Tienda Virtual Qkdas Art

Memoria de Proyecto Final de Máster

Desarrollo de Sitios y Aplicaciones Web

Área de Informática, Multimedia y Comunicación

Autor: Xavier Ferrarons Serra

Consultor: Víctor Cuervo

Profesor: Carlos Casado Martínez

Enero de 2021

Créditos/Copyright

Abstract

El principal objetivo de este proyecto es el de desarrollar una aplicación web que permita la publicación de un catálogo de productos así como poder realizar posteriormente su venta.

Para facilitar la clasificación de estos productos se puede definir distintos niveles de agrupación. Los productos se definen mediante una referencia, descripción, precio, stock y una galería de imágenes. Por su lado, los visitantes pueden registrarse y realizar la compra de aquellos productos que desee. Para efectuar el pago de estas compras se utilizará una pasarela de pago que va a permitir pagar mediante tarjeta de crédito. Finalmente, tanto compradores como vendedores tienen acceso a los pedidos resultantes de las compras para así por consultar su contenido y realizar la expedición correspondiente.

La aplicación tiene un diseño *responsive* adaptándose a los distintos dispositivos existentes en el mercado que tienen anchos de pantalla y otras características diferenciales.

Para el desarrollo del *front-end* se ha utilizado el *framework* Angular. Por otro lado, para el desarrollo del *back-end* se hace uso de distintos servicios facilitados por la plataforma Firebase. La aplicación web se integra con la pasarela de pago Stripe.

Palabras clave: Aplicación, web, catálogo, venta, comprador, vendedor, Angular, Firebase.

Abstract (english version)

The main objective of this project is to develop a web application that allows the publication of a product catalog, as well as being able to sell them.

In order to facilitate the classification of these products we can define different level groups. The products are defined by a reference, description, price, stock and an image gallery. On the other hand, visitors are able to register and make the purchase of those products they want. To make the payment of those products a payment gateway is used, allowing users pay by credit card. Finally, buyers and sellers will have access to the orders resulting from purchases in order to consult their content and carry out their expedition.

The application has a responsive design adapting to the different devices that exist I in the market that have different screen widths and other differential characteristics.

For the front-end development the Angular framework has been used. On the other hand, for the development of the back-end different services provided by the Firebase platform have been used. The web application is integrated with the Stripe payment gateway.

Keywords: Application, web, catalog, sell, buyer, seller, Angular, Firebase.

Índice.

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto y justificación del trabajo

En este proyecto se quiere dar soporte a una emprendedora que se encarga de realizar bisutería artesanal y facilitarle el trabajo de creación de su catálogo así como la posterior venta de sus artículos.

Tradicionalmente la venta de productos artesanales se realizaba en mercadillos y ferias. Sin embargo, la rápida evolución que ha sufrido en los últimos tiempos el sector tecnológico ha hecho que cada vez más usuarios tengan acceso a dispositivos que permiten la navegación web así como la compra *online.* La venta de productos artesanales también ha tenido que adaptarse a esta nueva realidad.

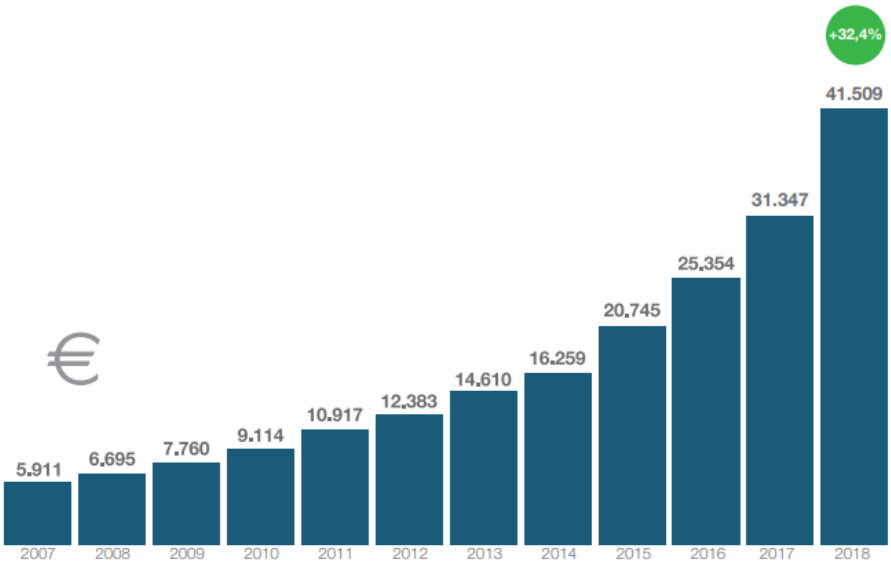


Figura 1: Evolución comercio electrónico en España (El Observatorio Cetelem, 2020)

En la actualidad existen dos tendencias:

- Marketplaces
- Webs propias

Marketplace

Un Marketplace es un **centro comercial online** que proporciona espacio a varios vendedores para mostrar sus productos. En esta "tienda de tiendas" intervienen tres partes, los compradores, los vendedores y la plataforma que permite que se relacionen, y donde se efectúa la transacción de pago y se encarga del envío del pedido.

Algunos de los *marketplaces* referentes en el sector de la artesanía son:

- Etsy es un mercado online en el que, a nivel internacional, se compran y venden productos de artesanía. Como empresa, ETSY no tiene productos propios ni almacenes; solo aportan la infraestructura para conectar a los artesanos con las personas interesadas en sus productos. En esta plataforma, podemos encontrar todo tipo de vendedores, desde artesanos independientes, hasta comerciantes. Esta plataforma está claramente enfocada a pequeñas y medianas empresas, con presupuestos ajustados, que quieren acceder al mercado online de una forma sencilla.
- Amazon Handmade Con esta plataforma, Amazon quiere competir directamente con ETSY, que en los últimos años, ha dominado el mercado de la artesanía online. Para ello, Amazon Handmade ofrece 3 servicios adicionales de los que no dispone su competencia: La posibilidad de personalizar el producto, el contacto directo del cliente con el artesano, dando la posibilidad de conocer la historia del creador a través de una ficha; y, por último, la garantía para el cliente de que está comprando un producto auténticamente hecho a mano, certificado mediante un proceso de verificación. Además, otro punto a favor de Amazon, es su amplia experiencia en el comercio online, que aumenta la confianza de compradores y vendedores.

Ventajas Marketplace

- **Ahorro de costes:** ellos ponen toda la infraestructura y te cobran una cuota mensual y/o una comisión por cada venta.
- **Posicionamiento:** Te olvidas del SEO y de invertir en publicidad online. A los *marketplaces* les interesa mucho estar bien posicionados para ser atractivos tanto para vendedores como para clientes finales.
- **Sinergias con productos de otros vendedores.** Si vendes cinturones, te interesa mucho estar en un sitio que también venda pantalones.
- **Gestión de pagos**: Los clientes pagan, en la mayoría de los casos, directamente al *marketplace* que luego te paga a ti. Ahorra costes bancarios.
- **Muchos *marketplaces* ofertan la logística** como parte de sus servicios.

Desventajas Marketplace

- **Más competencia**: Pone a tus competidores a un clic de distancia, sin salir si quiera de la misma web.
- **Te obliga a ajustar y revisar precios constantemente**. Y a compararlos con tus competidores.
- **No te deja libertad** para mostrar tus productos como tú quieras, hay que ajustarse a sus normas.
- **Es muy complejo crear marca** si estás en un *marketplace*. Su marca siempre estará por encima de la tuya, a no ser que la tuya sea lo suficientemente potente como para que el cliente la busque.
- **Puede llegar a ser muy costoso** si tu volumen de ventas es alto.

Web propia

La realización de una web propia requiere una inversión inicial para poder sufragar los costes iniciales de creación de la web, así como un trabajo constante de promoción y mantenimiento del sitio. No obstante, una de sus principales ventajas es que **podemos influir en su diseño y crear una imagen de marca**.

Así pues, podemos concluir que un *marketplace* es el lugar ideal para empezar a vender *online* sin realizar apenas desembolso económico y desentendiéndonos de toda la infraestructura tecnológica, y nos puede servir para testear la aceptación de nuestros productos en el mercado. Pero si lo que queremos es empezar a consolidar nuestra marca, como es el caso de nuestro cliente, **la mejor opción es el desarrollo de una web propia**.

2. Objetivos

A continuación se detallan los objetivos que persigue la aplicación. Se ha diferenciado entre principales (son necesarios para conseguir el objetivo del TFM) y secundarios (sería interesante realizarlos, pero dependerá de cómo vaya avanzando el proyecto).

2.1. Principales

- El vendedor debe poder publicar su catálogo de productos.
- El visitante debe poder consultar el catálogo.
- El visitante debe poder comprar los productos que desee.
- El visitante debe poder pagar informando una tarjeta de crédito.
- El vendedor debe tener un *feedback* de los productos que más gustan a los visitantes.
- El vendedor debe poder gestionar la expedición de los pedidos.
- Diseño web *responsive.*

2.2. Secundarios

- Gestionar las formas de pago de los usuarios para que no deban introducir los datos en cada nueva compra.
- Realización de pagos mediante Paypal.
- Posibilidad de establecer un chat entre comprador y vendedor para poder personalizar las ventas.
- Localización de los pedidos en tránsito.

3. Metodología

3.1. Metodología de Desarrollo

El sistema de entregas establecido para la ejecución de este proyecto hace que la metodología de desarrollo de *software* que más se ajusta a nuestras necesidades sea el modelo en cascada también conocido como *waterfall.*

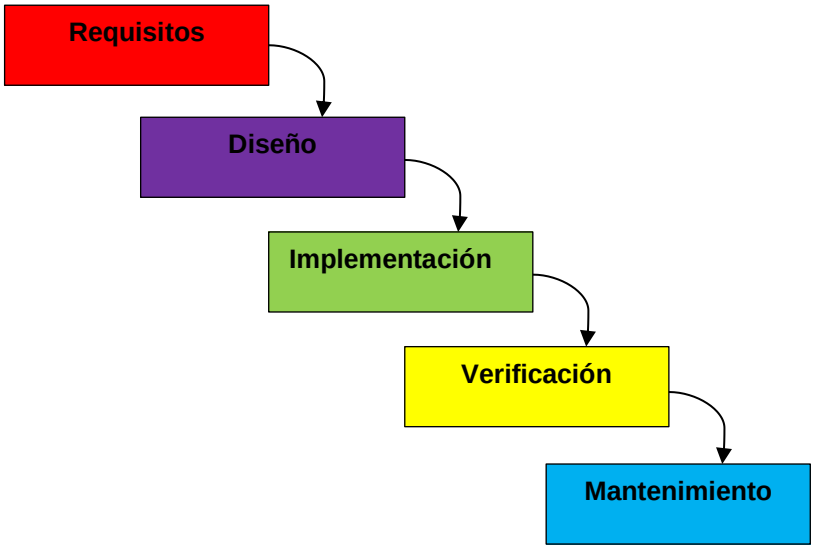


Figura 2: Ciclo de vida de desarrollo en cascada

En esta metodología, cada fase del ciclo de vida no debe comenzar hasta que esté totalmente terminada la fase anterior. Una fase se considera finalizada cuando se dispone del entregable final de dicha fase.

En este proyecto se puede realizar un mapeo entre los entregables a realizar y las distintas fases del ciclo de vida.

Etapa	Entregable
Requisitos	PEC1
Diseño	PEC2
Implementación	PEC3
Verificación y Mantenimiento	PEC4

Tabla 1: Relación entregas y fases ciclo de vida.

Esta metodología se ajusta especialmente en proyectos estables con requisitos no cambiantes y donde es posible predecir posibles problemas en la fase de diseño y actuar en consecuencia. En nuestro caso, el producto a desarrollar es un tipo de producto ya conocido en el mercado por lo que los requisitos están

bastante claros y no se prevén cambios durante la ejecución del proyecto por lo que esta metodología se ajusta al tipo de proyecto a desarrollar.

En la etapa de diseño se va a seguir una filosofía de diseño centrado en el usuario. Su premisa es que para garantizar el éxito de un producto hay que tener en cuenta al usuario en todas las fases del diseño.

El diseño centrado en el usuario se basa en un modelo de etapas que se suceden y retroalimentan constantemente.

Las etapas de este modelo son:

1. **Análisis:** conocer en profundidad los usuarios finales.
2. **Diseño:** Diseñar un producto que resuelva sus necesidades y se ajuste a sus capacidades, expectativas y motivaciones.
3. **Evaluación:** Poner a prueba lo diseñado, normalmente usando test de usuarios.

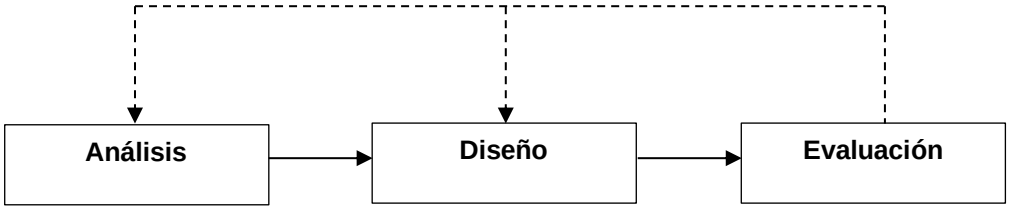


Figura 3: Modelo iterativo diseño centrado usuario (DCU)

3.2. Estrategia de desarrollo

El objetivo final es crear un producto consistente que permita a sus usuarios una experiencia continua y complementaria a través del ecosistema de dispositivos con los que interactúan diariamente.

Los usuarios, a lo largo del día, interactúan con dispositivos de tamaños y modos de interacción muy variados. A pesar de ello, debe buscarse una experiencia muy similar en todos los dispositivos ofreciendo las mismas funcionalidades principales.

En nuestros diseños debe tenerse en cuenta que los usuarios pueden empezar la interacción en un determinado dispositivo y acabar interactuando con otro muy distinto. Es por ello que debe ofrecerse una experiencia continua que permita enlazar procesos que empiezan y acaban en dispositivos diferentes.

Finalmente, aún y ofrecer las mismas funcionalidades principales en los diferentes dispositivos, hay que aprovechar al máximo las características de cada dispositivo.

En los últimos años, el teléfono móvil ha ido ganando importancia frente a otros dispositivos aunque no por ello debe menospreciarse los otros dispositivos. Es por ello que va a realizarse un diseño web *Responsive* en que nuestro diseño se ajuste a los distintos tamaños de pantalla, ya sea la de ordenadores, tabletas o teléfonos móviles.

Al ser los dispositivos móviles los dispositivos que se usan más para navegar por internet se va a seguir una estrategia de diseño *Mobile First*.

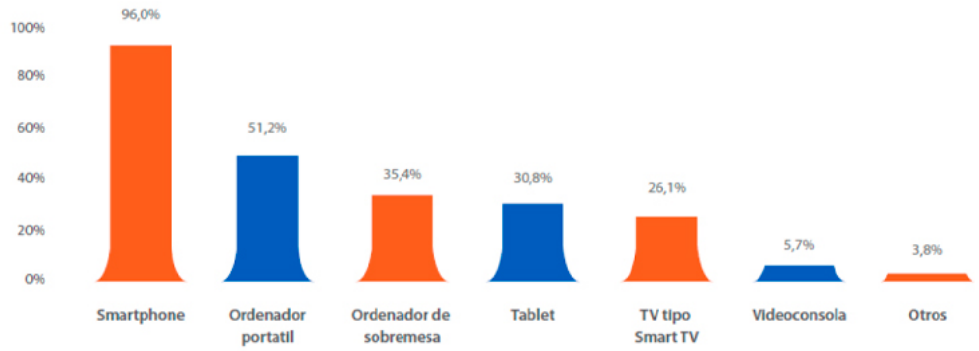


Figura 4: Tasa penetración navegación internet según el dispositivo utilizado en 2019 (Hernández, 2019)

Figura 5: Porcentaje usuarios que compran desde su móvil

El diseño *Mobile First* empieza el diseño pensando en las pantallas más reducidas para eliminar la información innecesaria y que las personas que accedan a la página web tengan una experiencia de contenidos y de velocidad de carga óptima Una vez realizado el diseño para los teléfonos móviles, se va trabajando progresivamente hacia la página web que podremos ver en una Tablet, un ordenador portátil…

Finalmente, nuestra web tiene pensado presentarse como una PWA (*progressive web application*). Una PWA debe cumplir las siguientes propiedades:

- **Progresiva**: debe funcionar para todos los usuarios, sin importar la elección del navegador.
- **Adaptable**: se adapta a cualquier factor de forma, es decir, debe poder ejecutarse tanto en escritorio, móvil, tablet o lo que venga en el futuro.
- **Independiente de la conectividad**: la conectividad del dispositivo no debe ser un limitante, sino que se debe poder disponer de la misma experiencia de usuario tanto si se dispone de conectividad a internet alta, media o baja.

- **Estilo app**: la experiencia de usuario debe proporcionar un *look & feel* como el de una app nativa con interacciones y estilo de navegación clásico de una aplicación móvil.
- **Fresca**: siempre se encuentra actualizada, al ser una aplicación web que está constantemente actualizada.
- **Segura**: solo funciona sobre HTTPS.
- **Instalable:** debe poderse instalar en los dispositivos de modo fácil.
- **Vinculable**: se puede compartir fácilmente proporcionando la URL, no requiere instalación compleja, ni descarga de datos para ello.

En resumen, las aplicaciones web progresivas son una evolución natural de las aplicaciones web que **difuminan la barrera entre la web y las aplicaciones**, pudiendo realizar tareas que generalmente solo las aplicaciones nativas podían llevar a cabo.

4. Plataforma de desarrollo

El hardware que se va a utilizar tiene las siguientes especificaciones:

MacBook Pro 15''
2,2 GHz Intel Core i7
16 GB 1600 MHz DDR3
SO 10.15.6

El desarrollo de este proyecto se va a realizar con la ayuda del siguiente software:

- **IDE:** Webstorm 2019.1
- **Edición memoria:** Microsoft Office
- **Gantt:** GanttProject
- **Diseño gráfico:** Gimp
- **Navegadores:** Google Chrome y Mozilla Firefox for Developers
- **Control de Versiones:** SourceTree

Y también se va a hacer uso de las siguientes *web apps:*

- **Creación *wireframes:*** Mockflow.com
- **Diagramas:** Draw.io
- **Control de versiones:** Github
- **Servidor:** Firebase
- **Copias seguridad documentación:** Google Drive

Para el desarrollo del *front-end* se va a hacer uso de las siguientes tecnologías: HTML5, CSS3, SASS, JavaScript y TypeScript a través de Angular.

Finalmente para el desarrollo del *backend* se va a hacer uso los siguientes productos de Firebase:

- *Firestore* como noSQL database
- *Auth:* como autotentificador de usuarios.
- *Storage* como almacén de imágenes
- *Cloud Functions* como activador de funciones.

5. Planificación

5.1. Planificación Temporal

La planificación de este trabajo viene determinada por cuatro fechas clave correspondientes a las 4 PEC's que deben entregarse durante la ejecución del proyecto. Dentro de cada una de estas PEC's debe realizarse una serie de tareas.

En esta fase inicial del proyecto todavía no se ha realizado una planificación detallada de la fase de implementación. Una vez definida la fase de diseño podrá realizarse un desglose detallado de las funcionalidades a implementar así como realizar su planificación.

Para realizar la planificación se ha considerado que el trabajo será realizado por un único desarrollador que dispone un promedio de 3h diarias de lunes a domingo. En caso de producirse imprevistos, hay que tener en cuenta que durante los fines de semana, en caso de ser necesario, podría dedicarse más tiempo.

A continuación pasa a detallarse las tareas a realizar en cada fase así como sus fechas de inicio y fin junto con la carga de horas prevista.

Tarea	Fecha inicio	Fecha fin	Carga (h)
PEC1 – Plan de trabajo	**16/09/20**	**29/09/20**	**42**
Contexto y justificación del trabajo	16/09/20	18/09/20	9
Objetivos del trabajo	19/09/20	21/09/20	9
Enfoque y método seguido	22/09/20	23/09/20	6
Planificación del trabajo	24/09/20	25/09/20	6
Documentación	26/09/20	29/09/20	12
PEC2 – Diseño	**30/09/20**	**28/10/20**	**87**
Usuarios y contextos de uso	30/09/18	02/10/20	9
Diseño conceptual	03/10/20	05/10/20	9
Prototipado	06/10/20	10/10/20	15
Evaluación	11/10/20	15/10/20	15
Definición de los casos de uso	16/10/20	20/10/20	15
Diseño de la arquitectura	21/10/20	23/10/20	9
Documentación	24/10/20	28/10/20	15
PEC3 – Implementación	**29/10/20**	**06/12/20**	**117**
Desarrollo	29/10/207	29/11/20	96
Pruebas	30/11/20	03/12/20	12
Documentación	04/12/20	06/12/20	9
PEC4 – Entrega Final	**07/12/20**	**04/01/21**	**87**
Pruebas	07/12/20	15/12/20	27

Memoria	16/12/20	25/12/20	30
Presentación	25/12/20	03/01/21	30
Defensa Virtual	**11/01/21**	**15/01/21**	

Name	Begin date	End date	Dura...
PEC1 - Plan de Trabajo	16-09-2020	29-09-2020	14
Contexto y justificación del trabajo	16-09-2020	18-09-2020	3
Objetivos del trabajo	19-09-2020	21-09-2020	3
Enfoque y método seguido	22-09-2020	23-09-2020	2
Planificación del trabajo	24-09-2020	25-09-2020	2
Documentación	26-09-2020	29-09-2020	4
PEC2 - Diseño	30-09-2020	28-10-2020	29
Usuarios y contextos de uso	30-09-2020	02-10-2020	3
Diseño conceptual	03-10-2020	05-10-2020	3
Prototipado	06-10-2020	10-10-2020	5
Evaluación	11-10-2020	15-10-2020	5
Definición de los casos de uso	16-10-2020	20-10-2020	5
Diseño de la arquitectura	21-10-2020	23-10-2020	3
Documentación	24-10-2020	28-10-2020	5
PEC3 - Implementación	29-10-2020	06-12-2020	39
Desarrollo	29-10-2020	29-11-2020	32
Pruebas	30-11-2020	03-12-2020	4
Documentación	04-12-2020	06-12-2020	3
PEC4 - Entrega Final	07-12-2020	03-01-2021	28
Pruebas	07-12-2020	15-12-2020	9
Memoria	16-12-2020	25-12-2020	10
Presentación	25-12-2020	03-01-2021	10
Defensa Virtual	11-01-2021	15-01-2021	5

Figura 6: Diagrama Gantt planificación proyecto

5.2. Riesgos

En todo proyecto existen una serie de riesgos que en caso de producirse pueden conducir a su fracaso. Para evitar esta situación y anticiparnos a los mismos, se ha definido un plan de riesgos donde se detalla la probabilidad de que ocurran, el impacto que tienen sobre el proyecto y las acciones previstas para mitigar su impacto.

Conocimientos insuficientes

En el desarrollo de este proyecto se tiene previsto el uso de tecnologías y herramientas de las que no se tiene un gran conocimiento por no haberse utilizado en profundidad anteriormente. Esta situación es altamente probable que ocurra y puede producir retrasos en la ejecución del proyecto, principalmente en la etapa de implementación. El impacto sobre el proyecto es bastante elevado por lo que se han previsto una serie de medidas correctoras.

En previsión a esta situación, tiempo antes del inicio de este proyecto se ha empezado a investigar y practicar en el uso de la base de datos noSQL de Firebase que es la que quiero utilizar en este proyecto. Se ha experimentado con distintas funcionalidades que se han considerado interesantes para la ejecución del proyecto.

La carga de horas de trabajo de los fines de semana y festivos no es del 100%. Así pues, en el caso de producirse imprevistos, se tiene un cojín de horas.

Mala planificación del proyecto

La falta de experiencia en proyectos de desarrollo de app's puede haber producido errores en la estimación de horas para la realización de ciertas tareas. Esta situación es altamente probable que ocurra y puede producir retrasos en la ejecución del proyecto. El impacto sobre el proyecto es bastante elevado por lo que se han previsto una serie de medidas correctoras.

Aunque no existen dos proyectos iguales, se ha revisado proyectos similares para obtener conocimiento sobre su planificación. En la fase de implementación, las funcionalidades se irán implementando por orden de importancia teniendo siempre como objetivo obtener lo antes posible una web utilizable con unas funcionalidades mínimas. De esta forma se pretende obtener un entregable lo antes posible. Al igual que en el riesgo anterior, también se dispone de la posibilidad de aumentar la carga de trabajo durante los fines de semana y festivos.

Capacidad de dedicación reducida

Por motivos externos tales como enfermedad o cambio responsabilidades, la dedicación de horas diarias al proyecto puede verse reducida. El impacto sobre el proyecto es medio por lo que en el momento de la planificación se ha tenido en cuenta.

Nuevamente hay que remitirse a las medidas establecidas en los dos riesgos anteriores.

Fallos de software i/o hardware

Errores de software y hardware pueden provocar la pérdida del proyecto. La probabilidad que esto ocurra no es muy alta aunque el impacto sobre el proyecto sería alto.

Para evitar este riesgo se ha establecido el uso de Google Drive para tener copias de seguridad de toda la documentación relacionada con el proyecto. En lo referente al código, se va a hacer uso de *Source Tree* para realizar un control de los cambios y al mismo tiempo disponer de una copia de seguridad del código fuente.

Finalmente, también se dispone de un segundo ordenador de sobremesa que dispone de todas las herramientas necesarias para la ejecución del proyecto.

6. Arquitectura

Para el desarrollo de nuestra aplicación web se ha seguido el patrón de diseño MVC (Modelo-Vista-Controlador). Este patrón separa los datos y lógica de negocio de su representación mediante el uso de tres tipos de componente:

- **Modelo**: representación de la información con la que el sistema opera. Gestiona los accesos a la información tanto en modo consulta como actualizaciones. Envía a la *vista* aquella información que se solicita en cada momento. Las peticiones de información llegan al *modelo* a través del *controlador.* En este proyecto el modelo será gestionado externamente por *Firebase.*
- **Controlador**: responde a eventos, normalmente provocados por el usuario e invoca peticiones al *modelo* cuando se hace alguna solicitud de información. Es aquí donde se implementa toda la lógica de negocio. En este proyecto los controladores se implementan en *typescript* mediante el *framework* Angular.
- **Vista**: a través de la interfaz de usuario se encarga de presentar el *modelo* en el formato adecuado. En este proyecto las vistas se definirán mediante ficheros *html* y *scss*.

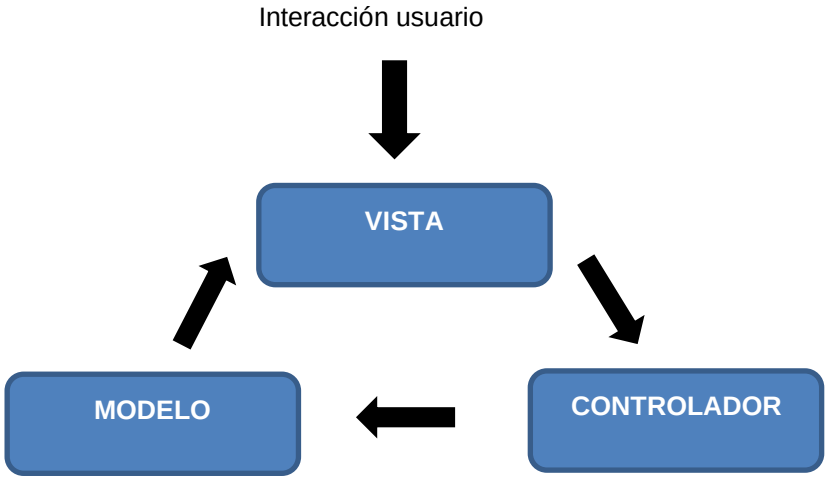


Figura 7: Patrón de diseño MVC

Para implementar este patrón de diseño nuestra aplicación va a hacer uso de 2 servidores.

- **Servidor Firebase:** servidor donde se aloja la aplicación y la base de datos que en nuestro caso será una base de datos noSQL. Firebase ofrece también otros servicios que también se utilizarán para nuestro desarrollo y que se detallan más adelante.
- **Servidor Stripe:** pasarela de pago que nos permite realizar los pagos en nuestra aplicación web.

i

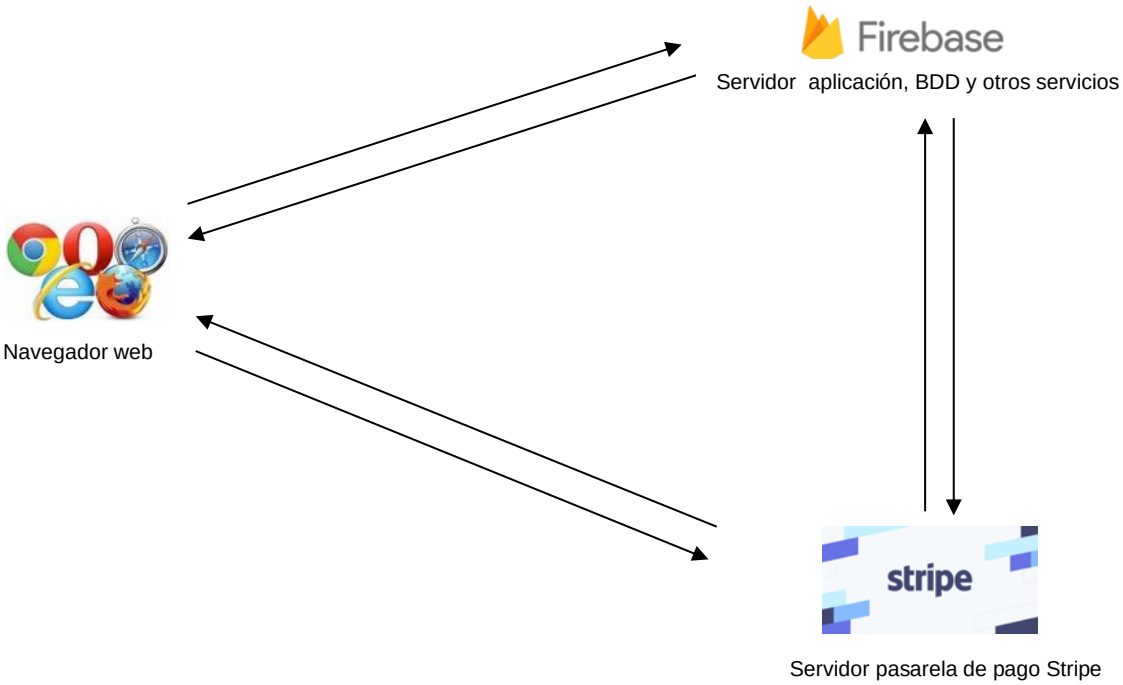


Figura 8: Arquitectura servidores

6.1. Aplicación web

El desarrollo de la aplicación web se va a realizar mediante el *framework* de *typescript* Angular 10. En este tipo de desarrollo se define un conjunto de componentes aislados que luego serán montados. La aplicación se cargará en una sola URL (SPA)[1] y posteriormente por medio de su *router* se podrá ir navegando por los distintos componentes. Uno de los inconvenientes de las *SPA* es que la primera carga puede tardar un poco ya que debe descargarse todos los códigos de *HTML*, *JavaScript* y *CSS*.

Lazy Loading

Para solventar el problema de la carga inicial se hará uso de la funcionalidad de Angular *Lazy Loading Module*. De esta manera los módulos se van cargando a medida que se va haciendo uso de ellos. Sin embargo, esta

[1] SPA: Una **single-page application** (**SPA**), o aplicación de página única, es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se cargan una sola vez[1] o los recursos necesarios se cargan dinámicamente cuando lo requiera la página.

opción también puede crear ciertos tiempos de demora en la carga de un determinado módulo si éste tiene mucho peso.

Para evitar también esta demora en la carga de un determinado módulo, Angular permite el uso de *Lazy Loading Module* junto con la opción *Preload.* Con esta opción Angular permite, una vez cargado el módulo inicial, carga de forma asíncrona y totalmente de forma transparente para el usuario el resto de módulos.

En nuestro desarrollo se ha optado por seguir el patrón de diseño ***Lazy Loading* sin la opción *Preload.***

Redux

En nuestra aplicación también se va a aplicar el patrón *Redux*. Este patrón nos permite conocer en cada momento el estado de nuestra aplicación. Pero no sólo eso, sino que también nos permite ver como ha ido evolucionando paso a paso y que acciones han provocado esos cambios.

Para ello *Redux* se base en 3 ideas principales:

- **Sólo existe una fuente de verdad:** es decir toda la información que usa nuestra aplicación se encuentra en una estructura definida previamente y se encontrará almacenada en un único lugar llamado *Store*.
- **El estado (*Store*) es de solo lectura**: El estado de nuestra aplicación representa el valor actual de la misma. La única manera de modificar el estado de nuestra aplicación será por medio de acciones, ya que las acciones son objetos planos que pueden ser registrados, serializados y almacenados para volver a ejecutarlos por cuestiones de depuración y pruebas.
- **Los cambios se realizan con funciones puras:** Este principio nos dice que un nuevo estado es creado en base en un estado anterior y una acción, y devuelve un nuevo estado.

La implementación de este patrón quizás pueda resultar algo compleja para aplicaciones sencillas. Sin embargo, su uso en aplicaciones medianas y grandes permite desacoplar la gestión del estado de los componentes centralizándolo en un único sitio y facilitando así la escalabilidad de la aplicación.

6.2. Firebase

Toda la gestión del *backend* se ha delegado a Firebase. Firebase es una plataforma creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad y de forma rápida. La plataforma se encuentra en la nube y está disponible para distintas plataformas (Android, iOS, Web). En este proyecto va a hacerse uso de las siguientes funcionalidades:

- **Cloud Firestore:** base de datos NoSQL flexible, escalable y ubicada en la nube. En este proyecto va a utilizarse Cloud Firestore como base de datos.

- **Authentication**: permite realizar la autentificación de los usuarios en las aplicaciones. La autentificación puede realizarse mediante distintas vías (contraseñas, números de teléfono, Google, Facebook
- **Cloud Storage**: sistema de almacenamiento de archivos. Los archivos pueden cargarse sin importar las condiciones de red, asíncronamente, y si una subida se interrumpe, se reanuda automáticamente cuando vuelve la conexión, continuando por donde se había parado. En este proyecto se utilizará *Storage* para almacenar las imágenes.
- **Cloud Functions:** permite ejecutar de forma automática el código en el *backend* en respuesta a distintos eventos. En nuestro proyecto entre otros se utilizará para gestionar los pagos, redimensionar las imágenes subidas al servidor o ejecutar ciertos disparadores ante eventos de la BDD.

6.3. Stripe

Para gestionar los pagos mediante tarjeta de crédito se ha elegido la pasarela de pago *Stripe*. De esta forma se conecta nuestra aplicación web con una cuenta de Stripe que será la encargada de recibir todos pagos que se realicen.

El flujo de trabajo para realizar un pago en Stripe es el siguiente:

- Al iniciarse un pago, inicialmente se solicita a *Stripe* un *id* de sesión de pago. Esta solicitud se recomienda hacer desde el servidor. En éste puede realizarse el cálculo del importe a pagar evitando así posibles ataques. Para activar este proceso se hará uso de *Cloud Functions*.
- El servidor de *Stripe* devuelve un *id* de sesión de pago que tendrá asociado un importe a pagar. Con este *id* de sesión de pago el dispositivo se conecta directamente al servidor de Stripe y como respuesta se obtiene el importe a pagar y un formulario para introducir los datos de pago. En este proceso se está delegando a Stripe la gestión y confidencialidad de los datos de pago así como la validación de los mismos. Finalmente, si todo es correcto se procede a iniciar el pago.
- Una vez finalizado el pago, puede recibirse en el servidor de Firebase mediante *Cloud Functions* la confirmación del pago y realizar las acciones oportunas.

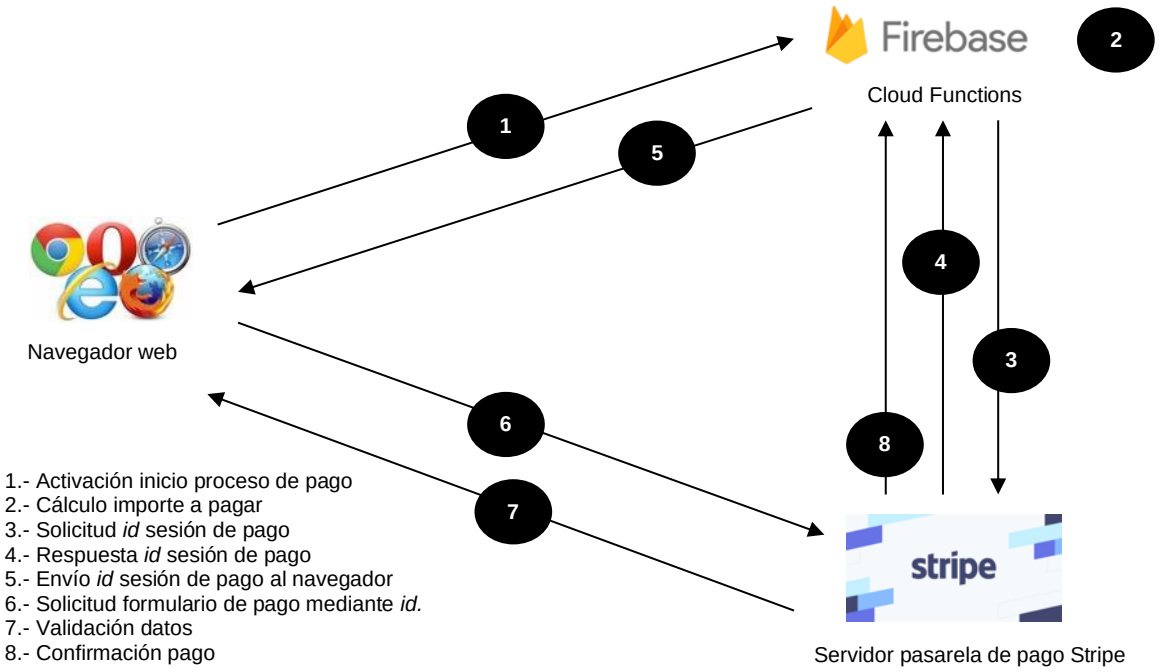


Figura 9: Flujo de datos en el proceso de pago.

7. Contenidos

Nuestra aplicación web está estructurada en tres secciones claramente diferenciadas:

- **Cabecera**: En ella se muestra el logo y nombre de la empresa. También se da acceso entre otros, al catálogo del vendedor, a la página de inicio y a la cesta de la compra. Finalmente, el usuario tiene acceso al login y/o registro. En el caso que el usuario se haya registrado y haya echo el login, tendrá acceso a su perfil. El contenido de la cabecera es *responsive* por lo que en función del ancho disponible algunas opciones pueden quedar escondidas en el menú “hamburguesa”.
- **Cuerpo**: En esta sección se muestran las distintas funcionalidades de nuestra aplicación web.
- **Pie**: Se dispone de acceso a las redes sociales del vendedor y se da información sobre el copyright y autoría de la aplicación.

A continuación pasa a realizarse una breve descripción de las distintas funcionalidades de nuestra aplicación web.

7.1. Página de Inicio

Al acceder a la aplicación web, inicialmente se presenta esta página en la que el vendedor hace una pequeña presentación.

7.2. Catálogo

En esta página se muestra los distintos productos del vendedor. Se muestra una pequeña imagen del producto junto a su precio. Cada producto tiene la posibilidad de ser añadido a la cesta de la compra. En el caso de pulsar sobre la imagen del producto, se abre un visor de imágenes en el que se ve la imagen principal del producto ampliada y en el caso de disponer de más imágenes, éstas se pueden ir visualizando.

En el catálogo únicamente aparecen los productos activos y de los que se dispone de stock.

7.3. Login

Desde el login el usuario tiene la posibilidad de identificarse en la aplicación mediante su email y contraseña. En el caso que no se haya registrado, el usuario tiene un enlace para poder darse de alta.

7.4. Perfil Usuario

Desde el perfil de usuario, los usuarios pueden consultar y/o modificar sus datos. Esta sección tiene dos secciones claramente diferenciadas:

- Datos personales

- Direcciones de envío

El usuario puede tener registradas varias direcciones de envío de entre las cuales una será la predeterminada. Al realizar un pedido, inicialmente se propone la dirección predeterminada aunque el usuario puede modificarla.

7.5. Agrupaciones

En vendedor debe definir las agrupaciones/subagrupaciones en las que clasifica sus productos. La aplicación permite una profundidad máxima de 4 niveles.

7.6. Listado Productos

Desde el listado de productos, el vendedor puede gestionar su catálogo dando de alta, modificando o eliminando los productos de su catálogo. Desde este mismo listado se tiene acceso al formulario de alta de productos. El producto viene definido por una referencia, descripción, stock y precio. Así mismo, también puede asociarle distintas imágenes. Como mínimo se debe definir una imagen y como máximo 4.

Para identificar rápidamente los productos de los que no se dispone de stock en el listado de productos aparecen en color rojo. Igualmente, para identificar rápidamente los productos que no están activos éstos aparecen en color gris.

7.7. Cesta

Como se ha comentado con anterioridad, los productos del catálogo pueden añadirse a la Cesta de la Compra. En esta cesta se encuentran los distintos productos que el usuario ha ido seleccionando pudiendo modificar la cantidad comprada. En esta cesta el usuario, también puede ver a cuanto asciende la compra. En cualquier momento se puede iniciar el proceso de compra. En este proceso, inicialmente se debe verificar que existe stock t confirmar la dirección de envío propuesta o bien modificarla. Una vez realizados estos pasos, se procede a efectuar el pago mediante tarjeta de crédito y a generarse el pedido correspondiente. Éste quedará en estado *pendiente de pago* hasta que no se confirme que el pago se ha realizado con éxito.

7.8. Listado Pedidos

Tanto el comprador como el vendedor tienen acceso al listado de pedidos. En el caso del vendedor, éste podrá visualizar todos los pedidos que le hayan realizado. Una vez confirmado su pago, se podrá proceder a la preparación del pedido y así poder realizar su expedición. El usuario, en todo momento, puede consultar el contenido de un pedido.

Por su parte el comprador puede consultar su histórico de pedidos pudiendo consultar su estado.

8. Diagramas UML

8.1. Diagrama de clases

Como ya se ha comentado, la base de datos que se va a utilizar es *Firestore* de la plataforma Firebase que es una base de datos noSQL[2]. Este tipo de base de datos tienen unas características que hay que tener en cuenta en el momento de realizar su diseño.

En *Firestore* los datos se almacenan en *documentos* que a su vez se almacenan en *colecciones.* Dentro de un documento se almacenan los atributos de los elementos pero también puede haber subcolecciones y objetos anidados.

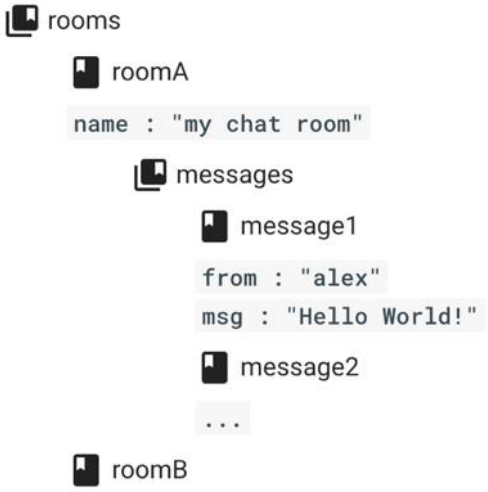


Figura 10: Ejemplo estructura de datos Firestore

En el ejemplo puede apreciarse que la colección *rooms* contiene los documentos *roomA*, *roomB*, … A su vez cada documento tiene el atributo *name* y una colección de *messages*. La colección de *messages* contiene los documentos *message1*, *message2*,… cada uno con sus atributos.

En este tipo de base de datos es aconsejable estructurar los datos pensando en la forma en que se van a realizar las consultas para así obtener el máximo rendimiento. Ésto, hace que en muchas ocasiones la base de datos no esté normalizada y algunos datos estén almacenados en más de una ubicación. Mediante este

[2] **NoSQL** (a veces llamado "no solo SQL") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no solo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.
D

diseño, los procesos de escritura se ven penalizados en rendimiento ya que una determinada operación de escritura puede tener que ejecutarse en varios sitios a la vez para así mantener todos los datos correctamente actualizados. Sin embargo los procesos de lectura podrán realizarse de forma más eficiente y obtener una mayor satisfacción por parte de los usuarios.

Antes de detallar como se van a estructurar nuestros datos en Firestore, se muestra mediante un diagrama de clases UML como se relacionan las distintas entidades que intervienen en el proyecto. Este diagrama permite tener una visión general de cómo se relaciona la información.

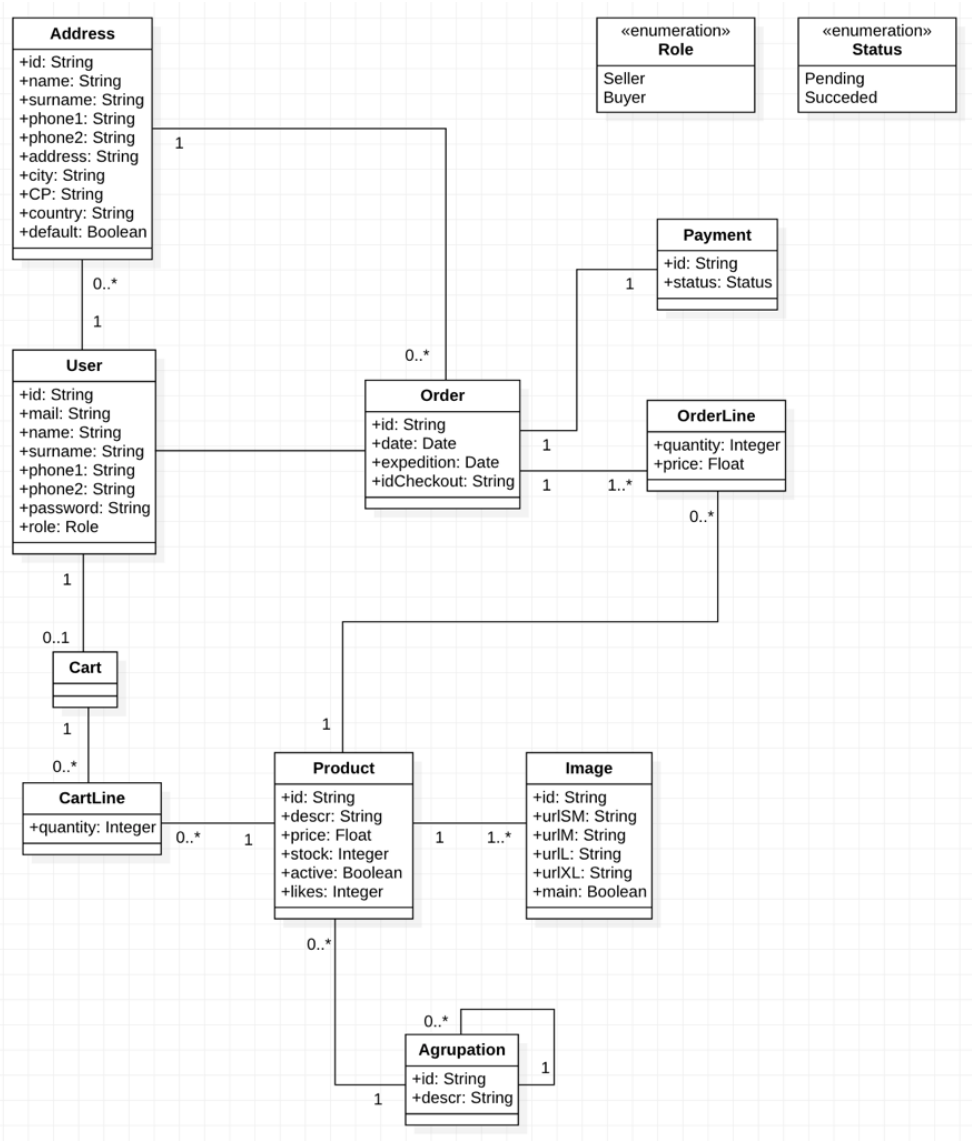


Figura 11: Diagrama UML

En la base de datos *noSQL* los datos se estructuran mediante documentos y colecciones de documentos. Como puede verse en la *Figura 12,* hay datos que se almacenan en distintas ubicaciones como por ejemplo los productos. Para cada una de las agrupaciones por ejemplo, se almacenan todos los productos que

pertenecen a dicha agrupación. Esto hace que las consultas sean directas pero en las actualizaciones deberá actualizarse el producto en todas ubicaciones. Esto mismo ocurre con los pedidos.

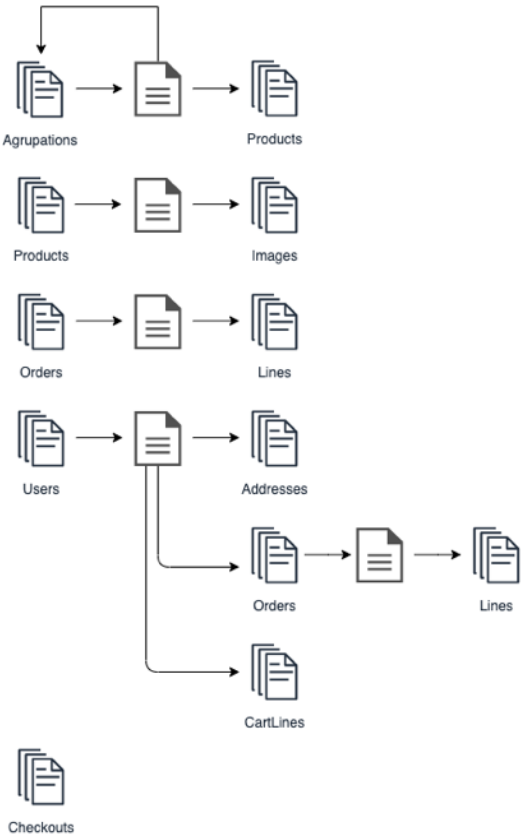


Figura 12: Estructura datos en Firestore

A continuación pasa a detallarse los atributos que se almacenan en cada uno de los tipos de documento.

User:

id: string
admin: string
email: string
name: string
surname: string
phone1: string
phone2: string

Address:

id: string
default: boolean
name: string
surname: string
phone1: string
phone2: string
address: string
city: string
CP: string
country: string

Cart:

id: string
product:: **Product**
price: float
quantity: integer

La autentificación de los usuarios se delega al servicio de Firestore *auth* que entre otros se encarga de gestionar datos con una alta sensibilidad como puede ser la contraseña del usuario.

En el caso de la cesta puede apreciarse que se guarda todos los datos del producto para evitar así tener que realizar subconsultas para cada una de las líneas de la cesta. En este caso los datos no se mantendrán actualizados.

Agrupation:
- Id: string
- path: string []
- pathDescription: string
- description: string
- hasChildren: boolean
- hasItems: boolean

Product:
- active: boolean
- likes: integer
- reference: string
- description: string
- quantity: string
- price: string
- mainImage: **Image**
- agrupation: **Agrupation**

Image:
- id: string
- isMain: string
- urls:
  - imgSM: string
  - imgM: string
  - imgL: string
  - imgXL: string

En las agrupaciones puede apreciarse que se guardan datos que podrían obtenerse mediante consultas, sin embargo se ha optado por realizar el cálculo de estos datos en el servidor en el mismo momento en que se realicen actualizaciones. Por ejemplo, el atributo *hasChildren* indica si la agrupación tiene a su vez subagrupaciones y el atributo *hasItems* indica si la agrupación tiene productos asignados. Estos dos atributos resultaran especialmente útiles en la gestión del componente que se va a utilizar para visualizar las *Agrupaciones*. Otros atributos que se van calculando en el momento de realizar actualizaciones son *path* y *pathDescription* que permiten disponer en todo momento del camino a seguir para llegar a una determinada agrupación así como las descripciones de todas estas agrupaciones.

En el caso de los productos, aún y disponer cada producto de una colección de imágenes, puede observarse que en el mismo producto se guardan nuevamente los datos de la imagen principal. Esto se hace para evitar subconsultas en el *Listado de Productos* y el *Catálogo* para cada producto deba realizarse una subconsulta ralentizándose así la presentación de los datos.

Como se ha comentado con anterioridad, nuestra aplicación web será *responsive*, por lo que en función del ancho de pantalla del dispositivo la información se presentará de una u otra forma. Este diseño *responsive* tiene en cuenta la resolución de imagen que cada dispositivo necesita evitando así descargar imágenes con un peso elevado en los dispositivos más pequeños que además normalmente son los que disponen de más restricciones en cuanto a límite de datos descargados y velocidad de descarga.

Order:

id: string
date: date
expedition: date
checkout: string
status: string
lines: OrderLine[]

OrderLines:

id: string
product:: **Product**
quantity: integer

Checkouts:

id: string
order: string

Finalmente, en los pedidos se sigue el mismo criterio que en la Cesta guardando las características de los productos en la misma línea del pedido.

8.2. Casos de uso

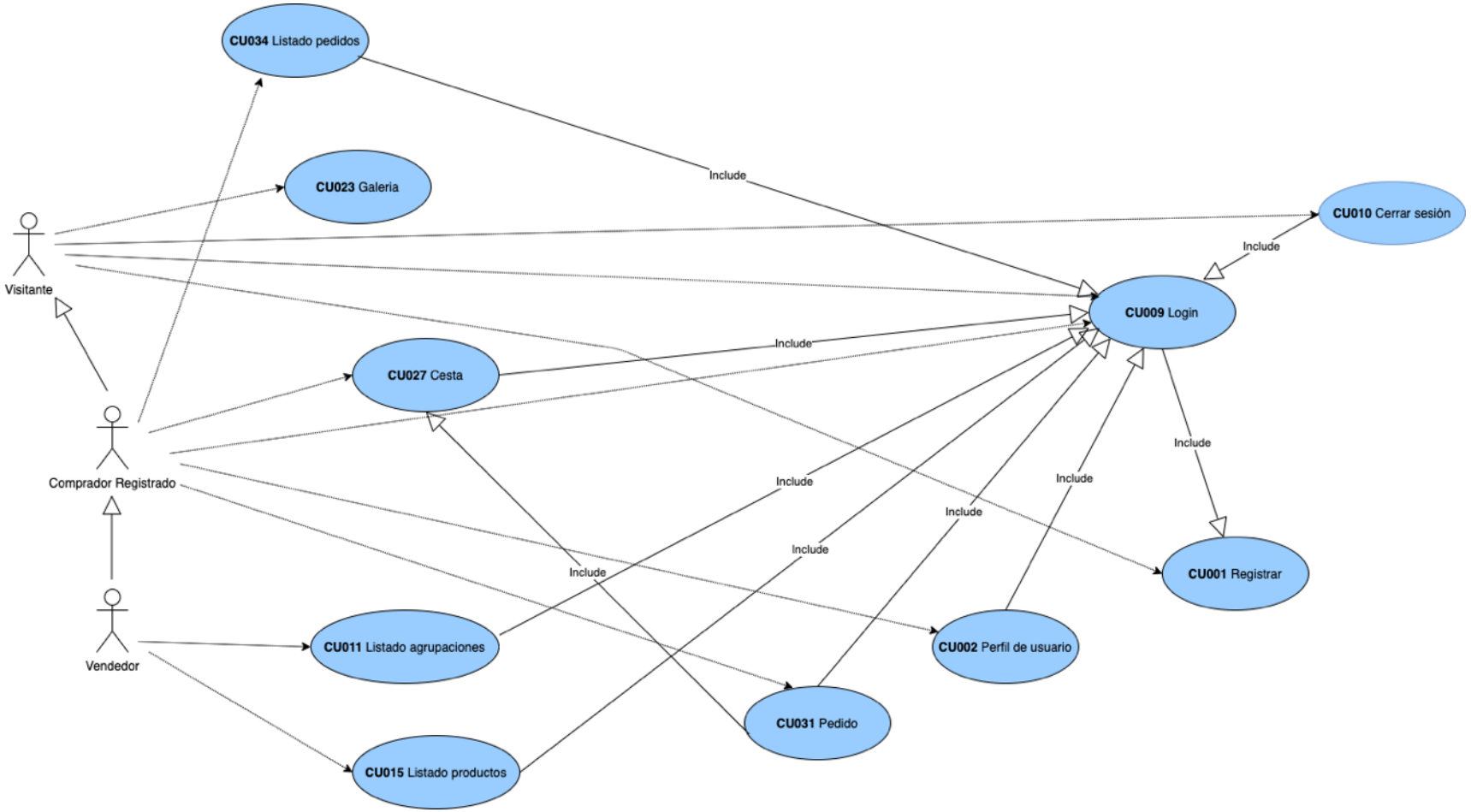


Figura 13: Casos de uso

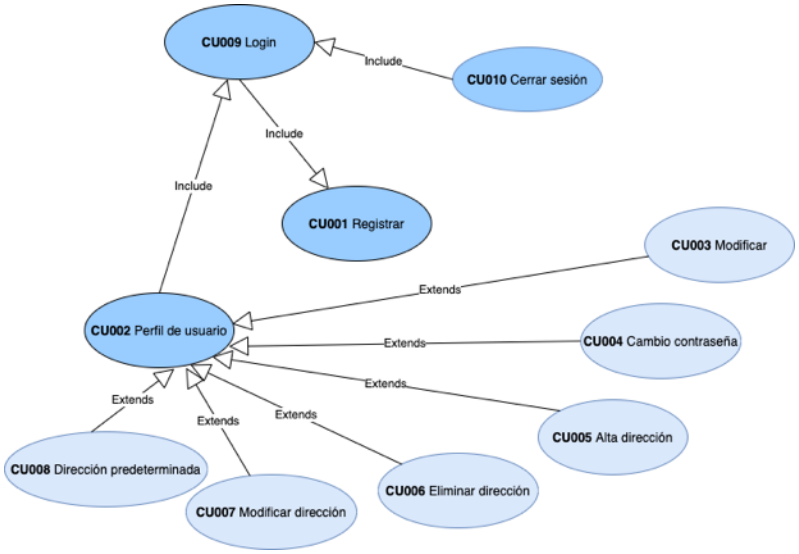


Figura 14: Casos uso relacionados usuario

CU001 Registrar

Resumen: Se registra un nuevo usuario en la aplicación.

Actores: Comprador.

Casos relacionados: CU009 Login.

Precondición: NA

Escenario Principal:

1. El usuario accede a la opción de Registrar disponible en la parte inferior de la pantalla de Login.
2. El sistema muestra un formulario con los datos que debe informar el usuario.
3. El usuario informa los datos y pulsa el botón Guardar.
4. El sistema valida los datos, los guarda, registra al usuario en la aplicación y lo envía a su perfil.

Escenario Alternativo y excepciones:

3a. El usuario pulsa el botón *Cancelar* del formulario de registro.

3a1. El sistema envía al usuario a la página de Inicio sin guardar los datos.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

Postcondición: El usuario queda registrado en el sistema.

CU002 Perfil de Usuario

Resumen: Se muestra los datos del usuario.

Actores: Comprador, Vendedor

Casos relacionados: CU002 Login.

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario pulsa la opción de consulta de su perfil.

2. El sistema muestra los datos del usuario. En la parte superior los datos personales y en la parte inferior las distintas direcciones de envío.

Postcondición: NA

CU003 Perfil de Usuario, Modificar

Resumen: Se modifica alguno de los datos personales del usuario.

Actores: Comprador, Vendedor

Casos relacionados: CU002 Perfil Usuario.

Precondición: El usuario ha hecho el login y se encuentra viendo su perfil.

Escenario Principal:

1. El usuario pulsa el botón asociado a la modificación de los datos personales.
2. El sistema muestra el formulario con los datos personales actuales.
3. El usuario modifica los datos que desea y pulsa el botón Guardar.
4. El sistema valida los datos, los guarda y vuelve a mostrar el perfil del usuario.

Escenario Alternativo y excepciones:

3a. El usuario pulsa el botón *Cancelar* del formulario.

3a1. El sistema envía al usuario nuevamente al Perfil de Usuario sin aplicar los cambios.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

5a. El usuario quiere modificar el email.

5a1. El sistema no deja modificar este dato ya que es una de las claves utilizadas para identificar al usuario en el sistema.

Postcondición: Los nuevos datos se reflejan en el perfil del usuario.

CU004 Perfil Usuario, Cambiar Contraseña

Resumen: Permite modificar la actual contraseña del usuario.

Actores: Comprador, Vendedor

Casos relacionados: CU002 Perfil Usuario.

Precondición: El usuario ha hecho el login y se encuentra viendo su perfil.

Escenario Principal:

1. El usuario pulsa el botón asociado al cambio de contraseña.
2. El sistema solicita la nueva contraseña.
3. El usuario pulsa el botón Aplicar.

El sistema valida los datos, los guarda y envía al usuario nuevamente al Perfil de Usuario.

Escenario Alternativo y excepciones:

3a. El usuario pulsa el botón *Cancelar*.

3a1. El sistema envía al usuario nuevamente al Perfil de Usuario.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

Postcondición: NA

CU005 Perfil Usuario, Alta dirección

Resumen: Permite dar de alta una nueva dirección de envío.

Actores: Comprador

Casos relacionados: CU002 Perfil Usuario.

Precondición: El usuario ha hecho el login y se encuentra viendo su perfil.

Escenario Principal:

1. El usuario pulsa el botón asociado al alta de una dirección de envío.
2. El sistema muestra el formulario asociado al proceso de alta.
3. El usuario informa los datos y pulsa el botón Guardar.
4. El sistema valida los datos, los guarda y envía al usuario nuevamente a su perfil.

Escenario Alternativo y excepciones:

3a. El usuario pulsa el botón *Cancelar*.

3a1. El sistema envía al usuario nuevamente a su perfil.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

4b. Es la primera dirección que define el usuario.

4b1. El sistema marca la dirección como dirección predeterminada.

Postcondición: La nueva dirección aparece en el perfil del usuario.

CU006 Perfil Usuario, Eliminar dirección

Resumen: Permite eliminar una determinada dirección de envío.

Actores: Comprador

Casos relacionados: CU002 Perfil Usuario.

Precondición: El usuario ha hecho el login y se encuentra viendo su perfil.

Escenario Principal:

1. El usuario selecciona la dirección a eliminar y pulsa el botón Eliminar.
2. El sistema elimina la dirección.

Escenario Alternativo y excepciones:

2a. El usuario elimina la dirección predeterminada.

2a1. El sistema no permite eliminar la dirección predeterminada.

Postcondición: La dirección ya no aparece en el perfil del usuario.

CU007 Perfil Usuario, Modificar dirección

Resumen: Permite modificar los datos de una determinada dirección de envío.

Actores: Comprador

Casos relacionados: CU002 Perfil Usuario.

Precondición: El usuario ha hecho el login y se encuentra viendo su perfil.

Escenario Principal:

1. El usuario selecciona la dirección objeto de la modificación y selecciona la opción Modificar.
2. El sistema muestra el formulario asociado a la modificación de la dirección de envío con los datos actuales de la dirección.
3. El usuario informa los nuevos datos y pulsa el botón Guardar.
4. El sistema valida los datos, los guarda y envía al usuario nuevamente a su perfil.

Escenario Alternativo y excepciones:

3a. El usuario pulsa el botón Cancelar.

3a1. El sistema envía al usuario nuevamente a su perfil sin realizar ningún cambio.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

Postcondición: Los nuevos datos de la dirección aparecen en el perfil del usuario.

CU008 Perfil Usuario, Dirección predeterminada

Resumen: Permite modificar la dirección de envío predeterminada.

Actores: Comprador

Casos relacionados: CU002 Perfil Usuario.

Precondición: El usuario ha hecho el login y se encuentra viendo su perfil.

Escenario Principal:

1. El usuario marca como predeterminada una dirección de envío.
2. El sistema guarda como predeterminada la dirección de envío

Escenario Alternativo y excepciones: NA

Postcondición: NA.

CU009 Login

Resumen: El usuario se identifica en la aplicación web.

Actores: Comprador, Vendedor

Casos relacionados: NA

Precondición: NA.

Escenario Principal:

1. El usuario pulsa el botón de login.
2. El sistema muestra una ventana donde se solicita el email y contraseña del usuario.
3. El usuario informa los datos y pulsa el botón Entrar.
4. El sistema valida los datos.

Escenario Alternativo y excepciones:

1.El usuario se ha identificado anteriormente y no ha cerrado la sesión.

1a1. El sistema recupera los datos de conexión y realiza el login.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra un mensaje de error.

Postcondición: El usuario tiene acceso a las funcionalidades propias de su rol.

CU010 Cerrar sesión

Resumen: El usuario cierra la sesión.

Actores: Comprador, Vendedor

Casos relacionados: CU009 Login.

Precondición: El usuario se ha identificado en la aplicación.

Escenario Principal:

1. El usuario pulsa el botón Cerrar Sesión
2. El sistema cierra la sesión del usuario, elimina los datos de conexión y lo envía a la página de inicio.

Escenario Alternativo y excepciones: NA

Postcondición: El usuario deja de tener acceso a las funcionalidades propias de su rol.

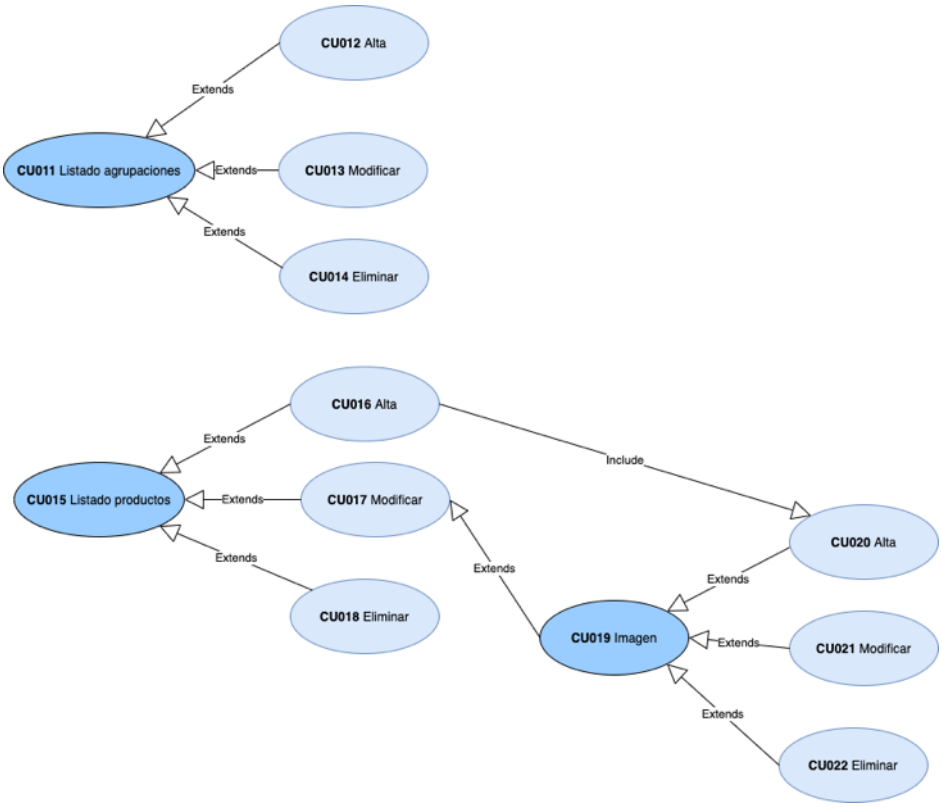


Figura 15: Casos de uso relacionados Gestión de Productos

CU011 Listado agrupaciones

Resumen: Se muestra el listado de agrupaciones.

Actores: Vendedor

Casos relacionados: CU009 Login.

Precondición: El usuario se ha identificado en la aplicación y tiene rol de vendedor.

Escenario Principal:

1. El usuario selecciona la opción Agrupaciones.

2. El sistema muestra las agrupaciones de primer nivel.

Escenario Alternativo y excepciones:

2a. El usuario quiere ver el detalle subagrupaciones de una determinada agrupación

2a1. El usuario selecciona la agrupación a consultar.

2a2. El sistema muestra el detalle de subagrupaciones.

Postcondición: NA

CU012 Listado agrupaciones, Alta

Resumen: Permite dar de alta una determinada agrupación.

Actores: Vendedor

Casos relacionados: CU011 Listado agrupaciones.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario pulsa el botón Nueva Agrupación.
2. El sistema muestra un formulario donde informar la descripción correspondiente a la agrupación.
3. El usuario informa la descripción y pulsa el botón Guardar.
4. El sistema valida los datos y los guarda permanentemente.

Escenario Alternativo y excepciones:

1a. El usuario quiere detallar una subagrupación.

1a1. El usuario selecciona la agrupación en la que se quiere añadir una subagrupación y selecciona la opción añadir de esta agrupación.

1b. El usuario quiere detallar una subagrupación de una agrupación con productos ya definidos.

1b1. El sistema muestra la opción añadir deshabilitada.

1c. El usuario quiere detallar una subagrupación de una agrupación de nivel de profundidad 4.

1c1. El sistema muestra la opción añadir deshabilitada.

4a. La descripción no se ha informado y está en blanco.

4a1. El botón Guardar está deshabilitado.

Postcondición: El listado de agrupaciones muestra el nuevo registro.

CU013 Listado agrupaciones, Edición

Resumen: Permite modificar la descripción de una determinada agrupación.

Actores: Vendedor

Casos relacionados: CU011 Listado agrupaciones.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario selecciona la agrupación objeto de la modificación y selecciona la opción Editar.
2. El sistema muestra un formulario con la descripción actual.
3. El usuario informa la nueva descripción y pulsa el botón Guardar.
4. El sistema valida los datos y los guarda permanentemente.

Escenario Alternativo y excepciones:

3a. La descripción se ha dejado en blanco.

3a1. El botón Guardar está deshabilitado.

4a. El usuario pulsa el botón Cancelar.

4a1. El sistema cancela la edición de la subagrupación.

Postcondición: El listado de agrupaciones muestra la nueva descripción de la agrupación.

CU014 Listado agrupaciones, Eliminar

Resumen: Permite eliminar una determinada agrupación.

Actores: Vendedor

Casos relacionados: CU011 Listado agrupaciones.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario selecciona la agrupación a eliminar y pulsa sobre la opción Eliminar.
2. El sistema valida los datos y elimina la agrupación permanentemente.

Escenario Alternativo y excepciones:

1a. El usuario selecciona una agrupación que tiene subagrupaciones o productos definidos.

1a1. El usuario se encuentra con la opción Eliminar deshabilitada.

Postcondición: La agrupación eliminada ya no aparece en el listado de agrupaciones.

CU015 Listado productos

Resumen: Permite acceder al listado desde donde puede gestionarse los productos.

Actores: Vendedor

Casos relacionados: CU009 Login.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El sistema muestra una pantalla en blanco informando que debe seleccionarse una determinada agrupación.
2. El usuario selecciona una determinada agrupación.
3. El sistema muestra los productos incluidos en dicha agrupación y sus sucesivas subagrupaciones. En color rojo aparecen los artículos sin stock y en color gris los productos no activos.

Postcondición: NA

CU016 Listado productos, Alta

Resumen: Permite dar de alta un nuevo producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario pulsa el botón de Alta.
2. El sistema muestra un formulario donde el usuario debe informar los datos del producto.
3. El usuario informa los datos y pulsa el botón Guardar.
4. El sistema valida los datos, los guarda permanentemente y envía al usuario al listado de productos.

Escenario Alternativo y excepciones:

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

Postcondición: El producto queda registrado de forma permanente (datos e imágenes).

CU017 Listado productos, Modificar

Resumen: Permite modificar ciertos datos del producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: El usuario con rol de comprador se ha identificado.

Escenario Principal:

1. El usuario selecciona el producto a modificar y selecciona la opción Editar.
2. El sistema muestra un formulario donde se muestra los datos del producto.
3. El usuario modifica los datos y pulsa el botón Guardar.
4. El sistema valida los datos, los guarda permanentemente y envía al usuario al listado de productos.

Escenario Alternativo y excepciones:

3a. El usuario quiere modificar la referencia del producto.

3a1. El sistema no permite modificar la referencia del producto.

3b. El usuario quiere modificar la agrupación del producto.

3b1. El sistema no permite modificar la agrupación del producto.

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

Postcondición: Las modificaciones del producto quedan registradas de forma permanente (datos e imágenes).

CU018 Listado productos, Eliminar

Resumen: Permite eliminar un determinado producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario selecciona el producto a eliminar y pulsa la opción Eliminar.
2. El sistema elimina permanentemente el producto.

Escenario Alternativo y excepciones: NA

Postcondición: El producto eliminado ya no aparece en el listado de productos.

CU019 Imágenes

Resumen: En el detalle del producto se muestra una miniatura de las imágenes asociadas a dicho producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: NA

Escenario Principal: NA

Escenario Alternativo y excepciones: NA

Postcondición: NA

CU020 Imágenes, Alta

Resumen: Permite añadir una nueva imagen en el detalle del producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario selecciona la opción añadir imagen.
2. El sistema muestra el navegador de ficheros.
3. El usuario selecciona la imagen a añadir.
4. El sistema muestra en el detalle del producto la imagen en miniatura.

Escenario Alternativo y excepciones:

3a. El usuario cancela el proceso.

3a1. El sistema vuelve a mostrar el detalle del producto sin insertar ninguna imagen nueva.

4a. Se da de alta la primera imagen del producto.

4a1. El sistema marca esta imagen como principal. En los listados será la imagen que se mostrará inicialmente.

Postcondición: NA

CU021 Imágenes, Modificación

Resumen: Permite modificar la imagen seleccionada en el detalle del producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario selecciona alguna de las imágenes en miniatura del detalle del producto.
2. El sistema muestra el navegador de ficheros.
3. El usuario selecciona la nueva imagen a añadir.
4. El sistema valida los datos y muestra en el detalle del producto la miniatura la nueva imagen.

Escenario Alternativo y excepciones:

3a. El usuario cancela el proceso.

3a1. El sistema vuelve a mostrar el detalle del producto sin modificar la imagen seleccionada.

Postcondición: NA

CU022 Imágenes, Eliminar

Resumen: Permite eliminar una determinada imagen del detalle del producto.

Actores: Vendedor

Casos relacionados: CU015 Listado productos.

Precondición: El usuario con rol de vendedor se ha identificado.

Escenario Principal:

1. El usuario selecciona la opción eliminar de alguna de las imágenes del detalle del producto.
2. El sistema elimina del detalle del producto la imagen seleccionada.

Escenario Alternativo y excepciones:

3a. El usuario quiere eliminar la imagen principal.

3a1. En la imagen principal el sistema no muestra la opción eliminar.

Postcondición: NA

Figura 16: Casos de uso relacionados Galería y Compra

CU023 Galería

Resumen: Permite acceder a la galería de productos.

Actores: Vendedor, Comprador.

Casos relacionados: NA

Precondición: NA

Escenario Principal:

1. El usuario selecciona la opción catálogo.
2. El sistema muestra una pequeña imagen y descripción de los productos activos y con stock.

Escenario alternativo y excepciones

3a. El usuario quiere ver los productos de una determinada agrupación.

3a1. El usuario selecciona una determinada agrupación.

3a2. El sistema muestra todos los productos que pertenecen a la agrupación seleccionada así como sus sucesivas subagrupaciones.

Postcondición: NA

CU024 Galería, Ver Imágenes

Resumen: Permite acceder a la galería de imágenes del producto seleccionado.

Actores: Vendedor, Comprador.

Casos relacionados: C023 Galería.

Precondición: NA

Escenario Principal:

1. El usuario pulsa sobre la imagen de un determinado producto.
2. El sistema muestra la imagen seleccionada ampliada así como una descripción detallada del producto.

Escenario alternativo y excepciones

2a. El usuario quiere ver las distintas imágenes del producto.

2a1. El usuario puede navegar a izquierda y derecha para ver las distintas imágenes del producto.

2b. El usuario quiere volver a la galería.

2b1. El usuario cierra el visualizador de imágenes.

2b2. El sistema muestra nuevamente la galería.

Postcondición: NA

CU025 Galería, Me Gusta

Resumen: Permite ver la aceptación que tienen los productos así como destacar aquellos artículos que más gustan a los usuarios.

Actores: Vendedor, Comprador.

Casos relacionados: CU023 Galería

Precondición: NA

Escenario Principal:

1. El usuario selecciona la opción Me Gusta.
2. El sistema incrementa los “Me Gusta” del producto seleccionado.

Escenario alternativo y excepciones: NA

Postcondición: En la galería se puede ver el incremento de los “Me Gusta”.

CU026 Galería, Añadir Cesta

Resumen: Permite seleccionar aquellos productos que el usuario quiere comprar.

Actores: Vendedor, Comprador.

Casos relacionados: CU023 Galería

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario selecciona la opción Añadir al Carrito.
2. El sistema añade el producto seleccionado a la cesta de productos a comprar.

Escenario alternativo y excepciones:

2a. El usuario no se ha identificado en la web.

2a1. El sistema muestra un mensaje de aviso indicando que el usuario debe identificarse en la web.

Postcondición: El producto queda guardado permanentemente dentro de la Cesta.

CU027 Cesta

Resumen: En la cesta el usuario tiene registrados aquellos productos que desea comprar.

Actores: Vendedor, Comprador.

Casos relacionados: CU009 Login

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario selecciona la opción de Consulta de la Cesta.
2. El sistema muestra la relación de productos añadidos a la cesta, mostrando su cantidad y precio. También se muestra el valor total de la compra.

Escenario alternativo y excepciones:

2a. El usuario cierra la sesión y tiene productos en la cesta.

2a1. El sistema guarda permanentemente el contenido de la cesta.

Postcondición: NA.

CU028 Cesta, Eliminar

Resumen: El usuario elimina uno de los productos añadidos en la cesta.

Actores: Vendedor, Comprador.

Casos relacionados: CU027 Cesta

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El selecciona la opción eliminar del producto objetivo.

2. El sistema elimina permanentemente el producto de la cesta.

Escenario alternativo y excepciones: NA

Postcondición: NA

CU029 Cesta, Modificar

Resumen: El usuario modifica la cantidad de un determinado producto.

Actores: Vendedor, Comprador.

Casos relacionados: CU027 Cesta

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario incrementa la cantidad a comprar y pulsa el botón actualizar.
2. El sistema registra permanentemente los cambios y recalcula el importe total de la cesta.

Escenario alternativo y excepciones:

2a. El usuario no pulsa el botón actualizar.

2a1. El sistema no registra los cambios y no recalcula el importe total de la cesta.

Postcondición: NA

CU030 Cesta, Tramitar Pedido

Resumen: El usuario empieza a tramitar el pedido de compra.

Actores: Vendedor, Comprador.

Casos relacionados: CU027 Cesta

Precondición: El usuario se ha identificado en la web y tiene la cesta de la compra con algún producto.

Escenario Principal:

1. El usuario empieza a tramitar el pedido de compra.
2. El sistema valida los datos de la cesta, recalcula los stocks de los productos, precios y totales de la compra, y crea el pedido con el estado *Pendiente de Pago*.

Escenario alternativo y excepciones:

2a. No hay suficiente cantidad de alguno de los productos.

2a1. El sistema muestra un mensaje de aviso para que el usuario realice los cambios oportunos.

2a. El stock del producto pasa a ser 0.

2a1. El sistema deja de mostrar el producto en la galería.

Postcondición: NA

CU031 Pedido

Resumen: El usuario ha empezado a tramitar el pedido de compra.

Actores: Vendedor, Comprador.

Casos relacionados: CU027 Cesta

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario empieza a tramitar el pedido de compra.
2. El sistema muestra la dirección de envío que se corresponde con la que tiene marcada como predeterminada en su perfil, así como el importe total del pedido y su contenido.

Escenario alternativo y excepciones: NA

Postcondición: El pedido queda registrado en el sistema.

CU032 Pedido, Cambiar Dirección de Envío

Resumen: El usuario cambia la dirección de envío del pedido de compra.

Actores: Vendedor, Comprador.

Casos relacionados: CU031 Pedido

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario quiere cambiar la dirección de envío del pedido y pulsa la opción de Modificar.
2. El sistema muestra las direcciones de envío que el usuario tiene definidas.
3. El usuario selecciona una de las direcciones.
4. El sistema establece como dirección de envío la nueva dirección.

Escenario alternativo y excepciones: NA

Postcondición: NA

CU033 Pedido, Pagar

Resumen: El usuario efectúa el pago del Pedido.

Actores: Vendedor, Comprador

Casos relacionados: CU031 Pedido

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El sistema crea el pedido y envía un email con su contenido al usuario.
2. El usuario inicia el pago del pedido.
3. El sistema muestra un formulario solicitando los datos de la tarjeta de crédito.
4. El usuario introduce los datos y pulsa el botón aceptar.
5. El sistema valida los datos.

Escenario alternativo y excepciones:

4a. Alguno de los datos introducidos no es correcto.

4a1. El sistema muestra mensajes de error de los campos incorrectos.

4b. El pago se ha efectuado correctamente.

4b1. El sistema marca el pedido como *Pagado*.

Postcondición: NA

Postcondición: NA

CU034a Listado de Pedidos

Resumen: El vendedor consulta los pedidos que le han efectuado.

Actores: Vendedor.

Casos relacionados: CU009 Login

Precondición: El usuario se ha identificado en la web y tiene el rol de vendedor.

Escenario Principal:

1. El vendedor consulta el Listado de Pedidos.
2. El sistema muestra un listado de todos los Pedidos registrados en el sistema.

Postcondición: NA

CU034b Listado de Pedidos

Resumen: El comprador consulta los pedidos que ha realizado.

Actores: Comprador.

Casos relacionados: CU009 Login

Precondición: El usuario se ha identificado en la web y tiene el rol de comprador.

Escenario Principal:

1. El comprador consulta el Listado de Pedidos.
2. El sistema muestra un listado de todos los Pedidos que ha realizado.

Escenario alternativo y excepciones: NA

Postcondición: NA

CU035 Listado de Pedidos, Detalle

Resumen: El usuario consulta el detalle de un determinado pedido.

Actores: Comprador i Vendedor.

Casos relacionados: CU033a Listado Pedidos.

Precondición: El usuario se ha identificado en la web.

Escenario Principal:

1. El usuario selecciona la opción *ver detalle.*
2. El sistema muestra el detalle del pedido.

Escenario alternativo y excepciones: NA

Postcondición: NA

CU036 Listado de Pedidos, Expedir

Resumen: El usuario marca un determinado pedido como expedido.

Actores: Vendedor.

Casos relacionados: CU033a Listado Pedidos.

Precondición: El usuario se ha identificado en la web y tiene el rol de vendedor.

Escenario Principal:

1. El usuario selecciona la opción Expedir.

2. El sistema marca el pedido como expedido asignándole como fecha de expedición la fecha actual.

Escenario alternativo y excepciones:

1a. El usuario quiere retroceder la expedición.

1a1. El usuario selecciona el pedido expedido y selecciona nuevamente la opción Expedir.

1a2. El sistema retrocede la expedición.

Postcondición: NA

CU037 Listado de Pedidos, Eliminar

Resumen: El usuario elimina un determinado Pedido.

Actores: Vendedor.

Casos relacionados: CU033a Listado Pedidos.

Precondición: El usuario se ha identificado en la web y tiene el rol de vendedor.

Escenario Principal:

1. El usuario selecciona la opción Eliminar.
2. El sistema elimina del sistema el pedido eliminado y rehace el stock de los productos eliminados.

Postcondición: NA

Planificación desarrollo

Una vez definidos los casos de uso que detallan las distintas funcionalidades de nuestra aplicación web se ha realizado la planificación de su desarrollo en función de dichos casos. Para realizar la planificación se ha agrupado los distintos casos de uso según su funcionalidad.

Tarea	Fecha inicio	Fecha fin	Carga (h)
CU001 Registrar	**29/10/20**	**31/10/20**	**9**
CU002 Perfil de Usuario	**01/11/20**	**04/11/20**	**4**
CU003 Modificar			
CU004 Cambio Contraseña			
CU005 Alta Dirección			
CU006 Eliminar Dirección			
CU007 Modificar Dirección			
CU008 Establecer Dirección Predeterminada			
CU009 Login	**05/11/20**	**06/11/20**	**2**
CU010 Cerrar Sesión			
CU011 Listado Agrupaciones	**07/11/20**	**11/11/20**	**5**
CU012 Alta			
CU013 Modificar			
CU014 Eliminar			
CU015 Listado Productos	**12/11/20**	**17/11/20**	**6**
CU016 Alta			
CU017 Modificar			
CU018 Eliminar			
CU019 Imagen			
CU020 Alta			
CU021 Modificar			
CU022 Eliminar			
CU023 Galería	**18/11/20**	**22/11/20**	**5**
CU024 Ver Imágenes			
CU025 Me gusta			
CU026 Añadir Cesta			
CU027 Cesta	**23/11/20**	**24/11/20**	**2**
CU028 Eliminar			
CU029 Modificar			
CU030 Tramitar Pedido			
CU031 Pedido	**25/11/20**	**28/11/20**	**4**
CU032 Cambiar Dirección Envío			
CU033 Pagar			
CU034 Listado Pedidos	**29/11/20**	**03/11/20**	**5**
CU035 Detalle			
CU036 Expedir			
		TOTAL HORAS:	**42**

Figura 17: Detalle Planificación desarrollo

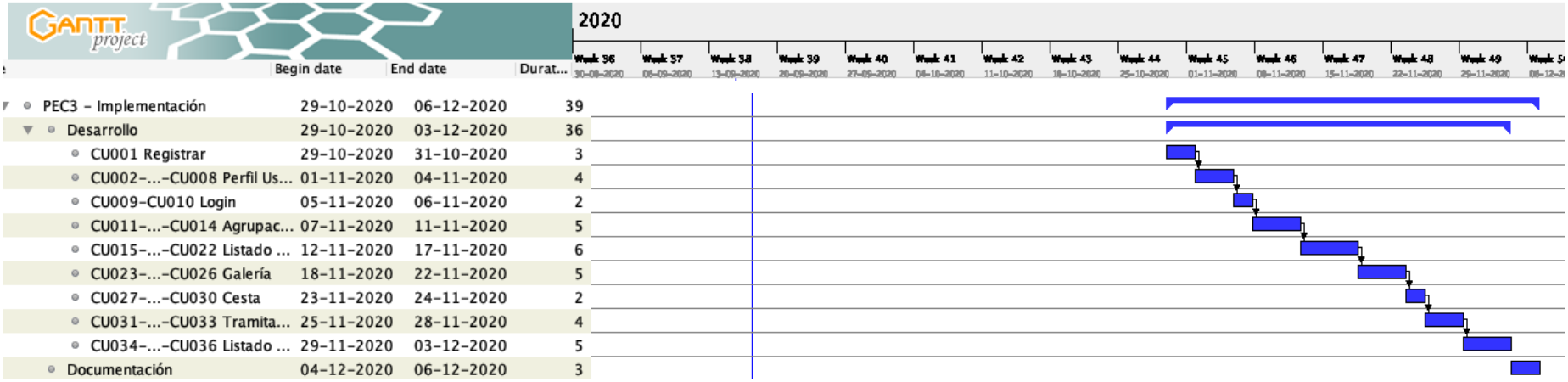


Figura 18: Gantt Planificación Desarrollo

9. Prototipado

A continuación se detalla el prototipado de los distintos *frames* que conforman nuestra aplicación web. Uno de los objetivos de nuestra aplicación es que sea *responsive* y se adapte a los distintos dispositivos existentes en el mercado. Es por ello que para realizar el prototipado se han tenido en cuenta los distintos anchos de pantalla:

- **xs:** screen and (max-width: 599px)
- **sm:** screen and (min-width: 600px) and (max-width: 959px)
- **md:** screen and (min-width: 960px) and (max-width: 1279px)
- **lg:** screen and (min-width: 1280px)

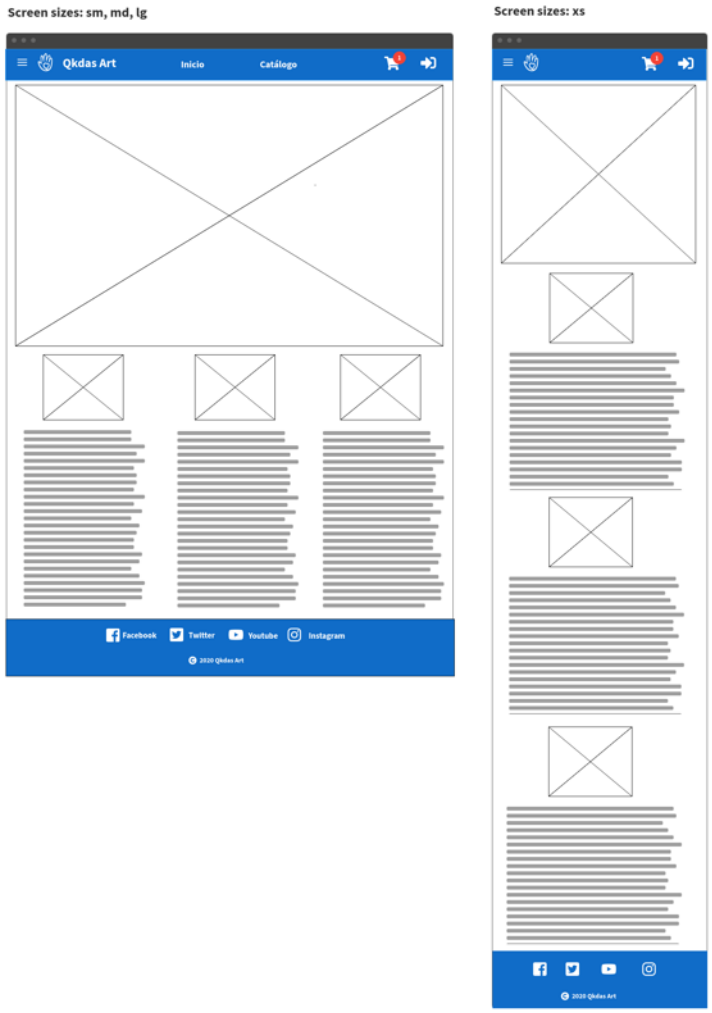


Figura 19: Pantalla Inicio

Figura 20: Pantalla Login (CU009)

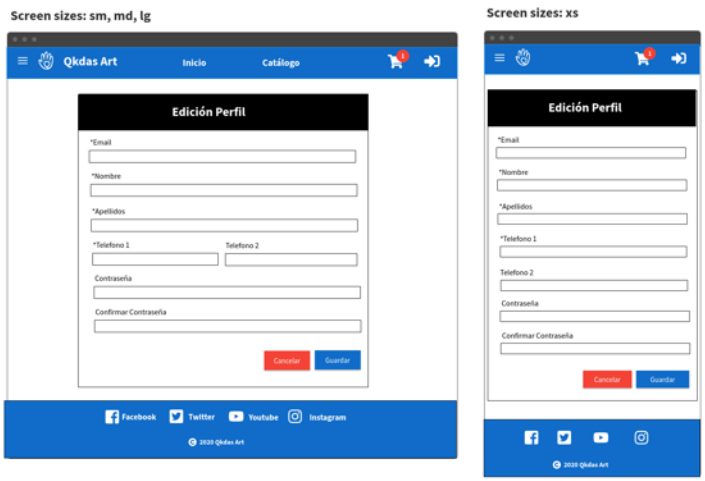


Figura 21: Pantalla Registro (CU001)

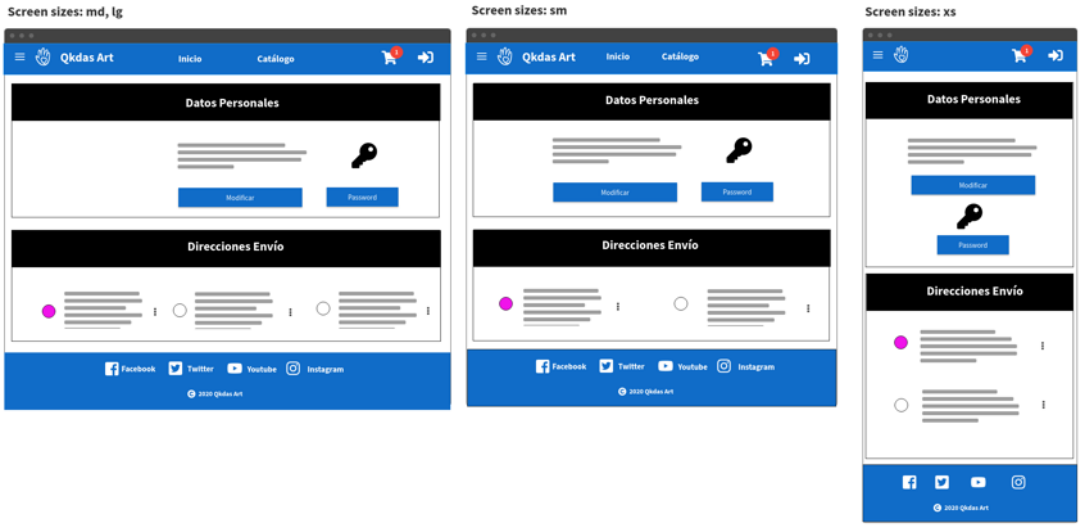


Figura 22: Pantalla Datos Personales (CU003, CU004, CU006, CU007, CU008)

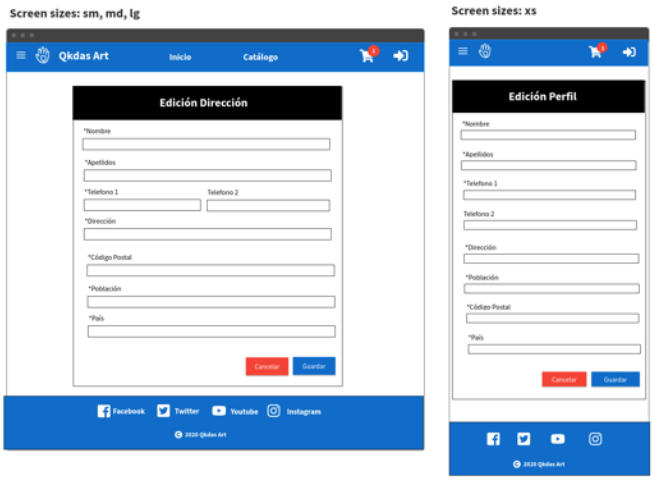


Figura 23: Pantalla Alta Direcciones (CU005)

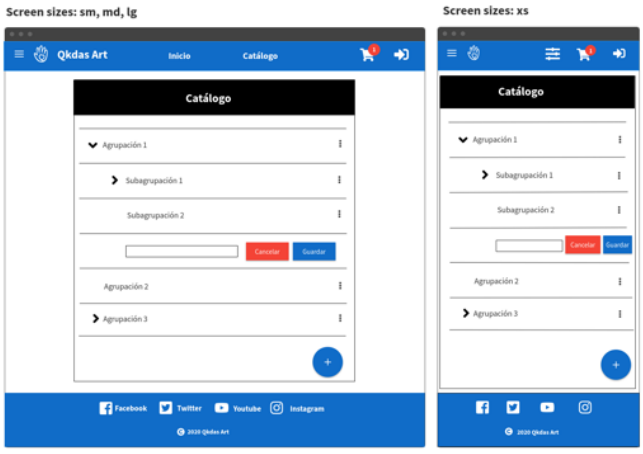


Figura 24: Pantalla Agrupaciones (CU011, CU012, CU013, CU014)

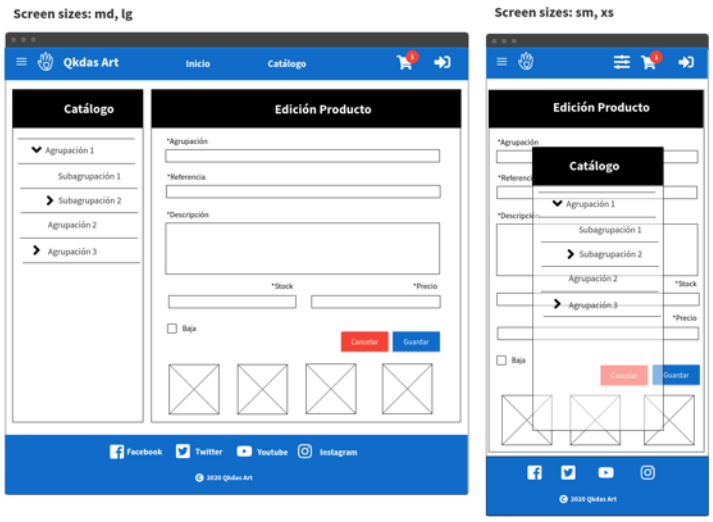


Figura 25: Pantalla Alta Productos (CU016, CU017, CU018, CU019, CU020, CU021, CU022)

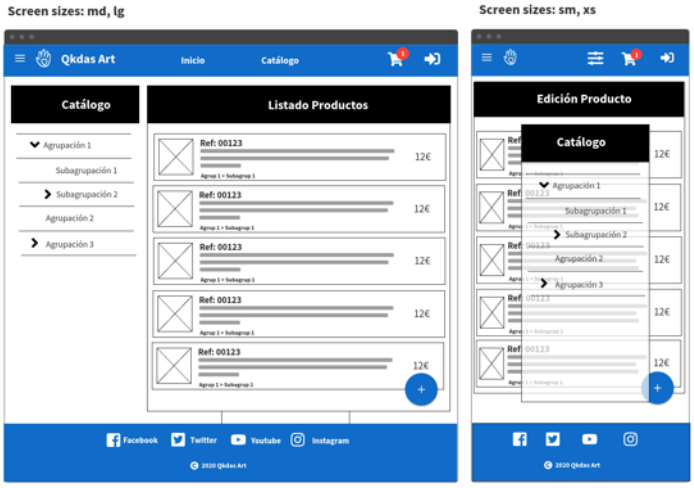
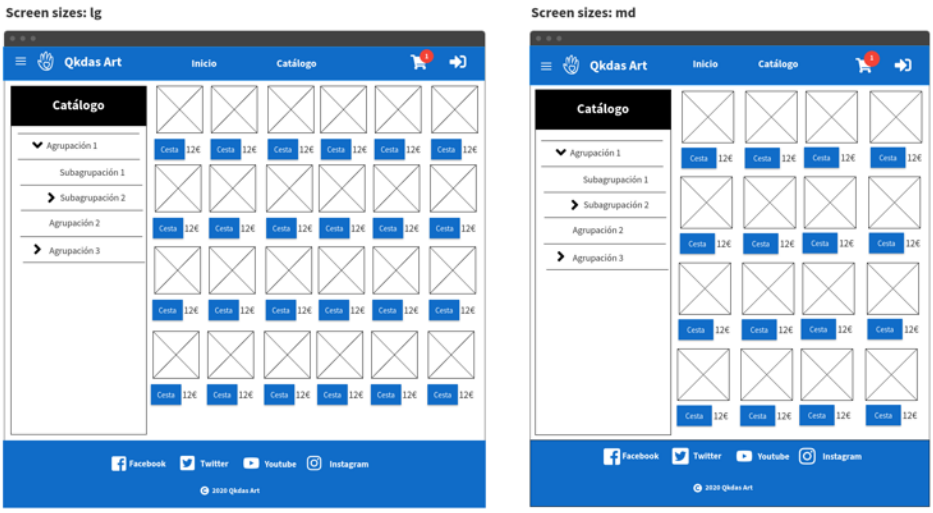


Figura 26: Pantalla Listado Productos (CU015)



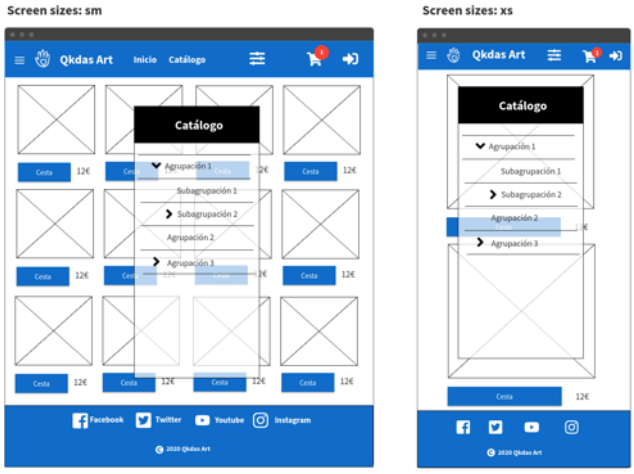


Figura 27: Pantalla Catálogo (CU023, CU024, CU025, CU026)

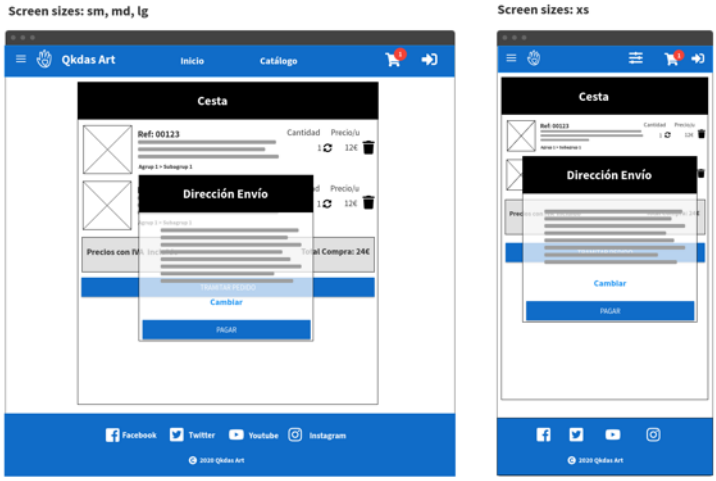


Figura 28: Pantalla Cesta (CU027, CU028, CU029, C030, CU031, CU032, CU033)

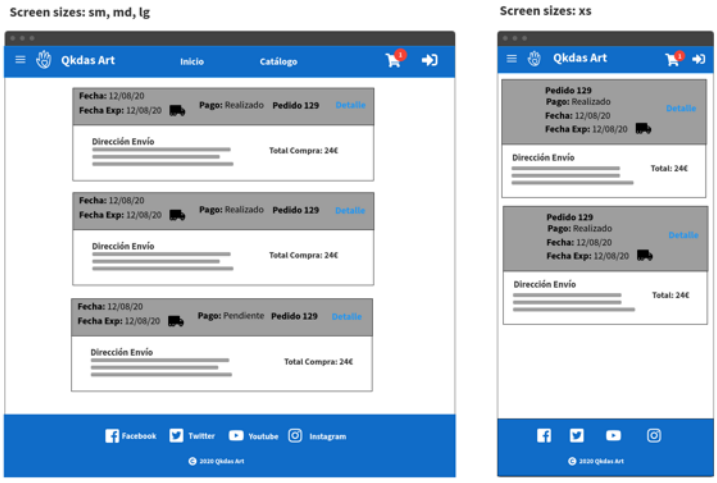


Figura 29: Pantalla Listado Pedidos (CU034, CU036)

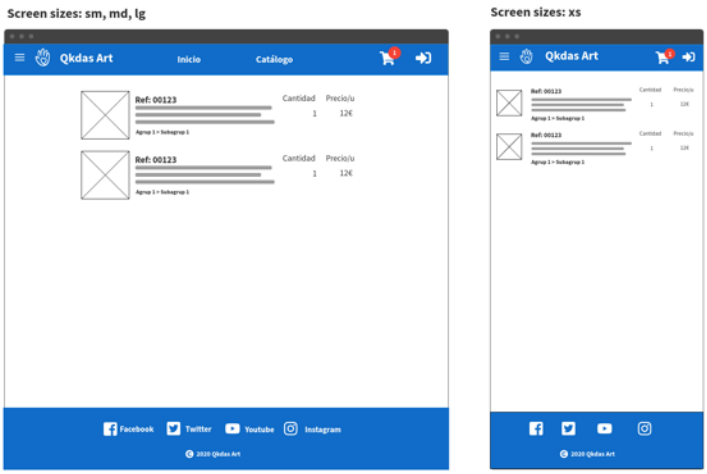


Figura 30: Pantalla Detalle Pedido (CU035)

10.Usabilidad

En nuestro proyecto, mientras se realizaba el diseño del prototipo, se ha ido comprobando que se cumplía con los principios de usabilidad definidos por Jakob Nielsen (Arenzana, 30). Estos principios son:

Visibilidad del estado del sistema

Siempre debe mantenerse informado al usuario del estado del sistema. En nuestra aplicación en cada pantalla se dispone de un título que orienta al usuario sobre la tarea que está realizando. Igualmente, en los dispositivos con un ancho de pantalla menor, en la pantalla correspondiente al catálogo de productos, se mostrará en formato de texto en la parte superior de la pantalla la agrupación a la que hacen referencia los productos visualizados.

En los dispositivos donde no pueda mostrarse el filtro de selección de agrupaciones, en el caso que el usuario tenga activa alguna selección, el botón correspondiente al filtro se resaltará con un color distinto.

En los procesos de carga de datos se mostrará un indicador que permita al usuario cerciorarse que se está ejecutando un proceso.

Relación entre el sistema y el mundo real

El sistema debe hablar el lenguaje del usuario con palabras o frases que a éste le sean familiares y que pueda reconocer con facilidad.

En nuestra aplicación se ha hecho uso de una iconografía que el usuario ya tiene interiorizada en su uso.

Control y libertad del usuario

Está dentro de la naturaleza humana equivocarse. Es por ello que debe darse al usuario la posibilidad de subsanar el error.

En nuestra aplicación siempre se da la posibilidad de retroceder las acciones realizadas mediante la modificación o eliminación de los datos. Al entrar en cada una de las pantallas, el usuario siempre tiene la posibilidad de retroceder sin realizar ninguna operación. Por ejemplo en la cesta de la compra el usuario tiene la posibilidad de eliminar un producto o bien modificar su cantidad.

Consistencia y estándares

En el diseño de toda la aplicación se ha seguido un mismo criterio manteniendo siempre una misma cabecera y pie. Igualmente, todas las pantallas muestran un mismo estilo de títulos y botones.

Prevenir errores

Tiene que prevenirse en la medida de lo posible que el usuario pueda cometer errores. Y en el caso que los cometa tiene que ponerse a su alcance todas las opciones posibles para poder corregirlo.

Es por ello que en nuestra aplicación, aunque en el prototipo inicial no pueda apreciarse, va a validarse la información adjuntada en los distintos campos verificando que la información sea correcta. En el caso que no lo sea se mostrará un mensaje indicando el motivo.

En procesos críticos como por ejemplo el cambio de contraseña, se valida que el usuario realmente no haya cometido ningún error en su introducción haciendo que repita el dato y verificando que en ambos casos se introduzca el mismo valor.

Reconocer antes que recordar

Hacer visibles acciones y opciones para qué el usuario no tenga que recordar información. En los casos en que es posible por espacio se muestra el selector de agrupaciones en la pantalla sin que el usuario deba pulsar ningún botón u opción.

Flexibilidad y eficiencia de uso

Hacer una aplicación preparada para todo tipo de dispositivos y usuarios. Nuestro diseño, tal como se puede apreciar en el prototipo, es *responsive* y se adapta a los distintos tipos de dispositivo existentes en el mercado. Hay que tener en cuenta que este diseño *responsive* no sólo implica una reorganización de la información en función del espacio disponible sino que también tiene en cuenta otros aspectos como por ejemplo el tráfico generado. Así pues, en nuestro desarrollo se tendrá en cuenta que en los dispositivos con menor espacio disponible no requieren la misma resolución que los dispositivos con mayor espacio disponible.

En la galería de imágenes el usuario puede consultar las distintas imágenes que tiene asociadas el producto. En el caso de los dispositivos con pantallas no táctiles la navegación se suele realizar con ayuda del ratón por lo que para realizar la navegación entre las distintas imágenes se utiliza dos botones a izquierda y derecha de la imagen. Por el contrario, los usuarios que utilizan dispositivos con pantallas táctiles tienen asociado el gesto de deslizar el dedo hacia la izquierda y/o derecha para navegar entre las distintas fotos. Es por ello que se hará uso de una librería que nos permita reconocer dichos gestos y tratarlos para que se pueda realizar la navegación.

Diseño estético y minimalista

En el diseño se ha intentado no sobrecargar la aplicación mostrando la información indispensable. Se ha utilizado un diseño *First Mobile* en el que el diseño se empieza pensando en los dispositivos más pequeños. Hay que tener en cuenta que resulta más fácil acomodar posteriormente los datos a dispositivos con más ancho que no a la inversa.

Ayuda a los usuarios a reconocer, diagnosticar y corregir los errores

Hay que intentar que todos los errores que puedan ocurrir en la aplicación estén en un lenguaje entendibles por todos.

En ciertos procesos se muestra mensajes en la parte inferior de la pantalla para informar al usuario. Para que el usuario sepa lo que está sucediendo se interceptan los mensajes que en este caso Firebase pueda dar y se convierten a un lenguaje entendible para el usuario.

Ayuda y documentación

La ayuda debe ser fácil de localizar, especificar los pasos necesarios y no ser muy extensa. En nuestra aplicación podría crearse un pequeño tour de presentación mostrando las principales funcionalidades.

11.Análisis mercado

En el punto *1.1 Contexto y justificación del trabajo* se ha expuesto la problemática de nuestro cliente y las soluciones que actualmente hay en el mercado. De las posibles vías se ha justificado la creación de una web propia.

Si lo que se quiere hacer es crear una tienda *on-line* de cero según las necesidades especificas del cliente, éste puede acudir a multitud de estudios de desarrollo web en los que le podrán confeccionar un presupuesto. Sin embargo, hay que tener en cuenta que el desarrollo de un sitio web totalmente personalizado puede tener un importe demasiado elevado para alguien que empieza en el mundo de la venta *on-line.*

La alternativa existente en el mercado que quizás se ajusta más a las necesidades de nuestro cliente es el modelo de comercialización en que el usuario paga por uso y mes también conocido con el nombre de SaaS[3]. En el mercado hay gran cantidad de oferta en cuanto a la confección de webs que permiten la venta *on-line.* Como ejemplos de este tipo de productos se tiene:

- Ecwid
- WooCommerce
- Shopify
- Prestashop
- Mozello

Cada uno de estos productos sigue distintas estrategias de desarrollo. Algunos de ellos son simples extensiones de *Wordpress* que es un CMS[4] y otros son desarrollos propios. Sin embargo todos coinciden en su estrategia de comercialización que consiste en determinar un precio por uso y mes en función de las funcionalidades disponibles. De esta forma los productos se ajustan a las necesidades de las distintas tipologías de cliente.

- Usuarios que empiezan en en mundo de la venta *on-line* con pocas transacciones
- Usuarios que tienen cierto movimiento

[3] **SaaS**: Software como un Servicio, abreviado ScuS1 (del inglés: Software as a Service, SaaS), es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC), a los que se accede vía Internet desde un cliente. La empresa proveedora TIC se ocupa del servicio de mantenimiento, de la operación diaria y del soporte del software usado por el cliente. Regularmente el software puede ser consultado en cualquier computador, se encuentre presente en la empresa o no.

[4] **CMS:** Un sistema de gestión de contenidos o CMS (del inglés content management system) es un programa informático que permite crear un entorno de trabajo para la creación y administración de contenidos, principalmente en páginas web, por parte de los administradores, editores, participantes y demás usuarios. Cuenta con una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio web. El sistema permite manejar de manera independiente el contenido y el diseño.

- Usuarios que requieren funcionalidades avanzadas por ejemplo de promoción, gestión expediciones…

Así pues este tipo de producto son competencia directa y habrá que tenerlos presentes en la definición de nuestra estrategia de comercialización.

12.Viabilidad

12.1.Coste proyecto

Para ver la viabilidad del proyecto lo primero que debe tenerse en cuenta es el total de horas destinadas a su ejecución. En el caso de este proyecto hay planificadas 333 horas.

Para hacer la valoración económica de estas horas se ha consultado varias fuentes. En el Boletín Oficial del Estado (Colectivo, 2018) se define en el convenio colectivo del sector de la Informática un total de 1.800h al año. Por otro lado el salario bruto medio de un programador junior en el año 2020 es de 19.026 € (Indeed, 2020). Por lo tanto el coste/h es de 10,57 €. Teniendo en cuenta que el total de horas planificadas es de 333h el coste total humano del proyecto es de 3.519,81€.

Además del coste humano hay que tener en cuenta otros tipos de costes. En este proyecto se hace uso de los servicios ofrecidos por la plataforma Firebase. En esta plataforma hay dos tipos de plan (Firebase, 2020):

- **Spark**: sin cargo pero con una serie de limitaciones.
- **Blaze**: incluye el plan *Spark* y en caso de superar dichos límites se cobra por uso.

Para poder operar con el servicio *Cloud Functions* y poder conectar con API's externas a la plataforma Firebase es necesario contratar el plan *Blaze*, así que este es el plan que se debe contratar. Así pues, se deberá tener en cuenta los costes que se puedan generar en Firebase en el total del proyecto.

Otro coste ha tener en cuenta es el alojamiento de nuestra aplicación web. Nuevamente se puede hacer uso del servicio de *Hosting* ofrecido por Firebase. En función del peso de nuestra aplicación este servicio es gratuito. En el caso de nuestra aplicación web no se superan los límites y por lo tanto no supone ningún sobrecoste.

En cuanto al coste del *software* utilizado, éste es gratuito por lo que no supone un incremento en el coste del proyecto. Por otro lado, al externalizar todos los servicios de hosting y no requerir ningún servidor no se tiene ningún coste de *hardware*.

Así de forma resumida se tiene:

Coste recursos humanos:	3.519,81 €
Plataforma Firebase:	Pago por uso / Gratuito con límites
Coste Software:	0,00 €
Coste Hardware:	0,00€

Por lo tanto, el coste viene determinado por los recursos humanos y los servicios de Firebase que se utilicen teniendo en cuenta que son gratuitos siempre que no se superen los limites del plan *Spark.*

Teniendo en cuenta los costes del proyecto debe establecerse un plan de viabilidad para obtener beneficios con la ejecución del proyecto.

12.2.Monetización

La ejecución de este proyecto parte de la petición de una artesana que desea disponer de su propio sitio web para vender sus productos. Una de las opciones para monetizar el proyecto sería cargar el total del coste del proyecto más el margen a la persona que lo ha solicitado. No obstante, el coste puede resultar demasiado elevado. Ante esta situación se opta por estandarizar la aplicación web para posteriormente personalizarla en ciertos aspectos (página de presentación y el *theme* aplicado). Tomando esta estrategia se puede repercutir el coste del proyecto entre varios clientes.

En nuestro caso para monetizar la aplicación se ha optado por definir distintas tarifas por uso mensual en función de ciertos parámetros. A continuación, se muestran algunos parámetros que pueden tenerse en cuenta:

- **Funcionalidades**: puede definirse un precio que sólo incluya la publicación del catálogo y otro que incluya la opción de realizar las ventas.
- **Cantidad de productos:** puede definirse diferentes precios en función de la cantidad de productos que haya publicados en el catálogo.
- **Cantidad ventas:** puede definirse distintos precios en función de las ventas que se realicen desde la aplicación web.

Así pues, puede definirse los siguientes planes:

	Básico	Medio	Avanzado
Catálogo	SI	SI	SI
Módulo Ventas	-	SI	SI
Cantidad Productos	<100	<100	-
Cantidad Ventas	-	< 10/mes	-
Precio	9 €	14 €	20 €

*La subscripción a la plataforma Firebase no está incluida en el precio

Figura 31: Plan de precios

Teniendo en cuenta estos precios para tener una orientación sobre las ventas necesarias para amortizar el proyecto, se establece los siguientes escenarios:

- **Situación 1:** 1 año: 15 planes básicos + 12 planes medios
- **Situación 2**: 1 año: 7 planes básicos + 8 planes medios + 6 planes avanzados

13. Proceso de desarrollo

Para agilizar el proceso de desarrollo se ha partido de una plantilla de inicio (Trajan, 2020). Esta plantilla de inicio facilita una estructura de directorios y librerías que permiten iniciar rápidamente el desarrollo de la aplicación. Esta plantilla de trabajo, entre otros, contempla:

- Angular 10
- Patrón Redux
- Local Storage
- Lazy Loading Modules
- Angular Material
- Temas personalizados (4 temas)

A continuación pasa a detallarse a grandes rasgos la estructura del código fuente asociado al proyecto.

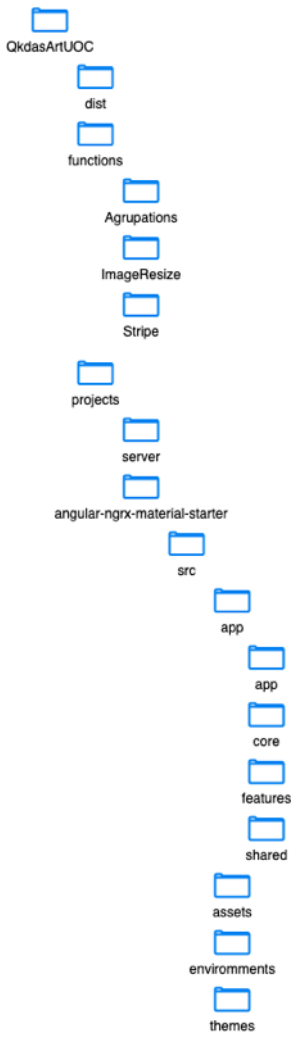


Figura 32: Estructura proyecto

Dist

En esta carpeta se encuentra la aplicación compilada.

Functions

En esta carpeta se encuentra las funciones tipo *Cloud Functions* que deben desplegarse en el servidor de *Firebase.* Se tiene tres tipos de función:

- *Agrupations*: funciones encargadas de actualizar datos que hacen referencia a las agrupaciones.
- *ImageResize*: funciones encargadas con la gestión de las imágenes.
- *Stripe*: funciones encargadas de la gestión de los pagos.

Projects → Server

En esta carpeta se encuentran todos los ficheros relacionados con el desarrollo de la parte servidor de la aplicación. En este proyecto se delega la gestión de la parte servidor a *Firebase* por lo que no se hace uso de esta carpeta.

Projects → angular-ngrx-material-starter

En esta carpeta se encuentra todo el código correspondiente a la parte *Front End* de la aplicación.

Angular-ngrx-material-starter → src

En esta carpeta se encuentran los distintos ficheros de configuración de nuestro proyecto así como las hojas de definición de estilo general.

Angular-ngrx-material-starter → src → app → core

En esta carpeta se encuentran todos los servicios y funcionalidades asociadas al patrón REDUX. Para cada una de las funcionalidades de la aplicación web se tiene los siguientes ficheros:

- filename.action.ts: *actions* patrón REDUX.
- filename.effects.ts: *effects* patrón REDUX.
- filename.models.ts: *models*.
- filename.reducers.ts: *reducers* patrón REDUX.
- filename.selectprs.ts: *selectors* patrón REDUX.
- filename.service.ts: *services*.

Angular-ngrx-material-starter → src → app → shared

En esta carpeta se encuentran utilidades que pueden ser usadas a lo largo de todo el desarrollo de la aplicación. Entre estas utilidades se encuentran las *directives*, *pipes* y los *validators*. En este módulo también se realiza la importación de módulos que son utilizados a lo largo de toda la aplicación como por ejemplo las funcionalidades de *Angular Material* y formularios entre otros.

Angular-ngrx-material-starter → src → app → features

En esta carpeta se encuentra el desarrollo de las distintas funcionalidades de las que consta nuestra aplicación. En el desarrollo de las distintas funcionalidades se ha seguido un mismo criterio en el que se ha

buscado en gran medida el desacoplamiento entre componentes. Para ello, en el desarrollo se ha diferenciado la parte visual de la parte encarga de gestionar las peticiones al patrón REDUX. La comunicación entre ambos componentes se realiza mediante parámetros de entrada/salida.

Así pues, por ejemplo la funcionalidad de login tiene la siguiente estructura de ficheros:

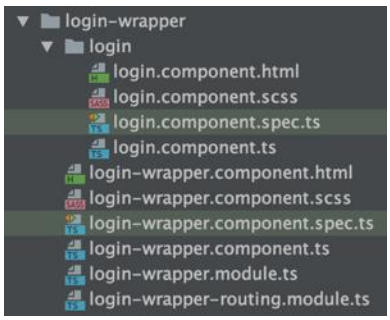


Figura 33: Estructura funcionalidad login

Angular-ngrx-material-starter → src → app → assets

En esta carpeta se encuentran las fuentes, imágenes, *javascript,* iconos, … utilizados por la aplicación.

Una vez definida la estructura de nuestro proyecto, en el siguiente punto pasa a detallarse las diferentes librerías de las que se ha hecho uso en el desarrollo del proyecto y el motivo de su utilización.

14. APl’s utilizadas

El desarrollo de la aplicación se ha realizado con el soporte de ciertas librerías que han agilizado en gran medida ciertos trabajos. A continuación pasa a detallarse las librerías utilizadas tanto para el desarrollo del *Front End* cómo de las *Cloud Functions.*

14.1. fs-extra

Esta librería incluye métodos que no están incluidos en el módulo *fs nativo.* En este proyecto se ha hecho uso de esta librería en la *Cloud Function* encargada de generar las diferentes resoluciones de imagen.

14.2. sharp

Esta librería permite redimensionar fácilmente distintos formatos de imagen. La aplicación es *responsive* adaptándose a los distintos tamaños de pantalla. Esto también incluye el facilitar la resolución de imagen óptima de cada pantalla evitando así enviar imágenes con un peso superior al necesario provocando un tráfico innecesario.

En esta aplicación el usuario puede adjuntar varias imágenes a un mismo producto. El usuario puede adjuntar imágenes con la resolución que desee. No obstante, una vez llegan al servidor se redimensionan quedando los siguientes anchos de imagen: 896px, 576px, 288px y 168px.

En la aplicación aparecen imágenes en distintos apartados. En algunos de ellos, el tamaño de imagen es fijo y/o tienen un ancho inferior a 168px por lo que es esta última resolución de imagen la que se utiliza. Ejemplos de esta situación son: detalle pedidos, cesta de la compra, listado de productos y formulario de productos. En otros apartados como por ejemplo el catálogo, el tamaño de imagen depende del tamaño de pantalla por lo en cada situación se elige la resolución de imagen que más se aproxima al tamaño óptimo.

14.3.nodemailer y nodemailer-smtp-transport

Estas librerías permiten el envío de emails desde aplicaciones *Node.js*. La aplicación, una vez creado el pedido, envía al cliente un email de confirmación con los datos del pedido para así poder realizar su seguimiento.

14.4. stripe

Esta librería permite el acceso a la API de *Stripe* desde aplicaciones servidor escritas con *JavaScript.* En la aplicación desde el servidor de *Firebase* se solicita un identificador de sesión a Stripe para poder realizar el pago desde la aplicación. Y una vez efectuado el pago, en el servidor se recibe su confirmación.

14.5. uuid

Esta librería permite la creación de UUID[5].

14.6. ngrx

Para agilizar la aplicación del patrón REDUX se ha hecho uso de varias librerías entre las que hay: *ngrx/effects, ngrx/entity, ngrx/router-store, ngrx/store, ngrx/store-devtools.* Algunas de estas librerías reducen en gran medida el código necesario para la implementación del patrón mientras que otras se centran en la visualización de los resultados.

14.7. Immer

Uno de los principios en los que se basa el patrón de diseño REDUX es que un determinado estado es inmutable. Este principio hace que a veces resulte laborioso determinar un nuevo estado a partir del anterior, principalmente en el caso de haber objetos imbricados. Esta librería permite definir el siguiente estado a partir del actual.

14.8. cripto-js

Esta librería permite la encriptación de datos mediante los algoritmos de encriptación estándar. En la aplicación, cuando un usuario realiza el *login* se almacenan los datos de conexión en el *local storage.* Al tratarse de datos sensibles, éstos se encriptan. De esta forma, si el usuario no realiza la desconexión, en posteriores conexiones no deberá identificarse nuevamente.

14.9. angular/fire

Esta librería permite trabajar con Firebase según las convenciones de trabajo de Angular.

[5] Un identificador único universal o universally unique identifier (UUID) es un número de 16 bytes (128 bits). Por tanto, el número de posibles UUID es de 1632, o unos 3,4 × 1038. En su forma canónica un UUID se expresa mediante 32 dígitos hexadecimales divididos en cinco grupos separados por guiones de la forma 8-4-4-4-12 lo que da un total de 36 caracteres (32 dígitos y 4 guiones). Por ejemplo: 550e8400-e29b-41d4-a716-446655440000

14.10. angular/material

El diseño de la interface gráfica de la aplicación se ha realizado con esta librería. *Angular Material* es una librería de estilos (como *Bootstrap*) basada en la guía de diseño de *Material Design*[6], realizado por el equipo de Angular para integrarse perfectamente con Angular.

14.11. hammer.js

Esta librería permite reconocer, entre otros, eventos de *touch, drag, swipe …* En la aplicación, en el catálogo de productos el usuario puede visualizar las distintas fotos de un determinado producto. Para navegar entre las distintas fotos se dispone de un foto a derecha e izquierda de la foto que se está visualizando. No obstante, en los dispositivos con pantalla táctil los usuarios están acostumbrados a realizar gestos sobre la misma pantalla como puede ser el deslizamiento del dedo de derecha a izquierda y a la inversa para navegar entre las distintas fotos. Es en esta situación donde se ha hecho uso de esta librería.

14.12. ng-lazyload-image

Esta librería permite el *lazy load* de las imágenes. Esta técnica a grandes rasgos lo que hace es ir cargando las imágenes a medida que se van necesitando. Esto hace que la carga inicial de las páginas sea mucho más eficiente principalmente en el caso que haya muchas imágenes por descargar.

14.13. ngx-virtual-scroller

En el caso de realizar listados de muchos registros, aunque sólo se visualice por pantalla una pequeña parte de los registros cargados, internamente en el DOM se tiene todos los registros. Esto hace que el proceso de visualización y navegación no esté optimizado. Para solucionar esta situación *Angular Material* dispone del componente *<cdk-virtual-scroll-viewport>* que permite cargar eficientemente largas listas de datos. Este componente dibuja y carga en el DOM sólo aquellos elementos que se visualizan. Este componente de *Angular Material* se ha utilizado para realizar el *Listado de Productos* que permite la gestión de los mismos. En este listado se realiza una carga inicial de productos un poco superior a la que se puede visualizar por pantalla y posteriormente se van realizando nuevas peticiones al servidor a medida que se van requiriendo nuevos datos.

[6] Material design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma. Fue desarrollado por Google y anunciado en la conferencia Google I/O celebrada el 25 de junio de 2014. Ampliando la interfaz de tarjetas vista por primera vez en Google Now.

Sin embargo, este componente de *Angular Material* no funciona para el caso en que en una misma fila se dibuje más de un elemento como ocurre en nuestro *Catálogo* ya que en función del ancho de pantalla en una misma fila se pueden mostrar varios productos. Es aquí donde se hace el uso de la librería *ngx-virtual-scroller* siguiendo no obstante el mismo criterio definido en el proceso de carga de datos del *Listado de Productos*.

15. Instrucciones de implantación

En el proceso de implantación hay que tener en cuenta los distintos servicios web que intervienen en la aplicación:

- *Firebase*
- *Stripe*

Para poder hacer uso de estos servicios tiene que abrirse una cuenta en cada uno de ellos y realizar las personalizaciones que se detallan en los siguientes puntos.

15.1. Firebase

Para abrir una cuenta en *Firebase* hay que identificarse con una cuenta de *Gmail* y seguidamente crear un proyecto. Una vez creado el proyecto, puede consultarse todos los datos de configuración que van a ser necesarios para poder realizar el despliegue de la aplicación.

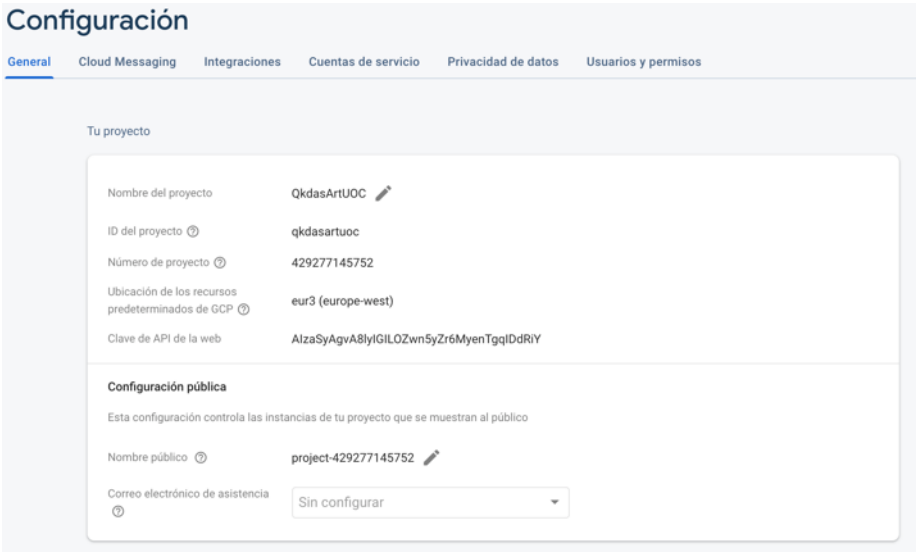


Figura 34: Datos configuración proyecto Firebase

Inicialmente, el plan de precios aplicado es el *Plan Spark*. Este plan es totalmente gratuito aunque tiene algunas limitaciones que impiden su uso en este proyecto. La principal, es que no permite el uso de API's externas a *Firebase,* y en este proyecto es necesario trabajar con el API de *Stripe*. Por este motivo se debe seleccionar el *Plan Blaze*. Este plan sigue teniendo las opciones gratuitas del *Plan Spark* pero a partir de ciertos parámetros de uso se aplica una tarifa de precios.

Una vez seleccionado el plan de precios correcto, ya se puede pasar a activar los servicios que se van a utilizar. Estos servicios son:

- *Authentication* (la autentificación que se utiliza en este proyecto es por email y contraseña)
- *Firestore* (base de datos NoSQL)
- *Storage* (almacén imágenes)
- *Hosting* (alojamiento aplicación de Angular)
- *Functions* (ejecución de funciones desde el servidor)

Una vez activados los servicios, en la base de datos NoSQL es necesario crear la colección *Orders* y el documento con *id=orderInfo* y campo *id=0*. Este campo es utilizado como contador de los pedidos.

15.2.Stripe

Al igual que *Firebase*, para poder hacer uso de los servicios de *Stripe* hay que abrir una cuenta y crear un proyecto. Una vez creado el proyecto, en la sección *Desarrolladores,* puede consultarse las claves necesarias para hacer uso de la API.

En nuestro proyecto, una vez *Stripe* recibe el pago, envía la confirmación al servidor de Firebase mediante una *url* de activación que hay que informar en el apartado *Webhooks*. El valor de esta *url* no se conoce hasta que no se realiza el despliegue de las *Cloud Functions* del proyecto que se pasa a detallar en el siguiente apartado.

15.3. Proyecto Angular

Junto con el proyecto se adjunta el código fuente de la aplicación. Una vez descomprimido el código fuente, el primer paso a realizar es instalar todas las dependencias del proyecto. Para ello se debe acceder a la carpeta raíz que es *QkdasArtUOC* y ejecutar el comando `npm install`. Este mismo comando debe ejecutarse en la carpeta *QkdasArtUOC → functions* para instalar las dependencias de *las Cloud Functions.*

Vinculación Firebase con Proyecto de Angular

Una vez ejecutados los pasos anteriores, ya se tiene el proyecto instalado localmente. Ahora debe vincularse el proyecto de Angular con el proyecto creado en *Firebase*. Para ello, debe seguirse los siguientes pasos:

PASO 1

Ejecutar el comando `firebase login` para conectar con nuestra cuenta de Firebase. Seguidamente con el comando `firebase projects:list` se puede ver todos los proyectos de *Firebase* vinculados a la cuenta.

PASO 2

En *QkdasArtUOC → projects* debe informarse el proyecto de *Firebase* al que se quiere vincular el proyecto de Angular.

```
{
  "projects": {
    "default": "qkdasartuoc"
  }
}
```

PASO 3

Inicializar el proyecto con el comando *firebase init*. En este proceso debe seleccionarse las funcionalidades que van a utilizarse que en nuestro proyecto son:

- Firestore
- Functions
- Hosting
- Storage

Una vez seleccionadas las funcionalidades, se solicita los siguientes datos:

```
What file should be used for Firestore Rules? (firestore.rules)
File firestore.rules already exists. Do you want to overwrite it with the Firestore Rules from
the Firebase Console? Y
What file should be used for Firestore indexes? (firestore.indexes.json)
? What language would you like to use to write Cloud Functions? (Use arrow keys) javascript
? Do you want to use ESLint to catch probable bugs and enforce style? Y
? File functions/package.json already exists. Overwrite? N
? File functions/.eslintrc.json already exists. Overwrite? N
? File functions/index.js already exists. Overwrite? N
? File functions/.gitignore already exists. Overwrite? N
? Do you want to install dependencies with npm now? Y
? What do you want to use as your public directory? dist/angular-ngrx-material-starter
? Configure as a single-page app (rewrite all urls to /index.html)? Y
? File public/404.html already exists. Overwrite? N
? File public/index.html already exists. Overwrite? N
? What file should be used for Storage Rules? (storage.rules)
```

Con este proceso ya se tiene el proyecto de Angular vinculado al proyecto de Firebase.

Cloud Functions

En la función relacionada con Stripe hay que realizar una serie de modificaciones. Ésta se encuentra en *functions → stripe → stripeFunctions.js* y los cambios a realizar son:

PASO 1

Indicar la *secret key* del API de Stripe en la línea de código que se indica:

```
const stripe = require('stripe')('sk_test_api_stripe')
```

PASO 2

Al abrir una sesión de Stripe, debe informarse a donde debe redirigirse al usuario una vez realizado el pago tanto si éste se efectúa con éxito como si ocurre cualquier fallo. Para ello debe informarse las dos *url's*. En cada instalación deberá informarse su dominio. Al realizar el *hosting* en Firebase, en el caso de no disponer todavía de un dominio facilita uno de pruebas. Los cambios a realizar son:

```
const session = await stripe.checkout.sessions.create({
      payment_method_types: ['card'],
      line_items: items,
      customer_email: `${userDoc.data().email}`,
      mode: 'payment',
      success_url:
`https://qkdasartuoc.web.app/#about/?stripe=success&order=${req.body.order`,
      cancel_url:
' https://qkdasartuoc.web.app/#/cart/?stripe=error ',
    });
```

PASO 3

Finalmente, una vez creado el pedido de compra, se envía un mail de confirmación al usuario. Es por ello que se debe informar los parámetros de configuración de la cuenta desde la que se va a enviar los *emails.*

```
var transporter = nodemailer.createTransport(smtpTransport({
        service: 'gmail',
        host: 'smtp.gmail.com',
        auth: {
          user: 'qkdasart@gmail.com',
          pass: 'password'
        }
}));
```

Front End

En el proyecto de Angular debe configurarse el valor de ciertas variables para asociar nuestro proyecto de Angular con los proyectos de *Firebase* y *Stripe*.

Se debe ir *a projects→ angular-ngrx-material-starer → environments* y definir en *environment.prod.ts* y *environment.ts* ciertos parámetros.

PASO 1

Debe informarse la *public key* del proyecto de *Stripe*.

```
pkStripeTest: 'pk_test_Stripe API'
```

PASO 2

La solicitud de inicio de sesión a *Stripe* se inicia desde el servidor de *Firebase* mediante la activación de la función *checkout* de *Cloud Functions*. Es por ello que debe informarse la *url* de activación de dicha función. El valor de la *url* se conocerá en el momento de desplegar las *Cloud Functions*.

```
stripeCheckout: 'https://us-central1-qkdasartuoc.cloudfunctions.net/checkout'
```

PASO 3

Finalmente también debe informarse los parámetros de nuestro proyecto de Firebase.

```
  firebaseConfig: {
    apiKey: 'AIzaSyAgvA8lyIGILOZwn5yZr6MyenTgqIDdRiY',
    authDomain: 'qkdasartuoc.firebaseapp.com',
    databaseURL: 'https://qkdasartuoc.firebaseio.com',
    projectId: 'qkdasartuoc',
    storageBucket: 'qkdasartuoc.appspot.com',
    messagingSenderId: '429277145752',
    appId: '1:429277145752:web:2a5394022eeb27ebeabeed',
    measurementId: 'G-0L6GZKF89N'
}
```

Despliegue del proyecto

En este proceso debe desplegarse tanto las *Cloud Functions* como el proyecto de Angular al servidor de *Firebase*.

PASO 1

Para desplegar las *Cloud Functions* debe ejecutarse el siguiente comando: `firebase deploy –only functions`. En este proceso Firebase proporcionará las *url's* de activación de las funciones *checkout* y *hooks* relacionadas con *Stripe*. Cabe recordar que la *url* relacionada con la función *hooks* debe informarse en la cuenta de *Stripe* y la *url* relacionada con la función *checkout* debe informarse en los parámetros del proyecto de Angular.

PASO 2

Finalmente hay que hacer la compilación del proyecto mediante el comando `firebase build –prod`. Los compilados resultantes se encuentran en *projects* → *angular-ngrx-material-starer* → *dist* y son los ficheros que debe subirse al servidor. Esta subida de los ficheros se realiza con el `comando firebase deploy –only hosting`.

15.4. Datos proyecto de pruebas

A continuación se detalla los datos de acceso a nuestro proyecto.

La url de acceso es: https://qkdasartuoc.firebaseapp.com/

En usuario administrador tiene las siguientes credenciales:

Email: xferraserra@gmail.com
Password: 123456

También hay dado de alta el siguiente perfil de comprador:

Email: nsmoris@gmail.com
Password: 123456

16.Proyección a futuro

En este proyecto se ha desarrollado las funcionalidades básicas de una aplicación web que permite la venta *online.* No obstante, se dispone de una aplicación web fácilmente escalable que permite la incorporación de mejoras. Entre estas mejoras puede diferenciarse entre:

- Correcciones y optimización código existente: se trata de corregir i mejorar las funcionalidades existentes.
- Nuevas funcionalidades: se trata de añadir nuevas funcionalidades a nuestra web que le den más valor.

16.1.Correcciones y optimización código existente

En producto resultante de nuestro proyecto es una PWA[7]. Existen multitud de herramientas que permiten auditar aplicaciones web. Una de estas herramientas es Lighthouse, que da un informe muy completo que incluye aspectos de rendimiento, accesibilidad, mejores prácticas y SEO.

En la aplicación existe el listado de productos y catálogo de productos que permiten realizar una selección por *agrupación.* No obstante, en estos listados podría añadirse nuevas opciones de filtrado como por ejemplo precio, novedades, …

Otro listado que podría mejorarse, es el listado de pedidos. En este listado no se ha aplicado la librería *ngx-virtual-scroller* para optimizar la carga de datos en el navegador ni tampoco se realiza la petición de datos mediante cargas *batch.* Una mejora en este listado sería la aplicación de estas optimizaciones. En este listado también sería importante implementar filtros de selección para facilitar la localización de pedidos. Esta funcionalidad es especialmente importante para el administrador de la página ya que puede tener muchos pedidos.

16.2. Nuevas funcionalidades

Las limitaciones de tiempo en el desarrollo de este proyecto ha hecho que haya tenido que limitarse las funcionalidades implementar. Sin embargo hay algunas mejoras que podrían dar más valor añadido a nuestra aplicación. A continuación se cita algunas de estas mejoras.

[7] Una **aplicación web progresiva** (**PWA** por sus siglas en inglés) es un tipo de software de aplicación que se entrega a través de la web, creado utilizando tecnologías web comunes como HTML, CSS y JavaScript. Está destinado a funcionar en cualquier plataforma que use un navegador compatible con los estándares. La funcionalidad incluye trabajar sin conexión, notificaciones push y acceso al hardware del dispositivo, lo que permite crear experiencias de usuario similares a las aplicaciones nativas en dispositivos móviles y de escritorio.

- **Nuevas opciones de registro**. Posibilidad de registrase mediante la cuenta de *Google* y *Facebook.*
- Posibilidad de recuperar la contraseña.
- **Gestionar las formas de pago** de forma que los usuarios no deban introducir los datos de la tarjeta en cada compra.
- Posibilidad de realizar los pagos mediante ***Paypal***.
- **Creación de un chat** para que visitantes y vendedor puedan comunicarse y por ejemplo poder personalizar los productos.
- **Localización de los pedidos** en tránsito mediante una vinculación a la página de seguimiento de la empresa encargada de realizar la expedición.

17. Conclusiones

El proyecto planteado inicialmente se ha podido implementar satisfactoriamente. El producto obtenido permite definir un catálogo de productos y realizar su venta. Como se ha comentado anteriormente, en función de la aceptación del sitio web hay multitud de mejoras que pueden irse incorporando al producto.

Aunque ya se había trabajado durante el máster con el *framework* Angular y el patrón *Redux* ha sido durante el proyecto donde se ha profundizado y visto con más detalle todas sus ventajas. En este proyecto el patrón *Redux* ha sido de gran utilidad en el proceso de revisión del código.

En cuanto a la gestión del *backend,* como se ha visto, ha sido delegada a *Firebase.* Esto ha permitido centralizar los esfuerzos en el desarrollo del *frontend* ya que el uso de distintos servicios disponibles dentro de *Firebase* ha facilitado en gran medida el trabajo. Sin embargo, inicialmente ha supuesto un esfuerzo adicional el adquirir los conocimientos necesarios.

Aunque inicialmente tenía cierto temor a los problemas que pudiera surgir en la inclusión de la pasarela de pago, finalmente se ha podido realizar sin ningún tipo de problema. Aunque la versión del API utilizada era bastante reciente, la documentación existente así como los ejemplos disponibles en la página de *Stripe,* es suficiente para poder realizar la implementación.

El desarrollo de una aplicación web requiere de conocimientos transversales que incluyen entre otros, el diseño, desarrollo *frontend* y *backend*. Tener conocimientos exhaustivos de todos los campos a veces puede resultar costoso.

19.Anexos

19.1.Entregables del proyecto

A continuación se detalla los entregables del proyecto así como su descripción.

Archivo: **PAC_FINAL_Ferrarons_Serra_Xavier.zip.**
Contiene la entrega completa, es decir, la documentación, la presentación, el video de presentación y el proyecto.

Archivo: PAC_FINAL_prj_Ferrarons_Serra_Xavier.
Contiene el front-end y *Cloud Functions* del *back-end*.

Archivo: PAC_FINAL_mem_Ferrarons_Serra_Xavier.pdf
Contiene la memoria del proyecto.

Archivo: PAC_FINAL_vid_ Ferrarons_Serra_Xavier
Contiene el video de defensa del proyecto en formato mp4, pdf, y pptx.

Archivo: PAC_FINAL_prs_ Ferrarons_Serra_Xavier.
Contiene la presentación pública del proyecto en formato mp4, pdf y pptx.

19.2.Bibliografía

Angular. 2020. Documentación de Angular. *Angular.* [En línea] 2020. https://angular.io/docs.

Angular Material. 2020. Componentes de Angular Material. *Angular Material.* [En línea] 2020. https://material.angular.io/components/categories.

Arenzana, David. 30. Prinncipios de usabilidad de Jakob Nielsen. [En línea] 2016 de marzo de 30. https://es.semrush.com/blog/usabilidad-web-principios-jakob-nielsen/.

Colectivo, Convenio Estatal. 2018. *Boletín Oficial del Estado.* [En línea] 2018. https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf.

El Observatorio Cetelem. 2020. ¿Conoces los nuevos hábito de consumo 2020? [En línea] 16 de 01 de 2020. https://elobservatoriocetelem.es/nuevos-habitos-de-consumo-2020/.

Firebase. 2020. Documentación de Firebase. *Firebase.* [En línea] 2020. https://firebase.google.com/docs?hl=es-419.

—. 2020. Planes de precios. *Firebase.* [En línea] 2020. https://firebase.google.com/pricing?hl=es-419#blaze-calculator.

Hernández, Juan. 2019. 32,6 millones de españoles navegan a diario con su móvil. [En línea] 06 de 11 de 2019. https://blog.hostalia.com/informes/326-millones-espanoles-navegan-diario-movil/.

Indeed. 2020. *Salarios para emleos de programador/a junior en España.* [En línea] 15 de 10 de 2020. https://es.indeed.com/salaries/programador-junior-Salaries.

ngrx. 2020. Documentación de ngrx. *ngrx.* [En línea] 2020. https://ngrx.io/docs.

npm. 2020. Buscador librerías javascript. *npm.* [En línea] 2020. https://www.npmjs.com/.

Pexels. 2020. Pexels, buscador imágenes. [En línea] 2020. https://www.pexels.com/es-es/.

Stripe. 2020. Documentación de Stripe. *Stripe.* [En línea] 2020. https://stripe.com/docs.

Trajan, Tomas. 2020. *Angular ngrx material starter.* [En línea] 2020. https://github.com/tomastrajan/angular-ngrx-material-starter.

Unsplash. 2020. Unsplash, Photos for everyone. [En línea] 2020. https://unsplash.com/.