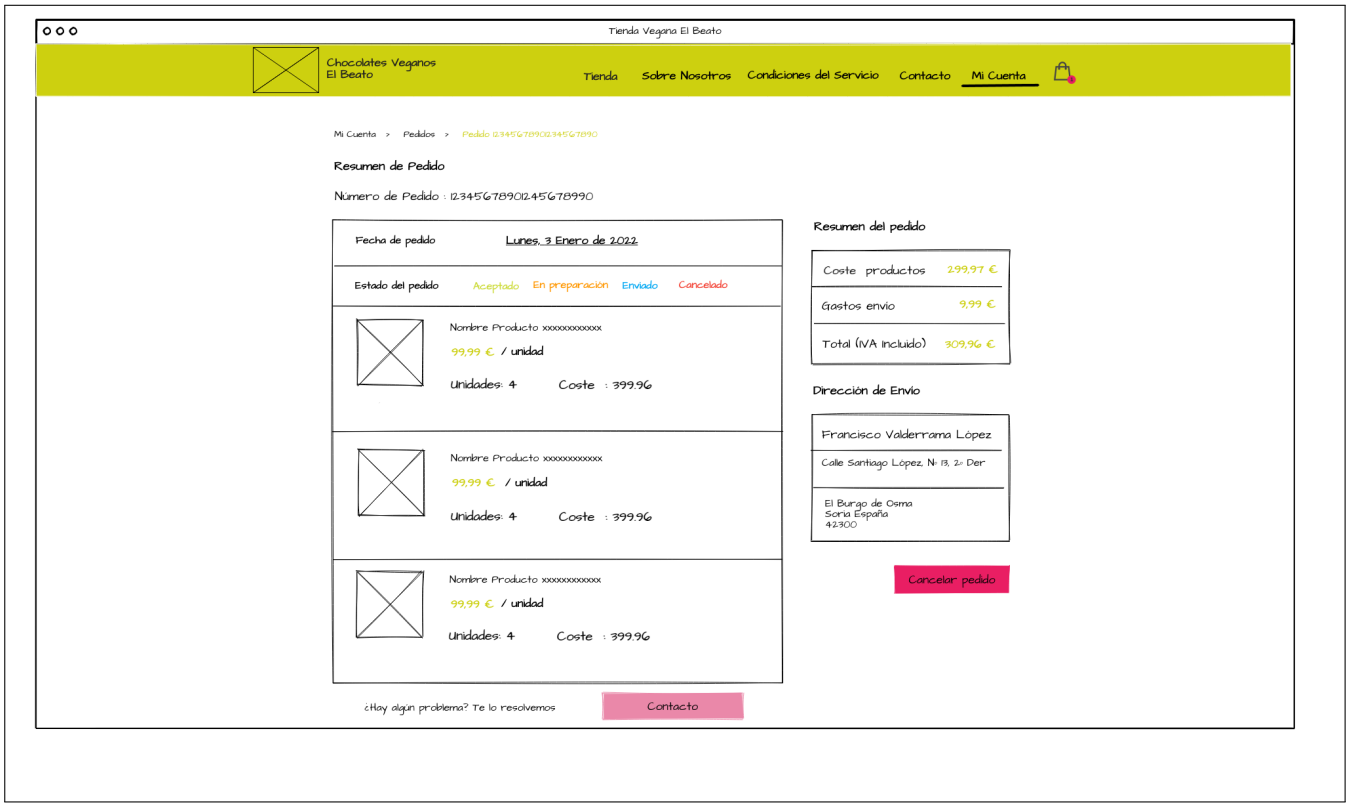


Figura 7.90: Vista de Pedido

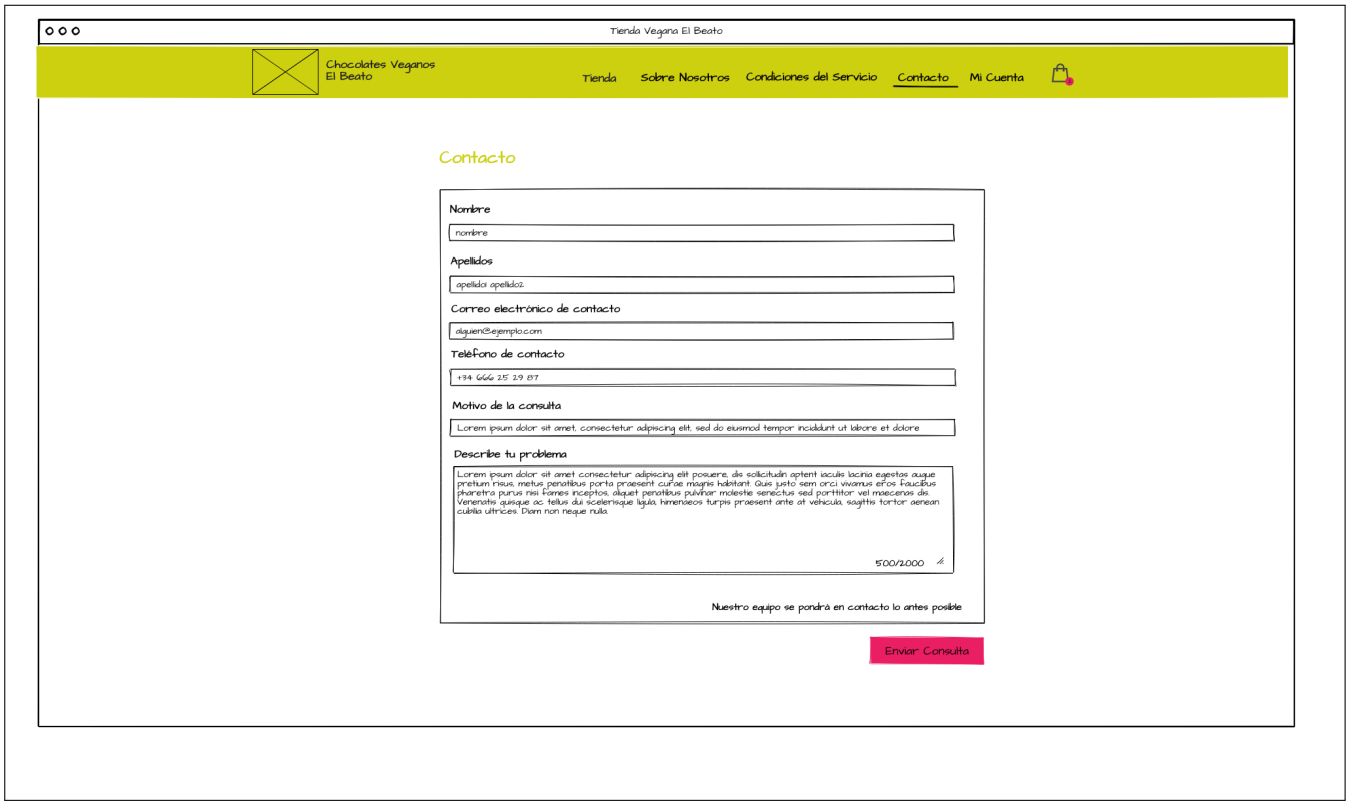


Figura 7.91: Vista de Formulario de Contacto

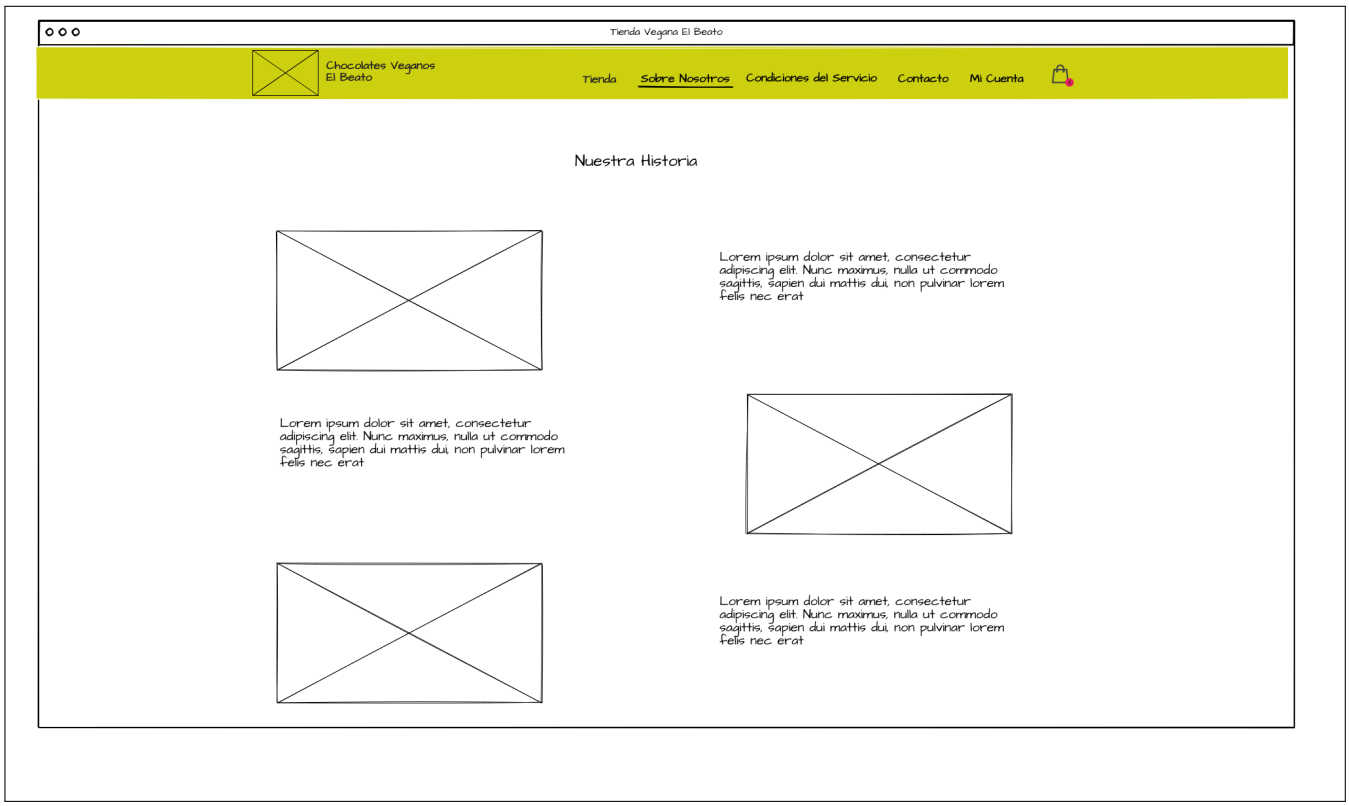


Figura 7.92: Vista Sobre Nosotros

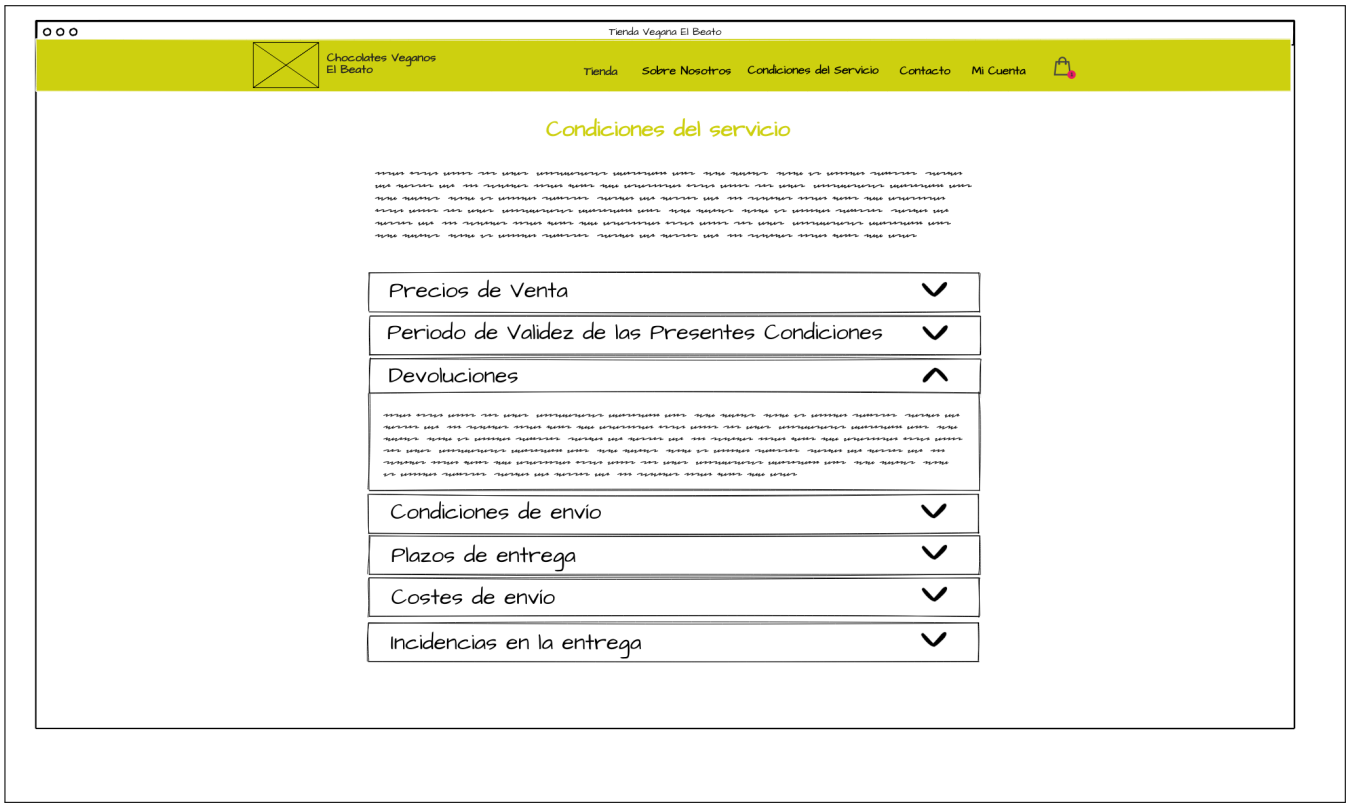


Figura 7.93: Vista Condiciones del servicio

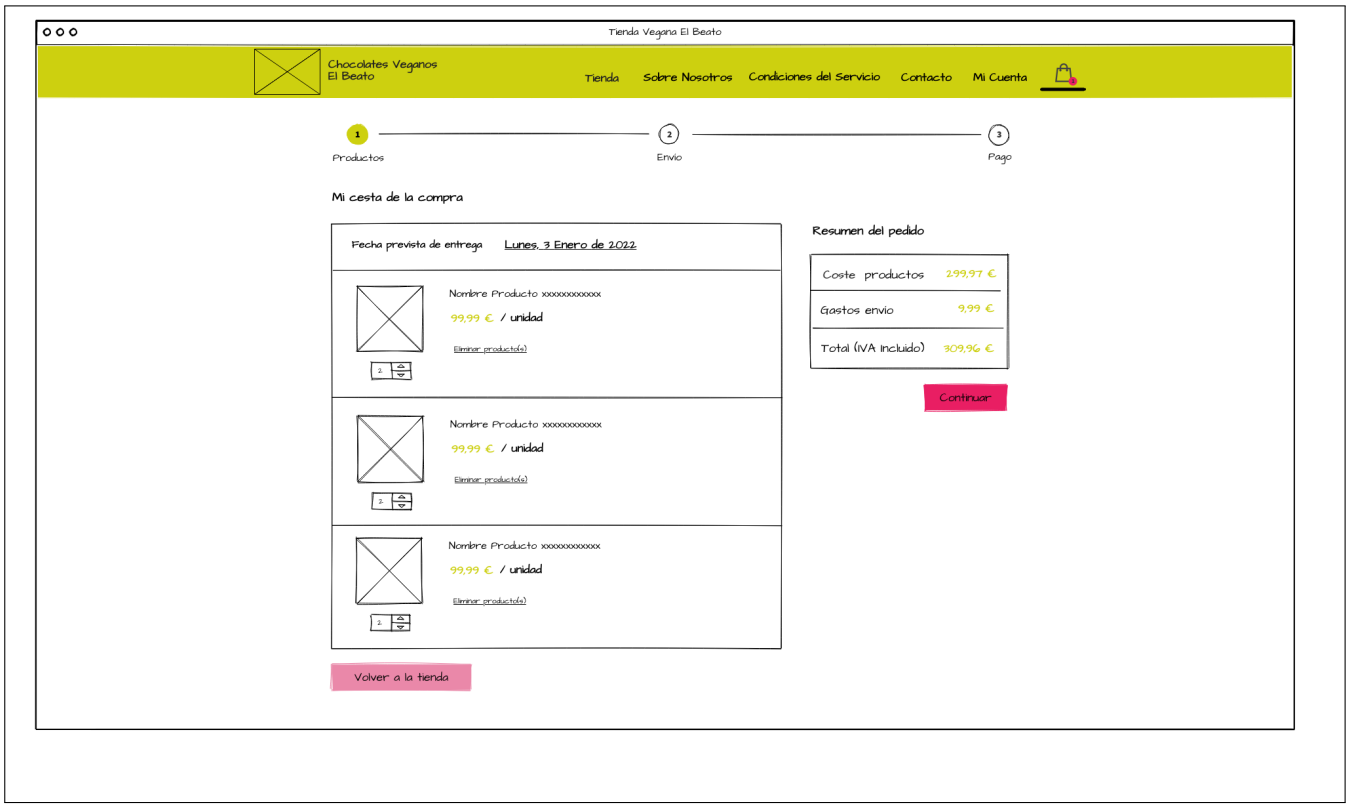


Figura 7.94: Vista Cesta de la Compra

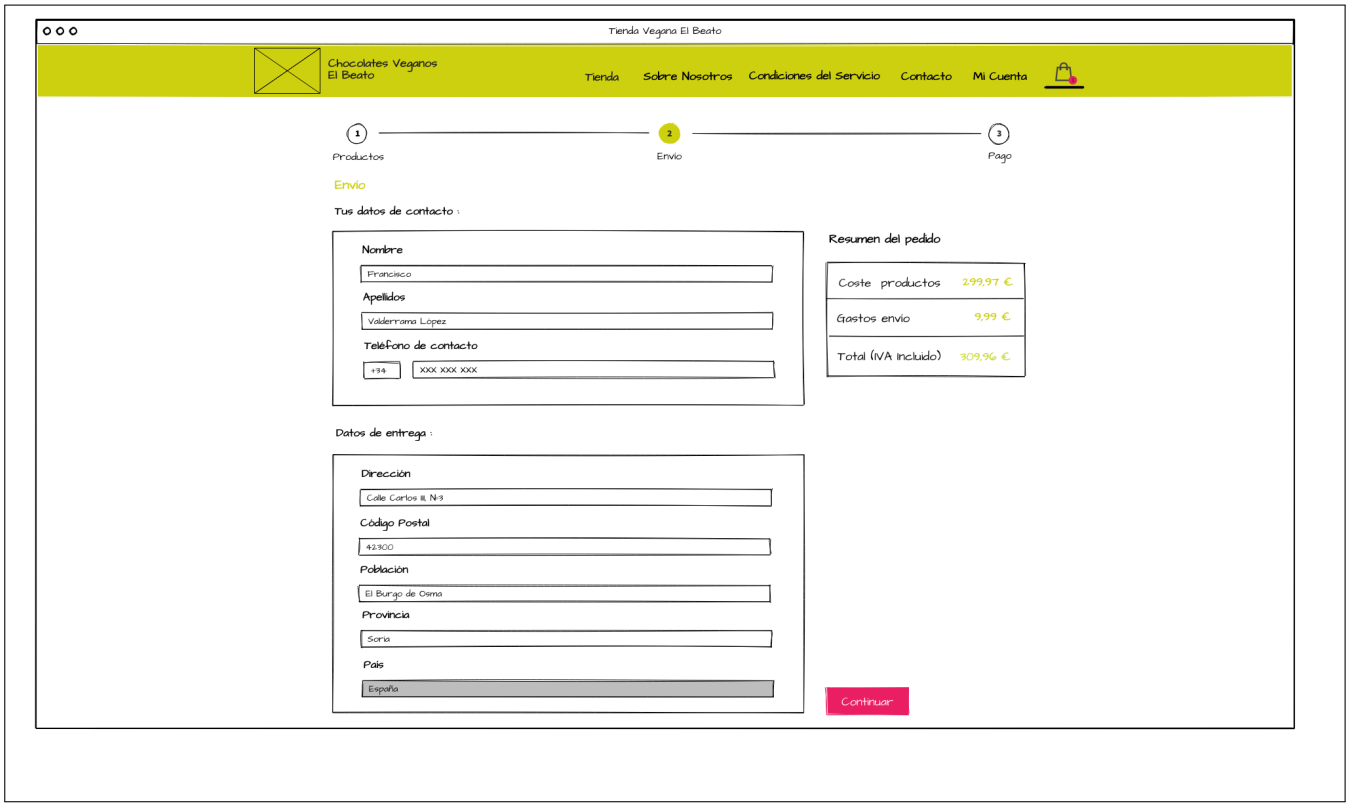


Figura 7.95: Vista completar pedido

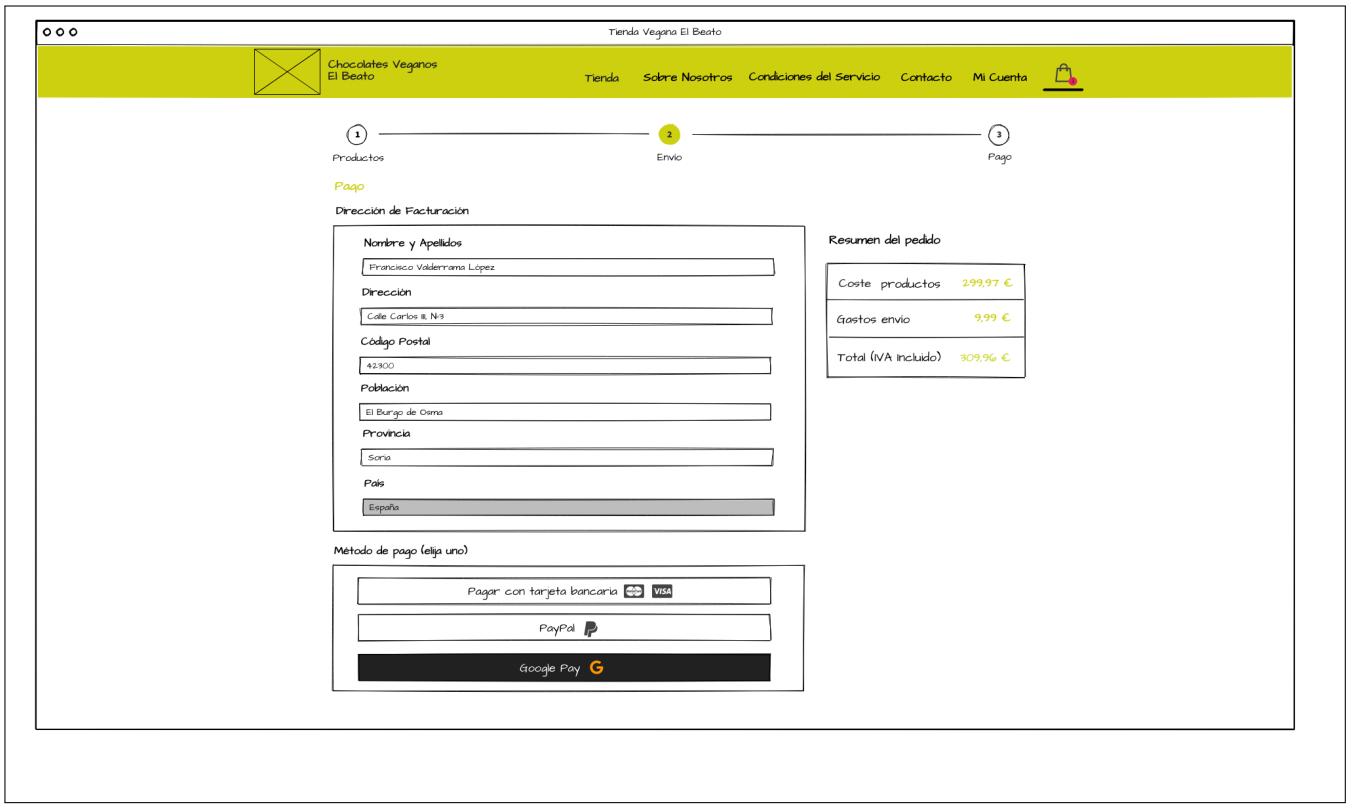


Figura 7.96: Vista de pago

Capítulo 8

Implementación

8.1. Estructura del directorio de código

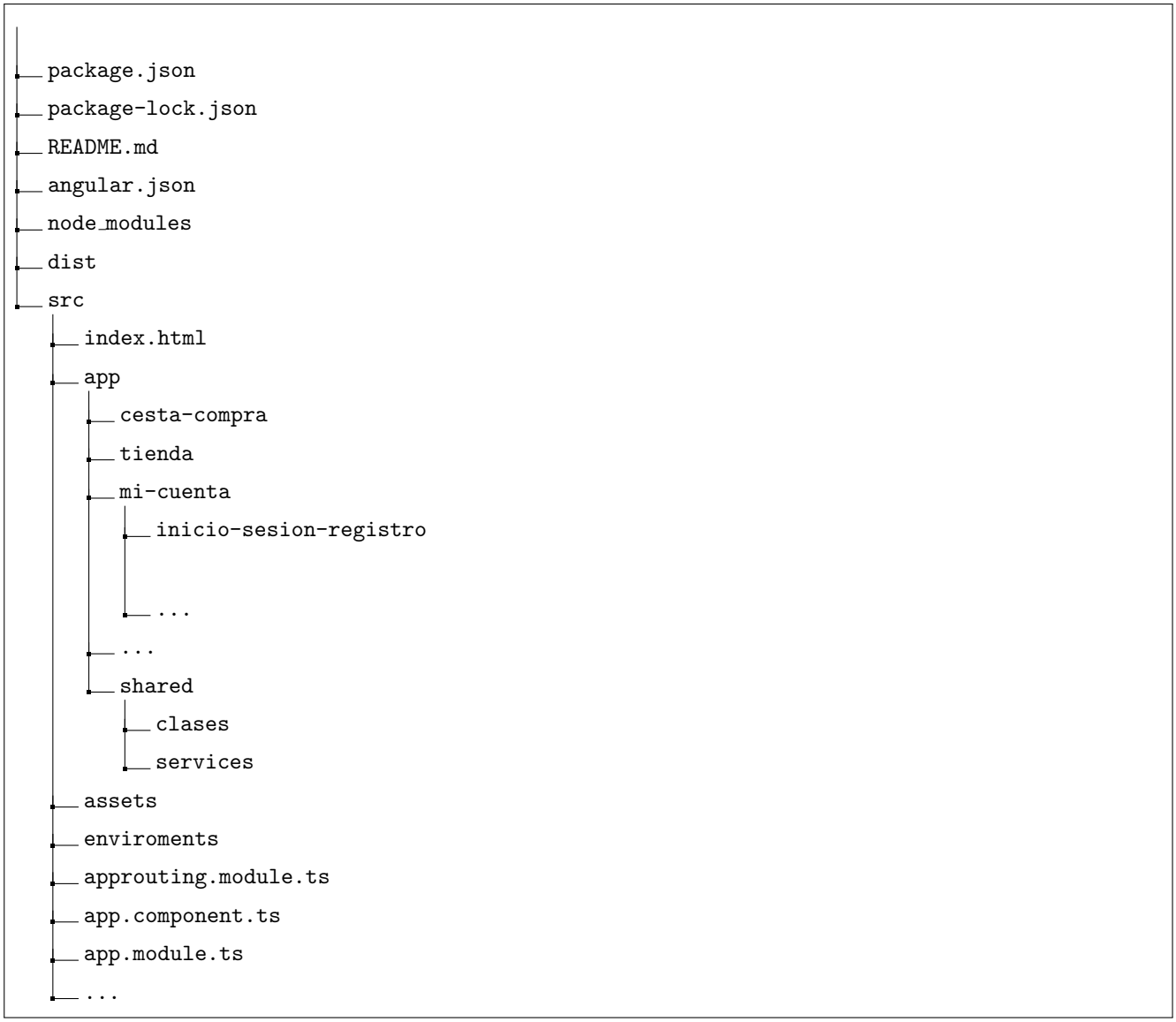


Figura 8.1: Estructura del código de la aplicación

En la raíz de el proyecto se encuentran los ficheros de configuración de `npm` y Angular, junto con un README y diversas carpetas que contienen código. A continuación se detallan en profundidad la carpetas continentes del código dentro de el directorio raíz:

- `node_modules`: contiene todos las dependencias del proyecto en forma de paquetes `npm` instalados para el correcto funcionamiento de la aplicación Angular.
- `dist`: posee el código compilado de la aplicación, listo para su despliegue.
- `src`: contiene los módulos, componentes, servicios y clases creadas en el proyecto.
  - `index.html`: es la página HTML donde se introduce el componente principal `app`.
  - `app`: contiene las carpetas con componentes, variables de entorno, servicios, el NgModule principal, el módulo de routing de la aplicación y los propios ficheros descriptivos de el componente `App`.
    - `shared`: contiene dos carpetas, una para las clases representativas de los tipos y otra para los servicios de la aplicación.
- **assets**: posee imágenes que se emplean en la aplicación.
- `environments`: posee los archivos de configuración de variables de entorno, los cuales poseen campos necesarios para inicializar el módulo Firebase en nuestra aplicación.

8.2. Cambios realizados en la aplicación final

8.2.1. Cambios realizados en la interfaz de usuario de la aplicación

Durante la fase de implementación se realizaron cambios en varios elementos de la interfaz de navegación, los cuales incluyen la eliminación de algunas vistas que no se consideraron necesarias, ya que solo empeoraban la experiencia de usuario. Por ejemplo las vistas de registro (Figura 7.87 y Figura 7.88) se fusionaron en una sola, lo cuál hacía más fácil la validación de datos introducidos y simplificaba el proceso de registro. La vista de pedido se modificó para poder incluirla dentro de la vista de perfil-cuenta (Figura 7.89) porque se entendió que era más fácil la navegación para el usuario de esta forma, al no perder nunca de vista la columna de navegación.

También se realizaron cambios en la vista de la página principal de la tienda, la cuál paso de tener los productos destacados expuestos en recuadros a exponerlos dentro de un *carousel* de productos.

Se realizaron cambios en la fuente de letra, la cuál se modificó por una más legible en toda la aplicación, menos en la barra de navegación por petición del cliente.

En la vista de categorías se eliminaron los botones de dentro de las categorías al no mostrarse finalmente una imagen para la categoría, por lo que se decidió dejar únicamente un texto que contuviese el nombre de la categoría y que fuese un enlace para acceder a los productos de la categoría.

8.2.2. Dirección de despliegue de la aplicación

La aplicación ha sido desplegada para su prueba y uso en la siguiente dirección:
`https://gleaming-terra-226017.web.app/`

Capítulo 9

Pruebas

Para poder comprobar el buen funcionamiento de la aplicación desarrollada se han llevado a acabo una serie de pruebas con usuarios.

9.1. Pruebas con usuarios

Son las pruebas más importantes dado que el principal objetivo de nuestra aplicación es la venta de productos es especialmente importante que los usuarios finales puedan hacer uso de la aplicación de una forma sencilla, rápida y que no sea frustrante para que no abandonen la compra.

La idea de estas pruebas es también encontrar posibles errores en la aplicación que no se hayan tenido en cuenta en la fase de desarrollo. Esto suele ocurrir porque al desarrollar no podemos tener presentes todas las casuísticas que el usuario final puede enfrentar, y normalmente los desarrolladores suelen realizar siempre las acciones de la formar en la que las han diseñado, cosa que los usuarios desconocen y, por ende, realizan secuencias de acciones que pueden ser alternativas e imprevistas.

Para realizar estas pruebas se ha tomado a diversos usuarios potenciales finales de la tienda y se le ha pedido que realicen una secuencia de acciones en la aplicación:

0 - Acceder al sitio web desde su dispositivo común para esta acción.

1 - Registrarse en la aplicación.

2 - Navegar hasta la tienda principal.

3 - Añadir un producto de la tienda a la cesta.

4 - Acceder a los detalles de un producto.

5 - Añadir 3 unidades de ese producto a la cesta.

6 - Ir a la cesta para ver los productos.

7 - Eliminar un producto de la cesta.

8 - Modificar la cantidad de un producto en la cesta.

9 - Iniciar sesión.

9 - Ir a la tienda otra vez.

10 - Ir a una categoría de productos.

11 - Seleccionar un producto y añadirlo a la cesta.

12 - Ir a la cesta y realizar un pedido.

13 - Cancelar el pedido que se acaba de realizar.

14 - Cambiar el teléfono.

15 - Cerrar sesión.

16 - Recuperar la contraseña.

En total se realizaron pruebas con 22 usuarios, de los cuales 18 eran usuarios habituales de tiendas on-line, 1 era completamente nuevo en este tipo de servicio on-line , y 3 eran compradores no habituales. En el transcurso de estas pruebas:

- Se descubrió un error al recuperar la contraseña, ya que la pantalla se quedaba en blanco al hacer click en el link enviado al correo para recuperarla. Se detectó que el error era debido a que la url que estaba configurada en Firebase para el la recuperación de la contraseña era errónea, y la aplicación mostraba una página en blanco, dado que a la url que se intentaba acceder no existía en la aplicación.
- Algunos usuarios manifestaron que el comportamiento de la barra de tareas les resultaba desagradable cuando accedían al sitio web desde dispositivos móviles, dado que esta se quedaba abierta de forma continua y no permitía ver la página a la que se había accedido. Se solucionó añadiendo una serie de atributos y valores a la *headbar* de la aplicación, concretamente los atributos "`aria-bs-toggle`" "`aria-bs-target`"los cuales son atributos provistos por bootstrap.
- Uno de los usuarios manifestó que el texto de la vista restablecer contraseña era confuso y era mejor que el texto fuera "Cambiar contraseña" en vez de el texto actual, el cuál era "Restablecer contraseña". Sin embargo, otros tres usuarios manifestaron que les sería menos intuitivo, dado que ellos no querían cambiar su contraseña al iniciar ese proceso, querían recuperar la anterior. Por ello se decidió dejar el texto en "'Restablecer contrasña".
- Algunos usuarios manifestaron que el comportamiento del carro de la compra al iniciar sesión era poco intuitivo, ya que, si añadían productos a la cesta y luego iniciaban sesión, los productos que habían añadido simplemente desaparecían. Los usuarios de prueba manifestaron que les sería más intuitivo que los productos que habían añadido sin iniciar sesión fueran añadidos a los de el carro anterior. Se deja como propuesta de mejora ya que se estima que puede ser una tarea de una duración elevada para su implementación.
- Dos usuarios manifestaron que los textos de las categorías les resultaban confusos, dado que el texto "Todas las categorías"les daba a entender que iban a ver todos los productos de todas las categorías, y no todas las categorías. También manifestaron que les resultaba extraño que se mostrasen 5 categorías en la vista de la tienda y 5 en la vista de categorías. Se decidió mostrar 2 categorías en la tienda y todas en la vista de categorías para hacerlo más intuitivo, y se implementó el cambio de forma inmediata.
- Un usuario quiso leer las condiciones del servicio durante el proceso de registro, y no pudo debido a que el enlace no tenía funcionalidad. Se solucionó de manera rápida.

- Durante el proceso de registro de uno de los usuarios realizó una secuencia de acciones que hizo aparecer un error en el registro del usuario. Al seleccionar y de-seleccionar la opción de usar la misma dirección para envío y facturación, el formulario no introducía bien los datos. Se procedió a su solución.
- Un usuario manifestó que le resultaba complicado detectar que un producto estaba agotado cuando lo veía en la página principal de la tienda ya que el color era muy parecido al del botón de añadir al cesta de la compra. Esto no fue considerado error porque el comportamiento solo fue insatisfactorio para este usuario.
- Un usuario manifestó que le resultaba confuso que se mostrasen las flechas de cambiar de imagen en el carousel de las imágenes de los detalles de un producto. Se procedió a modificarlo.
- Un usuario decidió enviar una solicitud de contacto y se descubrió que los datos no se enviaban de forma correcta. Se detectó que era porque se borraban los datos del formulario antes de enviarlos a la base de datos. Se solucionó de manera rápida.
- Un usuario manifestó que le resultaba poco intuitivo que al entrar a la vista de "Mi cuenta" a no hubiera ningún apartado ya seleccionado. Se decide hacer que la vista que se muestre al acceder sea la de "Mis datos personales". Se solucionó de manera satisfactoria.
- Un usuario detectó un problema al añadir a la cesta de la compra 3 productos de coste 3.3€que la suma de los precios de los productos se mostraba como 9.89999999999€, lo que hacía que se descuadrase el texto y que se saliese de los márgenes. Se procedió a solucionarlo.
- Dos de los usuarios mostraron su frustración al intentar iniciar sesión con Google o Facebook y ver que esos botones no hacían nada. Se decide ocultarlos en la vista de inicio de sesión para no dar lugar a otros mal entendidos por los usuarios.

Capítulo 10

Conclusiones y líneas de trabajo futuras

10.1. Conclusiones

Tras finalizar el proyecto se puede considerar que el desarrollo del mismo ha sido satisfactorio.

Con respecto a los objetivos inicialmente planteados, la herramienta desarrollada:

- Permite a los usuarios ver los productos que el administrador ha añadido al catálogo.
- Se ha implementado una cesta de la compra que permite llevar las acciones comúnmente asociadas a este elemento en un comercio electrónico.
- Permite que ños productos de la tienda se agrupen en categorías de forma correcta y se muestran donde deben.
- Es accesible desde cualquier territorio del mundo al que lleguen los servicios de Google.
- Permite puede navegar de un producto a otro y a categorías sin que suponga errores en la aplicación.
- Lleva un conteo de los productos que puede ser modificado por el administrador en la base de datos y modifica de forma automática las existencias de un producto al hacer pedidos que contengan ese producto.

De lo inicialmente planteado no se ha logrado que el sistema permitiera pagos por falta de tiempo. Eramos conscientes al plantearlo de su dificultad, por la limitación temporal con que se cuenta para la realización del TFG. Sin embargo, se decidió mantenerlo para así tenerlo presente durante todo el desarrollo y, de esta manera, dejar la aplicación preparada para que pueda ser añadido de manera sencilla en el futuro.

En lo que respecta a los objetivos personales se considera que se han cumplido todos:

- La licitación de requisitos y el trato con el cliente ha sido más complicada de llevar a cabo de lo esperado por la poca disponibilidad que tenía debido a sus ocupaciones.
- Se ha aprendido en profundidad el funcionamiento de uno de los frameworks de desarrollo web de tipo SPA más asentados en el mercado actual que es Angular y se han conocido otras muchas tecnologías que se emplean a día de hoy.
- Se ha aprendido a trabajar con servicios de Back-end as a Service y a trabajar con servicios de Hosting.
- Se ha mejorado en gran medida la habilidad para el desarrollo de elementos gráficos e interfaces web.

10.2. Lineas de trabajo futuras

Las líneas de trabajo futuras pasan por mejoras en la experiencia de usuario(muchas detectadas gracias a las pruebas realizadas) y la compleción del sistema para su puesta en marcha real:

- Pasarela de pago.
- Mejoras en carga de datos: sería bueno implementar que la carga de productos sea progresiva a medida que se hace scroll por las distintas páginas.
- Buscador: podría ser implementado para facilitar la búsqueda de productos.
- Conexión con la base de datos actual: convendría estudiar en un futuro la conexión con la base de datos del cliente para tenerlo todo centralizado y tener nuestro back-end actual como respaldo.
- Distintos idiomas: podría incluirse ya que en el territorio español se hablan distintas lenguas, o por si en un futuro se desea vender en el extranjero.
- Aumento del rendimiento: se podría estudiar dado que de esta forma mejoraría el posicionamiento SEO de la web en los buscadores.

Referencias

[1] About diagrams.net. https://www.diagrams.net/about. Ultimo acceso el 14/06/2022.

[2] About npm. https://www.npmjs.com/about. Ultimo acceso el 10/06/2022.

[3] Aeat amortizaciones. https://sede.agenciatributaria.gob.es/Sede/impuesto-sobre-sociedades/que-base-imponible-se-determina-sociedades/amortizaciones.html. Ultimo acceso el 13/06/2022.

[4] Angular fire npm package. https://www.npmjs.com/package/@angular/fire. Ultimo acceso el 10/06/2022.

[5] Angularfire. https://github.com/angular/angularfire/blob/master/README.md. Ultimo acceso el 08/06/2022.

[6] Astah profesional product page. https://astah.net/products/astah-professional/. Ultimo acceso el 14/06/2022.

[7] Calculadora de jornada de trabajo. https://www.sesametime.com/assets/calculadora-de-jornada-de-trabajo/. Ultimo acceso el 13/06/2022.

[8] Cli overview and command reference. https://angular.io/cli. Ultimo acceso el 10/06/2022.

[9] Desarrollo de software basado en componentes. https://www.ecured.cu/Desarrollo_de_software_basado_en_componentes. Ultimo acceso el 14/06/2022.

[10] Firebase authentication documentation. https://firebase.google.com/docs/auth. Ultimo acceso el 09/06/2022.

[11] Firebase cloudstorage documentation. https://firebase.google.com/docs/storage. Ultimo acceso el 09/06/2022.

[12] Firebase firestore documentation. https://firebase.google.com/docs/firestore. Ultimo acceso el 09/06/2022.

[13] Firebase hosting documentation. https://firebase.google.com/docs/hosting. Ultimo acceso el 09/06/2022.

[14] Git. https://en.wikipedia.org/wiki/Git. Ultimo acceso el 08/06/2022.

[15] Html. https://es.wikipedia.org/wiki/HTML. Ultimo acceso el 08/06/2022.

[16] Html5. https://es.wikipedia.org/wiki/HTML5. Ultimo acceso el 08/06/2022.

[17] Introduction to angular concepts. https://angular.io/guide/architecture. Ultimo acceso el 08/06/2022.

[18] Jquery web page. https://jquery.com/. Ultimo acceso el 10/06/2022.

[19] Mockflow wireframe. https://mockflow.com/wireframing/. Ultimo acceso el 14/06/2022.

[20] Ng bootstrap. https://ng-bootstrap.github.io/#/home. Ultimo acceso el 12/06/2022.

[21] Overleaf main page. https://es.overleaf.com/.

[22] Overleaf main page. https://www.prince2.com/eur/blog/effective-risk-management. Ultimo acceso el 12/06/2022.

[23] Primeng. https://www.primefaces.org/primeng/setup.

[24] rxjs documentation. https://github.com/Reactive-Extensions/RxJS/blob/master/readme.md. Ultimo acceso el 10/06/2022.

[25] Typescript. https://es.wikipedia.org/wiki/TypeScript. Ultimo acceso el 08/06/2022.

[26] Visual studio code documentation. https://code.visualstudio.com/docs. Ultimo acceso el 14/06/2022.

[27] What is angular? — angular documentation. https://angular.io/guide/what-is-angular. Ultimo acceso el 08/06/2022.

[28] What is bootstrap. https://www.javatpoint.com/what-is-bootstrap. Ultimo acceso el 10/06/2022.

[29] ¿qué es el diseño responsive? https://www.40defiebre.com/que-es/diseno-responsive. Ultimo acceso el 10/06/2022.

[30] Aritmetics. Qué es frontend - definición, significado y ejemplos. https://www.arimetrics.com/glosario-digital/frontend.

[31] Back4App. https://www.back4app.com/. Ultimo acceso el 09/02/2022.

[32] Nazaret Barrio. ¿qué son los créditos ects? descubre el nuevo sistema de créditos europeo. https://revistadigital.inesem.es/orientacion-laboral/para-que-sirven-los-creditos-ects/. Ultimo acceso el 15/03/2022.

[33] George Batschinski. Backend como servicio: ¿qué es un baas? https://blog.back4app.com/es/que-es-un-baas-backend-como-servicio/, 2020. Ultimo acceso el 09/02/2022.

[34] CARDETAIL. Cardetail car care products. https://www.cardetail.be/. Ultimo acceso el 08/02/2022.

[35] Ryan Carniato. Frameworks para frontend más populares en 2020. https://javascript.plainenglish.io/javascript-frameworks-performance-comparison-2020-cd881ac21fce, Junio 2020. Ultimo acceso el 09/02/2022.

[36] Ben Chacon, Scott & Straub. *Pro Git, Everything you need to know abot git.* APRESS, second edition edition, 1964.

[37] NeoAttack Contenidos. ¿qué es frontend y para qué sirve? https://neoattack.com/neowiki/front-end/, 2020. Ultimo acceso el 08/02/2022.

[38] Ministerio de empleo y Seguridad Social. Boletín oficial del estado español. https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf, Marzo 2018.

[39] Universidad de Valladolid. Plan de estudios - ingeniería informática. https://www.uva.es/export/sites/uva/2.docencia/2.01.grados/2.01.02.ofertaformativagrados/_documentos/inginformatica_distribucion.pdf. Ultimo acceso el 15/03/2022.

[40] Claire Drumond. Scrum, aprende a utilizar lo mejor de él. https://www.atlassian.com/es/agile/scrum. Ultimo acceso el 07/03/2022.

[41] Firebase. https://firebase.google.com/. Ultimo acceso el 09/02/2022.

[42] Ibon García. qué es el modelo-vista-controlador. https://carontestudio.com/blog/que-es-modelo-vista-controlador/, noviembre 2021. Ultimo acceso el 13/06/2022.

[43] Mike Hughes, Bob I& Cotterell. *Software Project Management*. Second edition, 1999.

[44] Project Management Institute. *PMBOK Guide*.

[45] Javier Santos Pascualena. ¿cuánto cuesta contratar un trabajador? https://www.infoautonomos.com/blog/cuanto-cuesta-contratar-un-trabajador/, 2021. Ultimo acceso el 13/06/2022.

[46] Mati Presta. Los 10 mejores marcos frotnend y backend. https://blog.back4app.com/es/los-10-mejores-marcos-de-frontend-y-backend/, 2022. Ultimo acceso el 08/02/2022.

[47] proyectosagiles.com. Qué es scrum. https://proyectosagiles.org/que-es-scrum/. Ultimo acceso el 07/03/2022.

[48] proyectosagiles.com. Requisitos para poder utilizar scrum. https://proyectosagiles.org/requisitos-de-scrum/. Ultimo acceso el 09/03/2022.

[49] S.A. Prozis.com. Prozis, tienda de alimentación deportiva. https://www.prozis.com/. Ultimo acceso el 08/02/2022.

[50] Juan Diego Pérez Jiménez. Qué es css y sus fundamentos. https://openwebinars.net/blog/que-es-css3/. Ultimo acceso el 08/06/2022.

[51] MongoDB Realm. https://www.mongodb.com/realm. Ultimo acceso el 09/02/2022.

[52] Jeff Schwaber, Ken ; Sutherland. The 2020 scrum guide. https://scrumguides.org/scrum-guide.html, 2020. Ultimo acceso el 07/03/2022.

[53] Amazon Web Services. https://aws.amazon.com/es/amplify/. Ultimo acceso el 09/02/2022.

[54] Kubekings S.L. Kubekings.com — tu tienda especializada en el cubo de rubik. https://kubekings.com/. Ultimo acceso el 08/02/2022.

[55] The Vegan Society. The vegan society. https://www.vegansociety.com/go-vegan/definition-veganism, 1988. Ultimo acceso 06/02/2022.

[56] Yasmani Tápanes. Los 6 mejores frameworks frontend. https://saasradar.net/mejores-framework-frontend/, Enero 2020. Ultimo acceso el 08/02/2022.

[57] PC Componentes y Multimedia SLU. Pccomponentes.com — tienda informática y tecnología online. https://www.pccomponentes.com/. Ultimo acceso el 08/02/2022.

Anexos

Anexo I

Manual de instalación para desarrolladores

Para facilitar el uso de el software se facilita el siguiente manual de preparación del entorno de desarrollo.

I.1. Requisitos previos a la instalación

Para poder preparar correctamente el entorno de desarrollo se requiere tener cierto software instalado en el equipo en el que se vaya a preparar el entorno.

Este software es:

- `Node.js` : versión `17.9.0` o superior.
- `npm`: versión `8.9.0` o superior.
- `git`: versión `2.25.1` o superior, si se desea clonar el repositorio.

I.2. Instalación

Para preparar el entorno es necesario seguir los siguientes pasos:

- Descomprimir los archivos del zip "`app-src`" en una carpeta o clonar el repositorio mediante git vía el siguiente comando:

```
$ git clone  https://gitlab.inf.uva.es/paborte/tfg.git
```

- Ejecutar en la carpeta `tfg-master`(la cuál esta dentro de la carpeta descomprimida) el comando

```
$ npm install
```

el cuál instala en la carpeta todas las dependencias del proyecto.

Con esto tendríamos preparado el entorno de desarrollo. Se recomienda abrir el entorno con un editor de código como Visual Studio Code.

I.3. Comandos

- `ng serve`: crea un servidor de desarrollo en el que se despliega nuestra aplicación. Este se actualiza cada vez que hay un guardado en alguno de los archivos del proyecto. Levanta un servidor en el puerto 4200 de nuestra máquina local, por lo tanto es accesible vía `localhost:4200/`.

- `ng build`: crea una versión optimizada y empaquetada de la aplicación para poder desplegarla de forma simple.

Anexo II

Manual de despliegue

Para poder desplegar el sistema, de la forma en la que se ha propuesto en este proyecto es necesario generar un artefacto. Para generar este artefacto es necesario ejecutar el comando "`ng build`", el cuál nos crea un artefacto desplegable en la carpeta "`/dist`". Para poder desplegar este artefacto en Firebase Hosting es necesario poseer unua cuenta de Google y haber inicializado un proyecto de Firebase. Dentro del propio Firebase existe un tutorial de como crear un proyecto, el cuál se muestra nada más acceder al servicio.

Una vez creado el proyecto Firebase es necesario tener montado el entorno de desarrollo para poder desplegar el artefacto. También es necesario instalar mediante `npm` la suite de herramientas de Firebase o CLI, `firebase-tools`. Este CLI puede ser instalado mediante el comando

```
$ npm install -g firebase-tools
```

el cuál instala el paquete de forma global en el sistema.

Tras esto es necesario iniciar sesión en nuestra cuenta de Google en el CLI de Firebase mediante el comando

```
$ firebase login
```

el cuál abre una ventana de navegador que nos permite iniciar sesión en nuestra cuenta de una manera simple.

Una vez iniciado el proyecto Firebase y teniendo iniciada sesión es necesario inciar el servicio de hosting desde el CLI de Firebase, para ello ejecutamos el comando

```
$ firebase init hosting
```

el cuál nos pedirá un proyecto para iniciar el hosting y debemos seleccionar el proyecto Firebase incializado anteriormente. También nos pedirá un directorio para usar como raíz del proyecto a desplegar. Aquí debemos indicarle "`/dist/tienda-chocolates-veganos`çomo nuestro directorio raíz. Posteriormente nos preguntará si queremos configurar el proyecto para desplegar una single-page-application, a lo que debemos contestar que sí.

Toda esta secuencia configura el archivo "`firebase.json`.[el] cuál debe quedarnos así:

```
{
"hosting": {
    "public": "dist/tienda-chocolates-veganos",
    "ignore": ["firebase.json", "**/.*", "**/node_modules/**"],
    "rewrites": [
        {
        "source": "**",
        "destination": "/index.html"
            }
        ]
    }
}
```

Con esto ya podemos ejecutar el comnado

```
$ fireabse deploy
```

El cuál compara los archivos del artefacto previamente desplegado y despliega los cambios en una red de contenidos global accesible mediante el enlace que provee el CLI al completar el proceso. En caso de que sea la primera vez que se despliega, despliega todo el artefacto.