ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

**Desarrollo de una plataforma marketmaker para comercios locales.**

**Development of a marketmaker platform for local businesses.**

Realizado por

**Javier Merino Aranda**

Tutorizado por

**Francisco José Jaime Rodríguez**

Departamento

**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2023

Fecha defensa:

Resumen

En este trabajo de fin de grado se realiza una aplicación de escritorio destinada a todo tipo de comercio local, en la que se ofrece la posibilidad a aquellos pequeños comercios que se han quedado estancados frente al rápido avance de las tecnologías de superar dicho obstáculo tanto tecnológico como económico que supone el coste de una aplicación a medida para un comercio frente al simple registro que se propone con la aplicación desarrollada. Muchos pequeños comercios no pueden permitirse ese gasto y algunos aún pudiendo tampoco lo hacen por el elevado coste de desarrollo, por lo que con esta plataforma que se ha desarrollado se facilitaría su expansión al mundo del comercio electrónico ofreciendo sus productos a los usuarios que se registren en la aplicación. Para llevar a cabo la realización de este proyecto, las principales tecnologías que se han empleado son Java, JavaFX y MySQL, haciendo uso de otras tecnologías para realizar todo el proceso que se lleva a cabo en un proyecto software atendiendo siempre a las exigencias de un trabajo de fin de grado.

El objetivo final es permitir que los comercios locales publiquen sus productos y puedan romper el muro que los separa del mundo de Internet, principalmente ante el auge de grandes marcas que han establecido su modelo de negocio en el comercio electrónico, como Amazon o eBay entre otros, donde se ofrece un amplio y diverso catálogo de productos. De esta manera, los usuarios tendrán la oportunidad de realizar compras en los comercios locales que se encuentren registrados en la aplicación, contribuyendo al aumento del número de ventas del comercio.

**Palabras clave: Java, JavaFX, casos de uso, SceneBuilder, base de datos, requisitos.**

Abstract

In this final degree project, a desktop application is developed for all types of local commerce, which offers the possibility to small businesses that have remained stagnant in the face of the rapid advance of technologies to overcome this obstacle and to expand into the world of e-commerce by offering their products to users who register in the application. To carry out this project, the main technologies used are Java, JavaFX and MySQL, making use of other technologies to carry out the whole process that takes place in a software project, always considering the requirements of a final degree project.

The final objective is to allow local businesses to publish their products and break the wall that separates them from the Internet world, mainly due to the rise of large brands that have established their business model in e-commerce, such as Amazon or eBay among others, where a wide and diverse catalogue of products is offered. In this way, users will have the opportunity to make purchases in local businesses that are registered in the application, contributing to an increase in the number of sales in commerce.

**Keywords: Java, JavaFX, SceneBuilder, database, use case, requirements.**

Índice

Resumen 1

Abstract 1

Índice 1

Introducción 1

1.1 Motivación 1

1.2 Objetivos 2

1.3 Estructura de la memoria 2

1.4 Tecnologías de desarrollo 3

1.5 Metodología de trabajo 4

Análisis 5

2.1 Obtención de requisitos 5

2.2 Actores 6

2.3 Requisitos funcionales 6

2.3.1 Requisitos administrador 6

2.3.2 Requisitos del cliente 9

2.3.3 Requisitos del comercio 11

2.3.4 Requisitos usuario anónimo 13

2.4 Requisitos no funcionales 14

2.5 Requisitos a futuro 16

Diagramas del sistema 19

3.1 Contexto y objetivo 19

3.2 Casos de uso 19

3.2.1 Administrador 20

3.2.2 Comercio 25

3.2.3 Cliente 31

3.2.4 Anónimo 35

3.3 Diagramas de secuencia 38

3.3.1 Registrar usuario 39

3.3.2 Solicitud acceso a la plataforma 40

3.3.3 Añadir producto al carrito 41

3.3.4 Creación de comercio 41

3.4 Diagrama de clases 42

3.4.1 Diagrama de clases lógico 43

Implementación de la aplicación 52

4.1 Visión general 52

4.2 Planificación 52

4.3 Prototipado 54

4.4 Desarrollo de la aplicación 65

4.5 Base de datos 67

4.6 Front-end de la aplicación 67

4.7 Pruebas a la aplicación 68

4.8 Problemas surgidos en el desarrollo 72

Conclusiones 74

5.1 Conclusiones 74

5.2 Mejoras de la plataforma en un futuro 75

Referencias 77

1

Introducción

1.1 Motivación

El uso de Internet, junto con los comercios electrónicos, está en pleno auge, donde se ha observado un gran aumento del uso de estos tras la reciente pandemia de COVID-19. Debido a esto, muchos comercios locales han quedado estancados al no dar el salto al mundo del comercio electrónico, perdiendo la oportunidad de expansión y las facilidades que conlleva estar expuesto en Internet gracias a la fácil accesibilidad que hay hoy en día.

Es por ello por lo que grandes empresas multinacionales que hoy en día son pioneras en el sector del comercio electrónico como Amazon o eBay ofrecen una amplia y diversa variedad de productos en sus plataformas que, como consecuencia, atraen a una gran cantidad de usuarios hacia estos comercios para realizar sus compras.

Con esta plataforma desarrollada en este trabajo de fin de grado, se busca ayudar a superar los obstáculos iniciales que supone para muchos pequeños comercios el dar el salto al comercio electrónico, tales como el coste de desarrollo de un portal/aplicación a medida y el desconocimiento de muchos de ellos a la hora de hacer uso de estas tecnologías.

Por ello, este trabajo de fin de grado tiene como objetivo dar la oportunidad, y en la medida de lo posible, sobrepasar la barrera del mundo electrónico debido al poco conocimiento de las tecnologías con las que se trabajan en los comercios electrónicos, publicando así sus productos en línea. Esto ayudará a aumentar el conocimiento de la marca y facilitar las compras, proporcionando un complemento a la presencia física del cliente tradicional.

1.2 Objetivos

Este trabajo de fin de grado aborda el desarrollo de una arquitectura cliente-servidor, donde el servidor alojará una base de datos SQL que almacenará la información necesaria, y el cliente será una aplicación de escritorio en Java que lanzará peticiones contra dicho servidor. Para simplificar el proceso de desarrollo, el servidor se implementa en modo local. En caso de pasar a producción y ser publicado en las redes, se desplegaría en la nube.

Los comercios (uno de los distintos roles de la aplicación) podrán darse de alta en la plataforma tras la aprobación de un usuario administrador y así publicar sus productos haciéndolos accesibles para los usuarios de tipo cliente. Estos podrán visualizar los mismos y realizar comprar si así lo desean, añadiéndolos a un carrito de compra y pasando por una pasarela de pago una vez confirmen la selección de productos, donde cabe mencionar que la implementación de dicha pasarela de pago queda excluida de los requisitos de este trabajo de fin de grado, ya que usualmente dicha plataforma de pago es proporcionada por las entidades bancarias.

1.3 Estructura de la memoria

La memoria se ha organizado en base a las distintas etapas del desarrollo del proyecto, quedando estructurada de la siguiente forma:

**Capítulo 1: Introducción.** En esta sección se exponen las razones que motivan a realizar este proyecto, estableciendo unos objetivos que cumplir para su elaboración tal y como se ha planeado en un principio.

**Capítulo 2: Análisis.** En este capítulo se llevará a cabo un completo análisis del sistema que se desarrolla mediante la identificación de los requisitos funcionales, no funcionales, y los requisitos a futuro en caso de que la aplicación pasara a producción y hacerla totalmente funcional.

**Capítulo 3. Diagramas del sistema.** Durante este capítulo se presentan gráficamente los distintos diagramas realizados en este proyecto de software para alcanzar la solución del sistema, donde la explicación del proceso, así como la explicación de los diagramas, será más fácil de entender gracias a los diagramas que se han realizado.

**Capítulo 4. Implementación de la aplicación.** Se llevará a cabo una explicación de la implementación de la aplicación y su desarrollo en las tecnologías inicialmente escogidas.

**Capítulo 5. Conclusiones.** Conclusiones extraídas a lo largo del desarrollo del proyecto y posibles mejoras en caso de que la aplicación pasara a producción.

1.4 Tecnologías de desarrollo

En este apartado se detallan las principales tecnologías que se han usado a lo largo del desarrollo e implementación del proyecto:

* **Trello:** Trello es uno de los softwares de administración de proyectos más utilizados en la actualidad. En este software se emplea el sistema Kanban con tarjetas, permitiendo utilizar listas para la clasificación de las tareas e ideas que han sido recopiladas durante el análisis del proyecto, agregar comentarios, agrupar tareas y asignar etiquetas en base a su importancia en la implementación utilizando códigos de colores que será descrito más adelante.
* **MySQL Workbench:** MySQL Workbench es un entorno de diseño, gestión y administración de bases de datos de código libre que utiliza MySQL. Se caracteriza por permitir una gestión de manera fácil y sencilla de las bases de datos SQL, donde se puede obtener una mejor visibilidad de las bases de datos gracias a las herramientas visuales que proporciona, así como un gran ahorro de tiempo a la hora de realizar consultas cotidianas de los datos almacenados en la base de datos.
* **Java:** Java es un lenguaje de programación multiplataforma, orientado a objetos, seguro y confiable para codificar en cualquier plataforma que se utilice hoy en día, desde móviles y software empresarial hasta servidores. Las razones por las que se sigue eligiendo este lenguaje es por sus bibliotecas ya incorporadas, la comunidad activa que posee y la posibilidad de ejecutar el código en cualquier plataforma sin necesidad de reescribirlo.
* **Visual Paradigm:** Es una herramienta CASE que ayuda a los equipos de software a transformar los requisitos del sistema extraídos y consensuados a diagramas precisos y exactos, dando soporte completo para UML.
* **PencilProject:** Es un software de código abierto donde se permite realizar prototipos de interfaces (mockups) en cualquier plataforma de escritorio, fácil de usar y que posee una gran cantidad de plantillas para ahorrar el mayor tiempo posible.
* **Eclipse:** Es una de las mayores plataformas de desarrollo integrado (IDE) más populares para el desarrollo de código, en la que se permite programar en lenguajes como C, C++, Java, JavaScript, Kotlin…
* **SceneBuilder:** Aplicación en la que se permite diseñar las interfaces de usuario de la aplicación que se desarrolla usando JavaFX.
* **JavaFX:** Lenguaje de programación combinado con Java con el que se crean y despliegan aplicaciones web y de escritorio en una amplia variedad de plataformas, tales como consolas de videojuegos, teléfonos móviles, la Web….
* **Java Database Connectivity (JDBC):** Interfaz de programación de aplicaciones (API) que permite que los programas realizados en Java establezcan una conexión a las bases de datos SQL.
* **Github:** Repositorio donde se aloja el proyecto utilizando el sistema de versiones Git. En este caso particular se ha utilizado el plugin de Github que proporciona Eclipse.
* **Amateras:** Plugin de Eclipse que se utiliza para la realización del diagrama de clases del proyecto, en el cual se puede añadir de forma automática las clases y métodos del proyecto.
* **FXML:** Lenguaje utilizado para la realización de Interfaces de una aplicación realizada en JavaFX.
* **JUnit:** Librería que se utiliza para la realización de pruebas unitarias en Java.

1.5 Metodología de trabajo

Se ha usado una metodología ágil derivada de Scrum con las adaptaciones requeridas para este proyecto, donde se han realizado entregas periódicas al tutor en base a lo que se ha planificado para esa entrega, siguiendo la división en sprints típica de *Scrum.*

La duración de cada sprint se ha determinado durante el progreso del proyecto en base a las tareas asociadas a cada uno, aunque se ha priorizado un periodo medio de dos semanas. Con esta metodología se ha podido permitir adaptar el desarrollo del proyecto a las necesidades del cliente. Por ello, antes del comienzo de cada sprint, se realiza una reunión donde se abarcan las modificaciones de los requisitos de la plataforma en caso de que haya sido necesario y se decide lo que se va a realizar durante las próximas dos semanas, manteniendo comunicación mediante correo electrónico en caso de cualquier duda relacionada con el proyecto, ya sean requisitos, implementación o elaboración de los diagramas. En caso de que fuese necesario, se realizaría una reunión de emergencia para abordar las dudas que hayan surgido durante el desarrollo del sprint que se encuentra en proceso.

Si el sprint es finalizado antes del tiempo establecido, se comunicaba al tutor la finalización de este para poner en marcha una nueva reunión y detallar el contenido que se va a realizar en el nuevo sprint.

2

Análisis

2.1 Obtención de requisitos

A continuación, se van a detallar los requisitos funcionales y no funcionales del sistema. Por ello, se ha tomado la decisión de clasificar los requisitos funcionales según el actor para profundizar y detallar lo máximo posible las funciones de cada actor, donde se va a poder observar que algunos requisitos en distintos actores son repetidos, pues las funcionalidades son similares, y por lo tanto, es el mismo requisito tanto para uno como para otro. Por otro lado, se ha llevado a cabo la obtención de unos requisitos a futuro, donde cabe mencionar que estos requisitos serían implementados en caso de que la aplicación pasara a producción, lo que le daría una funcionalidad extra adicional.

2.2 Actores

Antes de describir los requisitos, se van a describir los actores que participan en el sistema y las acciones que desempeñan en el mismo:

* **Usuario administrador:** Usuario con todos los privilegios sobre el sistema.
* **Usuario cliente:** Usuario que representa a los compradores de la plataforma.
* **Usuario anónimo:** Restringido a visualizar la plataforma y navegar por ella.
* **Usuario comercio:** Usuario que representa a los comercios inscritos en la plataforma.

2.3 Requisitos funcionales

Se procede a detallar los requisitos que describen la funcionalidad del sistema. Para ello, se ha decidido denotar los requisitos mediante las siglas RF seguidas de un número identificativo, ordenadas de manera ascendente, para una identificación más precisa y eficiente a la hora de explicar un requisito en concreto o de simplemente mencionarlo. En el caso de los requisitos que conlleven crear, visualizar, actualizar y eliminar (CRUD) serán desglosados en las partes nombradas. En caso de hacer algún comentario sobre la funcionalidad de un actor específicamente, se denotará al final de la tabla de requisitos.

2.3.1 Requisitos administrador

|  |  |  |
| --- | --- | --- |
| Identificador | Nombre | Descripción |
| RF-1 | Acceso a la plataforma | El administrador deberá iniciar sesión en la aplicación identificándose con un nombre de usuario y una contraseña. |
| RF-2 | CRUD administrador | Se podrá crear, borrar, actualizar y visualizar a un administrador. |
| RF-2.1 | Consultar administrador | Cualquier administrador podrá visualizar los datos de los demás administradores desde la base de datos. |
| RF-2.2 | Actualizar administrador | Cada administrador podrá cambiar en cualquier momento sus datos identificativos. Estos datos son el nombre, apellidos, y contraseña. Esta función se podrá realizar tanto en la base de datos como en el perfil de este una vez inicie sesión en la plataforma. |
| RF-2.3 | Eliminar administrador | Si un administrador es dado de baja de la plataforma, se eliminará toda información que le identifique quedando sin acceso a la plataforma y perdiendo así el rol que lo identifica. |
| RF-3 | Comprobación de solicitud | El administrador deberá comprobar la solicitud cumplimentada por el comercio, que podrá aceptar o rechazar. Una vez sea resuelta la solicitud, el comercio será notificado mediante correo electrónico. |
| RF-4 | CRUD comercio | El administrador podrá crear, visualizar, actualizar o borrar el comercio |
| RF-4.1 | Creación de comercio | Con la solicitud previamente aceptada, el administrador creará el comercio con los datos que figuran en dicha solicitud. Dicha creación será realizada en la base de datos del sistema, proporcionándole al comercio el usuario y la contraseña. |
| RF-4.2 | Visualización de comercio | El administrador podrá visualizar en todo momento el comercio y sus datos. |
| RF-4.3 | Actualizar comercio | El administrador puede actualizar los datos del comercio en caso necesario. |
| RF-4.4 | Eliminar comercio | Una vez el comercio quiera eliminar su cuenta de la plataforma, el administrador debe comprobar que la cuenta ha sido eliminada junto los todos los datos identificativos asociada a ella. |
| RF-5 | Actualizar perfil | El administrador podrá actualizar los datos de su perfil una vez inicie sesión en la plataforma. |
| RF-6 | Visualizar perfil | El administrador podrá visualizar los datos de su perfil en la plataforma. |
| RF-7 | Cerrar sesión | El administrador podrá cerrar sesión, siendo redirigido a la pantalla de inicio de la plataforma. |
| RF-8 | Visualizar producto | El administrador puede consultar el producto para obtener más información acerca de él una vez ha accedido al comercio y tecleado el nombre. |
| RF-9 | Buscar producto | El administrador podrá buscar el producto que desee, tecleando el nombre de este en el filtro de búsqueda. Cabe destacar que la búsqueda de productos será a nivel de comercio, dejando la búsqueda global como requisito a futuro. |
| RF-10 | Buscar comercio | El administrador podrá buscar el comercio que desee para facilitar la navegación al mismo, tecleando su nombre en un filtro de búsqueda. |

***Figura 1.*** *Requisitos del administrador*

*Nota: El administrador será creado por el desarrollador de la plataforma, introduciendo sus datos en la base de datos del servidor. Por lo tanto, los demás administradores no tendrán el privilegio de crear más administradores, ni un usuario anónimo se puede registrar como administrador en la plataforma una vez se realice.*

*Si algún administrador desea realizar compras, se debe crear una cuenta cliente.*

*A efectos de comprobación de funcionamiento, en cuanto a la notificación de la resolución de la solicitud, se ha creado una cuenta de correo electrónico de prueba común para todos los comercios para comprobar que el mensaje es enviado con éxito.*

2.3.2 Requisitos del cliente

|  |  |  |
| --- | --- | --- |
| Identificador | Nombre | Descripción |
| RF-11 | Registro de usuario | Para realizar compras en la plataforma,el cliente deberá registrarse en la plataforma. Los datos que debe introducir son el nombre de usuario, correo electrónico, contraseña, fecha de nacimiento, municipio y dirección. Así mismo, la contraseña del usuario será encriptada utilizando la librería *BCrypt [1]* que proporciona Java, añadiendo una ‘sal’ para reforzar aún más la contraseña a la hora de almacenarla en la base de datos. |
| RF-12 | Acceso a la plataforma | Todo usuario previamente registrado deberá iniciar sesión en la aplicación identificándose con un nombre de usuario y una contraseña. |
| RF-13 | Visualizar comercio | El cliente podrá visualizar todos los comercios para acceder al catálogo de productos de cada uno. |
| RF-14 | Buscar comercio | El cliente podrá buscar el comercio que desee para facilitar la navegación al mismo, tecleando su nombre en un filtro de búsqueda. |
| RF-15 | Visualizar perfil | El cliente puede visualizar los datos de su perfil. |
| RF-16 | Visualizar producto | El cliente puede consultar el producto para obtener más información acerca de él. Se debe haber tecleado el nombre de este anteriormente para poder acceder. |
| RF-17 | Buscar producto | El cliente podrá buscar el producto que desee, tecleando el nombre de este en el filtro de búsqueda. Cabe destacar que la búsqueda de productos será a nivel de comercio, dejando la búsqueda global como requisito a futuro. |
| RF-18 | Actualizar perfil | El cliente podrá visualizar, así como actualizar los datos de su perfil una vez inicien sesión en la plataforma. |
| RF-19 | Eliminar perfil | El cliente podrá darse de baja de la plataforma cuando desee, perdiendo el acceso a la misma y eliminando así todos sus datos identificativos de la base de datos del servidor. |
| RF-20 | Añadir productos al carrito | El cliente podrá almacenar los productos que desee comprar en un carrito de compra para su posterior “compra”. |
| RF-21 | Quitar productos del carrito | El cliente podrá eliminar aquellos productos que desee del carrito antes de confirmar la compra. |
| RF-22 | Confirmar compra | El cliente podrá confirmar la compra, mostrando simplemente un mensaje de que la compra ha sido realizada, ya que en nuestro caso no habrá transacciones de pagos. |
| RF-23 | Cerrar sesión | El usuario podrá cerrar sesión, saliendo a la pantalla de inicio de la plataforma. |

*Figura 2. Requisitos del cliente.*

2.3.3 Requisitos del comercio

|  |  |  |
| --- | --- | --- |
| Identificador | Nombre | Descripción |
| RF-24 | Solicitud de acceso a la plataforma | El comercio deberá rellenar una solicitud con sus datos para acceder a la plataforma, que deberá ser aprobado por un administrador. Deberá aportar el nombre del comercio, NIF, municipio, provincia, código postal, país, dirección, teléfono de contacto y email. Será notificado a un correo de pruebas creado una vez se resuelva la solicitud. |
| RF-25 | Acceso a la plataforma | Todo comercio, con su solicitud de acceso aprobada por el administrador, deberá iniciar sesión en la aplicación identificándose con el nombre de usuario y la contraseña que el administrador le proporcionará previamente. |
| RF-26 | CRUD productos | El comercio puede crear, visualizar, actualizar o borrar el producto que desee de su catálogo. |
| RF-26.1 | Creación de producto | El comercio tendrá la capacidad de crear los productos que desee. Para ello, deberá introducir el título de este, un identificador, un precio de venta, una descripción detallando el producto y una imagen. Cabe destacar que el almacenamiento y carga de dichas imágenes se harán en un paquete del proyecto. |
| RF-26.1 | Visualización de producto | El comercio podrá visualizar sus productos una vez estén creados y añadidos a la plataforma. |
| RF-26.3 | Actualizar producto | El comercio puede actualizar los datos introducidos en la creación siempre que lo desee. Dispondrá de un botón de edición que dejará cambiar cualquier dato identificativo del mismo salvo el identificador. Deben guardarse los datos para que la actualización se realice correctamente. |
| RF-26.4 | Eliminar producto | El comercio dispondrá de un botón para eliminar el producto que desee. Una vez se confirme el borrado, los datos asociados a ese producto quedarán eliminados del sistema. |
| RF-27 | Actualizar perfil | El comercio podrá actualizar los datos de su perfil una vez inicie sesión en la plataforma. |
| RF-28 | Visualizar perfil | El comercio podrá visualizar su perfil y los datos que lo identifican. |
| RF-29 | Buscar producto | El comercio podrá buscar el producto que desee en su comercio tecleando el nombre de este en el filtro de búsqueda. Igualmente, la búsqueda es a nivel de comercio. |
| RF-30 | Eliminar perfil | El comercio podrá darse de baja de la plataforma cuando desee, perdiendo el acceso a la misma y eliminando así todos sus datos identificativos de la base de datos del servidor. |
| RF-31 | Cerrar sesión | El usuario podrá cerrar sesión, saliendo a la pantalla de inicio de la plataforma. |

*Figura 3. Requisitos del comercio*

2.3.4 Requisitos usuario anónimo

|  |  |  |
| --- | --- | --- |
| Identificador | Nombre | Descripción |
| RF-32 | Visualizar producto | El usuario anónimo puede consultar el producto para obtener más información acerca de él una vez teclee el nombre de este. |
| RF-33 | Visualizar comercio | El usuario anónimo podrá visualizar todos los comercios para acceder al catálogo de productos de cada uno. |
| RF-34 | Buscar producto | El usuario anónimo podrá buscar el producto que desee en un determinado comercio, tecleando el nombre de este en el filtro de búsqueda. La búsqueda también será a nivel de comercio. |
| RF-35 | Buscar comercio | El usuario anónimo podrá buscar el comercio que desee para facilitar la navegación al mismo, tecleando su nombre en un filtro de búsqueda. |
| RF-36 | Registro de usuario | Para realizar compras en la plataforma,deberá registrarse en la plataforma. Los datos que debe introducir son el nombre de usuario, correo electrónico, contraseña, y fecha de nacimiento. |

*Figura 4. Requisitos usuario anónimo*

2.4 Requisitos no funcionales

En este apartado se detallan las restricciones que se deben imponer en el sistema y a la vez se definen sus atributos de calidad. Estos requisitos son importantes porque garantizan que el sistema no solo cumpla con las funcionalidades requeridas por los distintos usuarios, sino que además conlleve una serie de atributos de cales tales como usabilidad, seguridad y robustez. Para una identificación más precisa de los requisitos, se denotan mediante las siglas RNF, seguidas de un número identificativo y ordenadas de manera ascendente.

|  |  |  |
| --- | --- | --- |
| Identificador | Nombre | Descripción |
| RNF-1 | Usabilidad | Se debe asegurar que el sistema sea fácil de usar y comprender, con interfaces gráficas lo más intuitivas posibles. |
| RNF-1.1 | Test de uso de la plataforma | Se les otorgará a 10 personas aleatorias de un rango de edad considerable la aplicación una vez esté completada, y probar la aplicación para así recoger comentarios acerca de la interfaz y cosas a mejorar de la misma. Por ello, se considerará el requisito validado si al menos 8 de las 10 personas aprueban el modo de uso de la plataforma. |
| RNF-2 | Manual de usuario | Se elaborará un manual de usuario básico para el aprendizaje del manejo de la aplicación |
| RNF-3 | Seguridad | El sistema se debe proteger ante cualquier intento de acceso no autorizado. |
| RNF-3.1 | Contraseñas alfanuméricas | Por ello, **l**as contraseñas que se deben establecer deben tener mínimo 8 dígitos compuestas por letras, números, pudiendo contener también símbolos especiales de la siguiente lista (!,@, #, $, %, &, \*, -) |
| RNF-4 | Robustez | Se debe pasar satisfactoriamente una batería de pruebas previas al despliegue de este. El sistema deberá estar en la medida de lo posible libre de fallos, para lo cual tendrá que pasar una batería de pruebas previamente definida y que deberá satisfacerse cada vez que se haga una modificación en la plataforma. |
| RNF-5 | PC como plataforma destino | La aplicación se desarrollará en primera instancia para PC con sistema operativo Windows. |

2.5 Requisitos a futuro

En este apartado se detallan algunos requisitos a futuro que se podrían implementar en caso de que la aplicación pasara a producción para dotarla de funcionalidad adicional. Al igual que los requisitos funcionales y no funcionales, estos se denotan mediante las siglas RFU junto con un número identificativo que las enumerará.

|  |  |  |
| --- | --- | --- |
| Identificador | Nombre | Descripción |
| RFU-1 | Pasarela de pago | Se implementaría el uso de una pasarela de pago una vez el cliente quiera confirmar la compra, donde se establecería el acuerdo con una entidad bancaria para que el cliente sea redirigido a la pasarela de pago de la entidad, introduciendo sus datos bancarios. Una vez la compra sea realizada, se redirigirá al cliente a la aplicación. |
| RFU-2 | Plataformas futuras de desarrollo | Una vez la aplicación se desarrolle para PC con sistema operativo Windows, se podría ampliar para otros sistemas que se utilizan con alta frecuencia hoy en día (MacOS, iOS, Android). |
| RFU-3 | Comparador de productos | La idea para implementar sería hacer un metabuscador/comparador de productos, de forma que se pueda buscar el producto, sea del comercio que sea, y se muestre el más barato de todos. Cabe destacar que no se va a implementar no por el coste tecnológico, sino por las altas connotaciones empresariales que conlleva. |
| RFU-4 | Funcionalidad de cobro al comercio para rentabilizar la aplicación | La política de cobro está pendiente de definir. |
| RFU-5 | Lista de países del mundo junto sus respectivas ciudades | Se implementaría una lista de países a la hora de registrarse un usuario cliente o un usuario comercio para evitar fallos a la hora de escribir y, donde una vez el país sea elegido, se mostrarán las ciudades del país que previamente se ha elegido. De esta forma se pretende mejorar la eficiencia del proceso de registro del usuario. |

*Figura 5. Requisitos a futuro del proyecto*

3

Diagramas del sistema

3.1 Contexto y objetivo

En este capítulo, se pretende detallar los diagramas que se han realizado a partir de los requisitos extraídos ya que forman parte del proceso de ingeniería para la producción de software. Realizar todos los diagramas posibles para el análisis y diseño de un software es una tarea excesiva para una sola persona, por lo que el trabajo propuesto se ha centrado en tres tipos de diagrama, los cuales son suficientes para describir tanto la funcionalidad como la estructura de la aplicación: diagrama de clases, diagrama de casos de uso y diagrama de secuencia, los cuales serán abordados en sus respectivos capítulos.

3.2 Casos de uso

En este apartado, se procede a describir los casos de uso del sistema. Los casos de uso sirven para describir las interacciones que realizan los distintos actores con el sistema bajo desarrollo.

Tal y como se realizó en el apartado de los requisitos, se muestran los casos de uso en base a los actores que participan en el sistema, por lo que aparecerán algunas repeticiones en algunos de ellos, especificando para cada caso de uso su escenario principal, precondiciones, postcondiciones y un escenario alternativo en los casos que se considere apropiado por la relevancia/complejidad del caso de uso. Consideramos siempre como escenario principal de un caso de uso aquel que tiene lugar en la mayoría de las ocasiones que ocurre dicho caso de uso. También cabe la posibilidad de que algún caso de uso no precise de precondiciones o de escenario alternativo.

3.2.1 Administrador

Como se puede observar en la *Figura 6*, se han insertado las relaciones de inclusión donde se ha requerido. Esto quiere decir que cuando relacionamos dos casos de uso mediante esta relación, el caso de uso base que es el primero, incluye al segundo (siempre que se da el primero, también se da el segundo), pues no podría realizar su función y por tanto no se podrían cumplir sus objetivos. A continuación, se adjunta una imagen para visualizar los casos de uso que se han obtenido a partir de los requisitos funcionales redactados con su correspondiente tabla en la *Figura 6*:

Diagrama

Descripción generada automáticamente

*Figura 6. Casos de uso del actor administrador.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Precondiciones | Escenario principal | Postcondiciones | Escenario alternativo |
| Eliminar administrador | El administrador debe estar registrado en la plataforma. | 1. Un administrador pulsa el botón de borrado en la base de datos en la fila que contenga el administrador que quiera eliminar.  2. El sistema elimina toda información de administrador. | El administrador y todos sus datos quedan eliminados del sistema. | No consta. |
| Consultar administrador | El administrador debe estar en el sistema. | 1. El usuario teclea el nombre del administrador.  2. El sistema comprueba los datos introducidos.  3. La base de datos del servidor muestra los datos del administrador que coinciden con el nombre introducido. | Devuelve los datos del administrador tecleado. | 2.a. Error: El nombre no coincide con ningún administrador almacenado, vuelve al paso 1. |
| Actualizar administrador | El administrador debe estar registrado en el sistema. | 1. El usuario pulsa el botón de edición que hay en su perfil de la plataforma.  2. El sistema permite al usuario editar datos.  3. El usuario cambia los datos que crea oportunos.  4. El sistema comprueba que son correctos.  5. Los datos son actualizados correctamente. | Los datos del administrador quedan actualizados correctamente. | 4.a. Error, la contraseña introducida no cumple con la longitud que el sistema acepta. Vuelve al paso 1. |
| Cerrar sesión | El usuario debe haber iniciado sesión en el sistema. | 1. El sistema ofrece en todo momento un botón para cerrar sesión.  2. El usuario pulsa el botón.  3. El sistema limpia toda la información de sesión del usuario.  4. El usuario es redirigido a la pantalla principal satisfactoriamente, pasando de registrado a anónimo. | El usuario pasa de registrado a anónimo. | No consta. |
| Acceso a la plataforma | El usuario debe estar registrado en la plataforma. | 1. El sistema muestra un formulario para introducir un nombre de usuario y una contraseña.  2. El usuario rellena el formulario.  3. El usuario pulsa el botón de iniciar sesión.  4. El sistema valida los datos.  5. El usuario inicia sesión en la plataforma con éxito. | El usuario accede a la plataforma como usuario identificado. | 4.a. El sistema detecta que los datos introducidos son incorrectos, muestra un mensaje de error y vuelve al paso 1. |
| Visualizar perfil | Debe tener un usuario creado en la plataforma y haber iniciado sesión. | 1. El usuario hace clic en su nombre de perfil.  2. El sistema muestra el perfil con sus datos. | El usuario obtiene los datos pertenecientes a su perfil. | No consta. |
| Actualizar perfil | Debe acceder a la plataforma y acceder a los datos de su perfil. | 1. El usuario accede a su perfil  2. El usuario pulsa el botón de editar.  3. El usuario edita algunos datos y confirma los cambios.  4. El sistema comprueba si son correctos.  5. El perfil es actualizado satisfactoriamente. | El perfil queda actualizado con los nuevos datos introducidos. | 4.a. Error: la contraseña establecida es incorrecta. El sistema devuelve un mensaje de error y vuelve al paso 3. |
| Visualizar producto | Debe haber accedido al comercio, visualizarlo y posteriormente haber tecleado el nombre del producto. | 1. El usuario hace clic en el producto que desea visualizar.  2. El sistema muestra el producto junto a todos sus datos. | El usuario visualiza el producto junto a sus detalles. | No consta. |
| Visualizar comercio | Debe haber tecleado el nombre del comercio y accedido al mismo. | 1. El usuario pulsa en el nombre del comercio.  2. El sistema devuelve el catálogo de productos satisfactoriamente. | Obtiene el comercio que se ha tecleado. | 1.a. Error: No se ha encontrado el comercio tecleado, vuelve al paso 1. |
| Buscar producto | Debe haber accedido al comercio tecleado. | 1. El usuario teclea el nombre de un producto  2. La petición es enviada al sistema, que busca el producto que coincida con el nombre proporcionado.  3. El sistema devuelve el producto buscado. | Devuelve el producto tecleado por el usuario. | 2.a. Error: no se ha encontrado el producto buscado, vuelve al paso 1. |
| Buscar comercio | No existen precondiciones. | 1. El usuario introduce el nombre del comercio en la barra de búsqueda.  2. El sistema verifica si el nombre introducido es correcto y se encuentra en el mismo.  3. El sistema devuelve el comercio que coincide con el nombre tecleado satisfactoriamente. | Se muestra el comercio tecleado correctamente. | 2.a. Muestra un mensaje de error informando de que el comercio no existe, vuelve al paso 1. |
| Aceptar solicitud | La solicitud debe haber sido rellenada y enviada por el comercio | 1. El administrador comprueba los datos introducidos en la solicitud.  2. Se verifican todos los datos comprobando que son correctos, aprobando la solicitud. | La solicitud es aprobada por el administrador, dándole acceso a la plataforma al comercio con un usuario y una contraseña que se le proporcionará por correo. | No consta. |
| Crear comercio | La solicitud del comercio debe haber sido aceptada. | 1. El administrador introduce los datos del comercio que provienen de la solicitud.  2. El sistema verifica los datos.  3. Los datos son introducidos satisfactoriamente en la base de datos del servidor. | El comercio es creado y está listo para su uso. | No consta. |
| Eliminar comercio | El comercio debe estar registrado en la base de datos. | 1. El administrador da la orden de eliminación de la tupla correspondiente al comercio en la base de datos.  2. El sistema borra la tupla correspondiente con éxito. | El comercio es eliminado del sistema, perdiendo el acceso a la aplicación. | No consta. |
| Actualizar comercio | El comercio debe estar registro en la base de datos. | 1. El administrador modifica la tupla del comercio correspondiente.  2. El comercio es actualizado con los nuevos datos. | El comercio es actualizado con los nuevos datos introducidos. | No consta. |
| Rechazar solicitud | La solicitud debe haber sido realizada por el comercio y comprobada por el administrador. | 1. Se verifican los datos de la solicitud.  2. Se comprueba que hay datos erróneos, por lo que rechaza la solicitud. | La solicitud es rechazada por el administrador. | No consta. |
| Comprobar solicitud | La solicitud debe haber sido rellenada y enviada por el comercio | 1. El usuario comprueba los datos enviados del comercio por la solicitud.  2. El usuario verifica que son correctos y aprueba la solicitud. | La solicitud es resuelta por el administrador. | No consta. |

*Figura 7. Casos de uso del administrador.*

3.2.2 Comercio

Se procede a detallar los casos de uso del usuario comercio, representando las relaciones que establecen los distintos casos de uso junto a las relaciones de inclusión que se establecen si así procede. Como se hizo con el administrador, se detallarán al completo los casos de uso que se han realizado a partir de los requisitos funcionales de este usuario. Para apoyar la explicación de estos casos de uso, se proporciona la *Figura 8* (realizada con la herramienta Visual Paradigm). La *Figura 8* detalla cada uno de los casos de uso con los que interactúa el actor comercio*.*

**

*Figura 8. Visualización de casos de uso del usuario comercio.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Precondiciones | Escenario principal | Postcondiciones | Escenario alternativo |
| Actualizar producto | El producto debe haberse creado. | 1. El comercio hace clic en el nombre del producto.  2. El comercio pulsa el botón de edición.  3. El comercio cambia los datos que deseen ser editados.  4. El comercio guarda los cambios.  5. El sistema valida si los datos son correctos.  6. Los datos son actualizados con éxito. | El producto es actualizado con los nuevos datos. | 5.a. Error, datos incorrectos introducidos, vuelve al paso 1. |
| Crear producto | Ser un usuario comercio. | 1. El comercio pulsa el botón de crear producto.  2. Introduce los datos correspondientes al producto.  3. El sistema valida los datos.  4. Los datos son correctos y el producto es creado satisfactoriamente. | El producto es creado con los datos introducidos por el comercio. | 3.a. Error, el formato del nombre no es válido. Vuelve al paso 2. |
| Eliminar producto | Ser un usuario comercio. | 1. El comercio accede al producto pulsando en su nombre.  2. El comercio hace clic en el botón para eliminar.  3. El producto es eliminado tanto de la plataforma como de la base de datos del servidor. | El producto es eliminado del sistema junto con sus datos. | No consta. |
| Visualizar producto | El producto debe haber sido creado anteriormente. | 1. Debe haber accedido al comercio, visualizarlo y posteriormente haber tecleado el nombre del producto.  2. El sistema muestra el producto junto a todos sus datos. | El usuario visualiza el producto junto a sus detalles. | No consta. |
| Buscar producto | El producto debe haber sido creado anteriormente. | 1. El usuario teclea el nombre de un producto.  2. La petición es enviada al sistema, que busca el producto que coincida con el nombre proporcionado.  3. El sistema devuelve el producto buscado. | Devuelve el producto tecleado por el usuario. | 2.a. Error: no se ha encontrado el producto buscado, vuelve al paso 1. |
| Actualizar perfil | Debe haber accedido a la plataforma y visualizado su perfil. | 1. El usuario accede a su perfil.  2. El usuario utiliza el botón de editar.  3. El usuario edita algunos datos y confirma los cambios.  4. El sistema comprueba si son correctos.  5. El perfil es actualizado satisfactoriamente. | El perfil queda actualizado con los nuevos datos introducidos. | 4.a. Error: la contraseña establecida es incorrecta, vuelve al paso 3. |
| Visualizar perfil | Debe tener un usuario creado en la plataforma y haber iniciado sesión. | 1. El usuario hace clic en su nombre de perfil.  2. El sistema muestra el perfil con sus datos. | El usuario obtiene los datos pertenecientes a su perfil. | No consta. |
| Eliminar perfil | Debe tener creado un usuario en la plataforma | 1. El usuario pulsa en el botón de eliminar perfil.  2. El sistema borra sus datos de la base de datos del servidor perdiendo así el acceso a la plataforma. | El usuario es eliminado de la plataforma. | No consta. |
| Cerrar sesión | El usuario debe haber iniciado sesión en el sistema. | 1. El sistema ofrece en todo momento un botón para cerrar sesión.  2. El usuario pulsa el botón.  3. El sistema limpia toda la información de sesión del usuario.  4. El usuario es redirigido a la pantalla principal satisfactoriamente, pasando de registrado a anónimo. | El usuario pasa de registrado a anónimo. | No consta. |
| Acceso a la plataforma | El usuario debe estar registrado en la plataforma. | 1. El sistema muestra un formulario para introducir un nombre de usuario y una contraseña.  2. El usuario rellena el formulario.  3. El usuario pulsa el botón de iniciar sesión.  4. El sistema valida los datos.  5. El usuario inicia sesión en la plataforma con éxito. | El usuario accede a la plataforma como usuario identificado. | 4.a. El sistema detecta que los datos introducidos son incorrectos, muestra un mensaje de error y vuelve al paso 1. |
| Solicitud acceso | Ser un comercio. | 1. El comercio pulsa el botón de solicitud de acceso.  2. Rellena los datos del formulario.  3. El sistema valida si los datos son correctos.  4. El formulario es enviado. | La solicitud de acceso es enviada y lista para ser revisada por un administrador. | 3.a. Error: algunos datos son erróneos, vuelve al paso 2. |

Figura 9. Casos de uso del comercio.

3.2.3 Cliente

Se procede a detallar los casos de uso del usuario cliente. Al igual que los demás usuarios, se detallan los casos de uso en los que el cliente participa, así como las figuras correspondientes para visualizar y describir el funcionamiento de estos en el sistema que se pretende desarrollar.



*Figura 10. Visualización de casos de uso del cliente*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Precondiciones | Escenario principal | Postcondiciones | Escenario alternativo |
| Cerrar sesión | El usuario debe haber iniciado sesión en el sistema. | 1. El sistema ofrece en todo momento un botón para cerrar sesión.  2. El usuario pulsa el botón.  3. El sistema limpia toda la información de sesión del usuario.  4. El usuario es redirigido a la pantalla principal satisfactoriamente, pasando de registrado a anónimo. | El usuario pasa de cliente a anónimo. | No consta. |
| Acceso a la plataforma | El usuario debe estar registrado en la plataforma. | 1. El sistema muestra un formulario para introducir un nombre de usuario y una contraseña.  2. El usuario rellena el formulario.  3. El usuario pulsa el botón de iniciar sesión.  4. El sistema valida los datos.  5. El usuario inicia sesión en la plataforma con éxito. | El usuario accede a la plataforma como usuario identificado. | 4.a. El sistema detecta que los datos introducidos son incorrectos y vuelve al paso 1. |
| Visualizar comercio | El usuario debe haber buscado el comercio previamente. | 1. El usuario pulsa en el nombre del comercio.  2. El sistema devuelve el catálogo de productos satisfactoriamente. | El usuario obtiene el comercio cuyo nombre ha tecleado, junto a sus productos. | 1.a. Error: No se ha encontrado el comercio tecleado, vuelve al paso 1. |
| Visualizar producto | El producto deber haber sido creado. | 1. Debe haber accedido al comercio, visualizarlo y posteriormente haber tecleado el nombre del producto.  2. El sistema muestra el producto junto a todos sus datos. | El usuario visualiza el producto junto a sus detalles. | No consta. |
| Buscar producto | El producto debe haber sido creado anteriormente. | 1. El usuario teclea el nombre de un producto.  2. La petición es enviada al sistema, que busca el producto que coincida con el nombre proporcionado.  3. El sistema devuelve el producto buscado | Devuelve el producto tecleado por el usuario. | 2.a. Error: no se ha encontrado el producto buscado, vuelve al paso 1. |
| Buscar comercio | El comercio debe existir en la base de datos de la plataforma. | 1. El usuario introduce el nombre del comercio en la barra de búsqueda.  2. El sistema verifica si el nombre introducido es correcto y se encuentra en el mismo.  3. El sistema devuelve el comercio que coincide con el nombre tecleado satisfactoriamente. | Se muestra el comercio tecleado correctamente. | 2.a. Muestra un mensaje de error informando de que el comercio no existe, vuelve al paso 1. |
| Visualizar perfil | Debe tener un usuario creado en la plataforma y haber iniciado sesión. | 1. El usuario hace clic en su nombre de perfil.  2. El sistema muestra el perfil junto a los datos que lo identifican. | El usuario obtiene los datos pertenecientes a su perfil. | No consta. |
| Actualizar perfil | El usuario debe haber iniciado sesión y haber visualizado su perfil. | 1. El usuario accede a su perfil.  2. El usuario utiliza el botón de editar.  3. El usuario edita algunos datos y confirma los cambios.  4. El sistema comprueba si son correctos.  5. El perfil es actualizado satisfactoriamente. | El perfil queda actualizado con los nuevos datos introducidos. | 4.a. Error: la contraseña establecida es incorrecta, devuelve un mensaje de voz y vuelve al paso 3. |
| Eliminar perfil | Debe tener creado un usuario en la plataforma. | 1. El usuario pulsa en el botón de eliminar perfil.  2. El sistema borra sus datos de la base de datos del servidor perdiendo así el acceso a la plataforma. | El usuario es eliminado de la plataforma. | No consta. |
| Confirmar compra | El usuario debe haber añadido al menos un producto al carrito. | 1. El usuario accede al carrito.  2. El usuario pulsa el botón de compra.  3. El sistema muestra un mensaje confirmando la compra y vaciando el carrito. | El sistema muestra un mensaje de confirmación de la compra una vez se pulse el botón. | No consta. |
| Almacenar en carrito | Debe de haber iniciado sesión como usuario cliente. | 1. El usuario pulsa el botón de 'Añadir al carrito' del producto.  2. El producto es añadido al carrito satisfactoriamente, incrementando el número de productos en el mismo. | Los productos son almacenados en el carrito listos para confirmar la compra. | No consta. |
| Quitar productos del carrito | Debe de haber mínimo un producto añadido al carrito. | 1. El usuario accede al carrito.  2. El usuario pulsa el botón de eliminar producto del carrito.  3. El producto es eliminado y el carrito se actualiza correctamente. | El producto es eliminado del carrito. | No consta. |

*Figura 11. Casos de uso del cliente*

3.2.4 Anónimo

Finalmente, se describen los casos de uso relacionados con el usuario anónimo, en el cual se puede observar un diagrama más reducido debido a las funciones que este usuario posee en la plataforma, centrándose principalmente en la búsqueda y visualización de los productos y comercios de la plataforma. Así pues, se procede a describir detalladamente las acciones que puede llevar a cabo el usuario anónimo en la plataforma en las *figuras 12 y 13* respectivamente*.*



*Figura 12. Visualización de los casos de uso del usuario anónimo.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Precondiciones | Escenario principal | Postcondiciones | Escenario alternativo |
| Buscar comercio | El comercio debe estar registrado en la plataforma. | 1. El usuario introduce el nombre del comercio en la barra de búsqueda.  2. El sistema verifica si el nombre introducido es correcto y se encuentra en el mismo.  3. El sistema devuelve el comercio que coincide con el nombre tecleado satisfactoriamente. | Se muestra el comercio tecleado. | 2.a. Muestra un mensaje de error informando de que el comercio no existe, vuelve al paso 1. |
| Registro de usuario | Ser usuario anónimo y no tener usuario en la plataforma. | 1. El usuario rellena el formulario de registro y pulsa en confirmar.  2. El sistema comprueba que no existen colisiones con los datos de los usuarios ya existentes.  3. El sistema muestra un mensaje de éxito al registrarse.  4. El sistema almacena los datos en la base de datos del servidor. | El usuario queda registrado correctamente en la plataforma. | 2.a. Error: Ya existe un usuario con el mismo nombre, vuelve al paso 1. |
| Buscar producto | El producto debe haber sido creado anteriormente. | 1. El usuario teclea el nombre de un producto.  2. La petición es enviada al sistema, que busca el producto que coincida con el nombre proporcionado.  3. El sistema devuelve el producto buscado. | Devuelve el producto tecleado por el usuario. | 2.a. Error: no se ha encontrado el producto buscado, vuelve al paso 1. |
| Visualizar comercio | El comercio debe existir, y debe haber realizado la búsqueda de este. | 1. El usuario pulsa en el nombre del comercio.  2. El sistema devuelve el catálogo de productos satisfactoriamente. | Devuelve los datos de los productos pertenecientes a dicho comercio. | No consta. |
| Visualizar producto | Que el producto haya sido creado. | 1. Debe haber accedido al comercio, visualizarlo y posteriormente haber tecleado el nombre del producto.  2. El sistema muestra el producto junto a todos sus datos. | El usuario visualiza el producto junto a sus detalles. | No consta. |

*Figura 13. Casos de uso del usuario anónimo.*

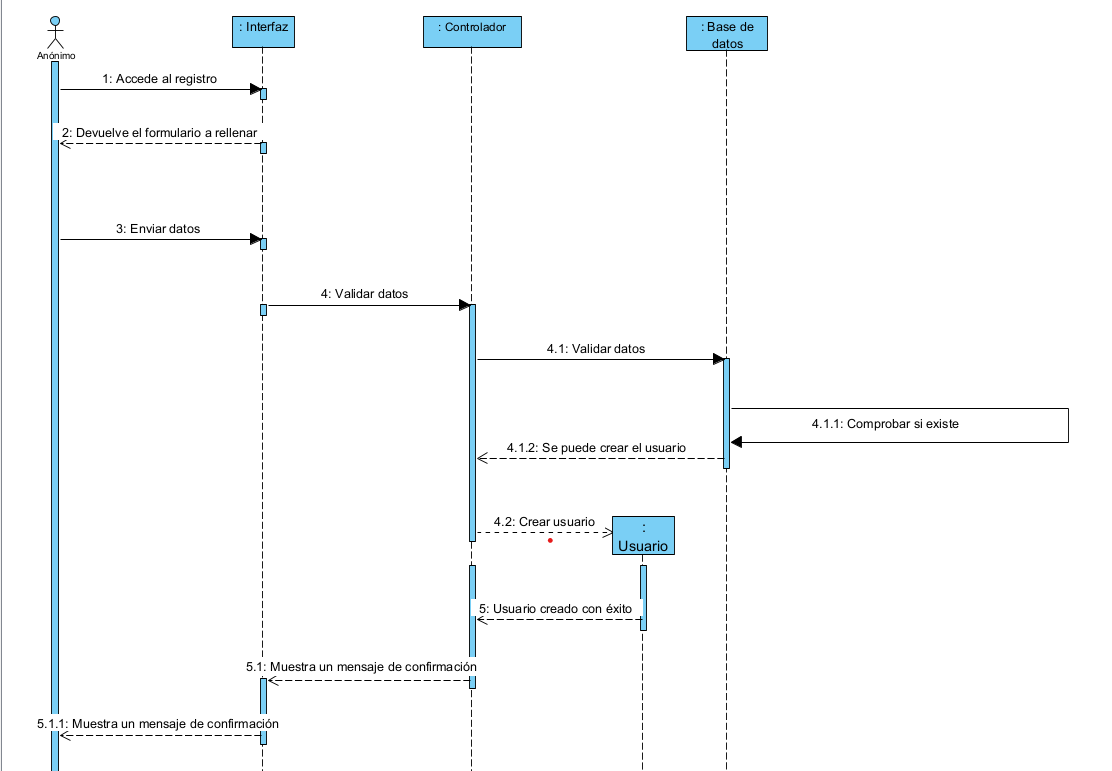
3.3 Diagramas de secuencia

En este apartado se abordan los diagramas de secuencia realizados de la plataforma desarrollada. Inicialmente se planteó la posibilidad de elaborar un diagrama de secuencia por cada escenario de caso de uso, pero debido a la gran cantidad de casos de uso que se han elaborado, el coste de realización de todos estos diagramas no resulta viable, aún más teniendo en cuenta que muchos de ellos son bastante claros de por sí, junto con la especificación de requisitos. Por consiguiente, se ha centrado el esfuerzo en los que, a nuestro criterio, son los cinco casos de uso más relevantes del sistema bajo desarrollo, proporcionando un diagrama de secuencia para el escenario principal de cada uno de ellos. Con estos diagramas se pretenden mostrar el comportamiento general del caso de uso, con el objetivo de determinar los objetos (participantes) necesarios para la implementación de este, junto con las interacciones entre los distintos participantes.

Los objetos que intervienen son representados mediante rectángulos en la parte superior de sendas líneas discontinuas verticales, las cuales representan la dimensión temporal. Los mensajes pasados entre los objetos se representan con flechas horizontales, diferenciando entre dos tipos: los mensajes síncronos, que bloquean al objeto emisor hasta que reciba una respuesta se representan con la punta de flecha rellena, y los mensajes asíncronos, que terminan una vez se envía el mensaje y se representan con una cabeza de flecha hueca. Cabe destacar que la respuesta al mensaje síncrono es representada mediante una línea discontinua hacia el emisor.

Por último, en caso de que el diagrama lo necesite, se detallarán algunos aspectos acerca del mismo de manera sucinta, pues la figura correspondiente a cada uno de los diagramas muestra los pasos que se deben seguir para realizar el caso de uso y el comportamiento de este.

3.3.1 Registrar usuario



*Figura 14. Diagrama de secuencia del registro de usuario.*

3.3.2 Solicitud acceso a la plataforma

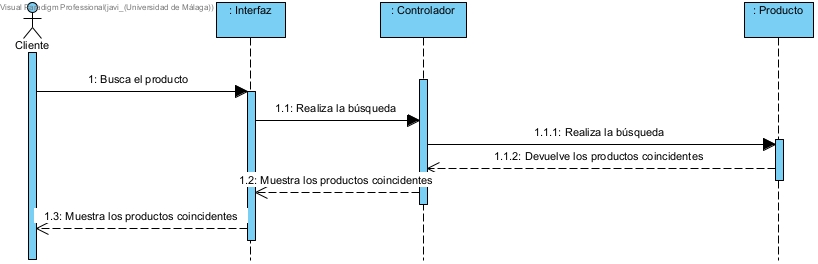
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 15. Diagrama de secuencia de la solicitud de acceso a la plataforma por parte del actor comercio.*

En la *Figura 15* solamente se tiene en cuenta el caso en el que el administrador acepta la solicitud del comercio.

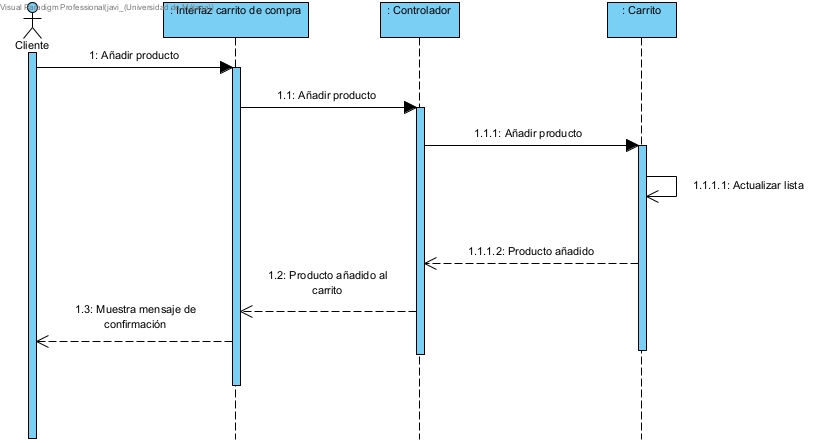
3.3.3 Buscar producto



*Figura 16. Diagrama de secuencia de la búsqueda de un producto.*

En el diagrama de la *Figura 16* se muestra el caso en el que el producto tecleado es encontrado.

3.3.3 Añadir producto al carrito



*Figura 17. Diagrama de secuencia para añadir un producto al carrito.*

3.3.4 Creación de comercio

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

*Figura 18. Diagrama de secuencia de la creación del comercio.*

Cabe destacar que este diagrama se lleva a cabo si el administrador aprueba la solicitud enviada por el comercio.

3.4 Diagrama de clases

En este apartado se procede a realizar el diagrama de clases del proyecto, ayudándonos para ello en los diagramas UML ya realizados mostrados en las secciones anteriores del documento. Con este diagrama se pretende ilustrar la estructura del sistema en base a sus clases, atributos y métodos, ofreciendo una mejor compresión visual del modelo lógico de los datos del sistema.

Es importante destacar que en este diagrama no se incluye la forma en la cual los datos son tratados y procesados a lo largo de la ejecución ya que esa función se representa en los diagramas de secuencia y en los diagramas de casos de uso, ni tampoco las funciones getter y setter de los atributos, pues aumentaría la complejidad visual de los diagramas. Por ello, se procede a la explicación del diagrama de clases lógico junto a sus métodos presentándose la *Figura 18* como apoyo a esta explicación.

3.4.1 Diagrama de clases lógico

Escala de tiempo

Descripción generada automáticamente

*Figura 19. Diagrama de clases del sistema.*

CarritoController: Clase donde se muestran los datos pertenecientes al carrito del cliente. El método 'initialize' es un método que se utiliza para inicializar una vista FXML, donde en este caso carga la información relativa a los productos que ha añadido el cliente previamente a su carrito, el precio total de este y guarda esa información en la variable 'productos', donde al hacer clic en un producto cualquiera, se muestran dos botones para actualizar la cantidad y eliminar el producto del carrito que se almacenan en la variable 'productosOptions'. En el método 'showTotalPrice' se calcula el precio total de los productos añadidos al carrito, donde se establece una conexión a la base de datos para obtener la información necesaria para obtener los datos del carrito, mostrando el resultado en la etiqueta 'totalPrecio'. El método 'confirmPurchase' establece una conexión con la base de datos donde se 'confirma' la compra del cliente. Ya que la pasarela de pago está excluida de los requisitos, este método borra el carrito del cliente de la base de datos, dejando vacío el carrito, mostrando un mensaje de confirmación de compra y a continuación se actualiza la ventana del carrito. Con el método 'goBack' se regresa a la anterior pantalla.

HomeController: Esta clase se utiliza para representar la pantalla principal de la aplicación. En esta clase, en el método 'initialize' se realiza un carrusel donde se muestran los comercios que hay registrados en la base de datos del sistema, donde se realiza una búsqueda en el método 'searchCommercesDB' en la base de datos almacenando en la variable 'nombrescomercios' para, una vez seleccionados todos los comercios, se almacena cada nombre en la variable carrusel y se ejecuta la animación gracias al método 'startAnimation'. En el método 'handleRegisterButton' se avanza a la interfaz de registro de usuario, donde en el método 'handleLoginButton' se realiza la misma función, pero con el inicio de sesión. El método 'searchCommerce' es utilizado para encontrar el comercio tecleado, cuyo nombre se guarda en la variable 'searchField', obteniendo todos los nombres de la base de datos y una vez recopilados, se comprueba nombre a nombre, guardando el posible nombre encontrado en la variable 'nombreBD' el cual, si coincide con el tecleado, la variable booleana 'commerceFound' pasa a ser true, se crea el botón del comercio para que se pueda acceder detalladamente a los productos de este. En caso de que no se encuentre el comercio se lanza un mensaje de error indicando que no se pudo encontrar el comercio tecleado. Por último, el método 'getidcommerce' obtiene el id del comercio recibiendo por parámetro el nombre del comercio.

LoginAdminController: En esta clase se han realizado los métodos necesarios para llevar a cabo el inicio de sesión del usuario administrador. En el método 'handleLogin' se lleva a cabo el inicio de sesión del usuario administrador, donde si hay algún error a la hora de iniciar sesión con las credenciales necesarias se muestra un mensaje indicándolo. El método 'handleLogout' cierra la sesión del usuario, volviendo a la pantalla de inicio de sesión. El método 'isValidCredentials', que sirve de apoyo al inicio de sesión, comprueba las credenciales de inicio de sesión introducidas y pasadas por parámetro, autorizando el acceso accediendo a la base de datos y comprobando si las credenciales introducidas coinciden con el registro correspondiente del sistema, en el que se lleva a cabo el hash de la contraseña del administrador introducida y la sal guardada en la base de datos para comprobar que efectivamente coincide. El hasheo de la contraseña en SHA-256[3] se lleva a cabo en el método 'hashPassword256'. El método 'handlegoBack' hace regresar al usuario a la anterior pantalla.

LoginComercioController: Esta clase se encarga del inicio de sesión del usuario comercio, pues los métodos que se muestran en la *Figura 18* son similares a los del administrador.

LoginController: En esta clase se realiza el inicio de sesión del cliente, que como se puede observar, es similar a los anteriores salvo que a la hora de hacer hash a la contraseña se utilizar la librería BCrypt[1] de Java.

LoginMenuController: En esta clase se implementa la funcionalidad del menú de login según la elección del usuario, por lo que se ha llevado a cabo la creación de tres variables booleanas que se establecerán a true dependiendo del usuario que sea. Si es administrador, se ejecutará el método 'handleAdminButton', que redirigirá a la pantalla de inicio de sesión del administrador, de la misma forma que los métodos 'handleNextScreen' que corresponde al cliente y 'handleCommerceButton' al usuario comercio. Ya que es fundamental que la navegación entre pantallas se realice de forma correcta, se crearon las variables booleanas pertenecientes a cada uno de los usuarios, pues se puede generar una confusión entre las pantallas de la aplicación y por consiguiente una navegación no deseada entre estas.

PaginaAdmin: Se ha realizado en esta clase los métodos realizados para la página principal del administrador. Se ha llevado a cabo la realización de un carrusel donde se muestran los nombres del comercio, donde el método 'searchCommercesDB' lleva a cabo la recopilación de los nombres de los comercios, almacenándolos en la variable 'nombrescomercios', que una vez recopilados se añade nombre por nombre a la variable 'carrusel', que una vez formado lleva a cabo la simulación de animación en el método 'startAnimation'. Por otro lado, encontramos el método 'handleLogout' para cerrar la sesión, el método 'handleGoAdminProfile' que es el encargado de mostrar el perfil del administrador y los datos de este. El método 'handleGoRequest' se encarga de visualizar las solicitudes emitidas por los comercios para su correspondiente revisión. En el método 'searchCommerce' se realiza una búsqueda en la base de datos del comercio tecleado a partir del nombre introducido, en el cual si se encuentra es almacenado en la variable. Si se encuentra, la variable 'commerceFound' se establece a true y se detiene la búsqueda añadiendo el resultado a la variable 'carruselComercios' y haciendo que los comercios encontrados puedan ser pulsados y se pueda mostrar su información perteneciente en pantalla. De la misma manera se lanzará un error si el comercio no es encontrado. Para obtener el id del comercio que se pretende buscar, se ha creado el método 'getidcommerce' encargado de buscar el comercio a partir del nombre introducido por teclado.

PaginaClienteController: En esta página se ha llevado a cabo la realización de los métodos de la página principal del cliente. Como se puede observar, se ha creado un método 'handleLogout' para cerrar sesión, 'handlegoProfile' para visualizar el perfil del cliente y los métodos 'searchCommercesDB', 'initialize' y 'startAnimation' para el carrusel de los comercios. También se ha realizado el método 'searchCommerce' que busca el comercio introducido por pantalla y que es posteriormente almacenado en la variable 'searchField' y 'getidcommerce' para encontrar el id del comercio buscado. Como novedad, se ha creado el método 'handleCart' para visualizar el carrito del cliente.

PaginaComercioController: Clase que se crea para manejar la página principal del comercio. Se encuentran métodos como 'handleLogout' para el cierre de sesión, 'handlegoProfile' para la visualización de su perfil y 'handleGoProducts' para la visualización de los productos publicados por el comercio, junto a los tres botones que disparan estos métodos.

PerfilAdminController: En esta clase se realizan las funciones que se pueden llevar a cabo en el perfil del usuario administrador. Primero, se lleva a cabo la inicialización y visualización de los datos del administrador mediante el método 'initialize' guardándolos en las etiquetas creadas por cada campo. El método 'handleEditButton' es disparado por el botón 'editDataAdmin', el cual crea un formulario con los campos que sirven para identificar a un administrador si se desea editar. Se cargan los datos iniciales que lo identifican y una vez editados, se pueden guardar o cancelar y volver a la pantalla anterior. Si es el primer caso, se envía una orden de actualización al registro correspondiente en la base de datos. Si los cambios se realizan exitosamente, se muestra un mensaje informando al usuario sobre ello. Para llevar a cabo el cambio de contraseña del administrador, se crea el método 'changePassword', que con la ayuda de los métodos 'hashPasswordSHA256' que hace un hash de la nueva contraseña establecida que se pasa por parámetro junto la sal que es llevada a cabo por el método 'cadenaAleatoria', restablece la contraseña escrita por teclado por el usuario, recogida en las variables 'passField' y 'repeatPassField', comprobándose que no ha habido error alguno, el cual muestra un mensaje de error en caso de no coincidir. Este último método hace uso del método 'numeroAleatorioEnRango' para crear una sal totalmente aleatoria. También se observa el método 'goBack' para regresar a la anterior pantalla.

PerfilClienteController: En esta clase se han realizado las funciones que se pueden llevar a cabo en el perfil del cliente. Inicialmente se encuentran métodos como 'handleLogout' para el cierre de sesión e 'initialize' para cargar los datos del usuario en su perfil. En 'handleDeleteProfile' se realiza el borrado del perfil del cliente, donde se hace una búsqueda de los datos de este en la base de datos para su correcta eliminación, mostrando la alerta correspondiente en caso de que sea o no llevado a cabo. En 'handleEditButton' se realiza la misma función que en el usuario administrador solo que los campos del cliente son distintos, pero el proceso es similar: se cargan los datos actuales del cliente en el formulario que se crea al disparar el botón 'editButton', en el que una vez editados los mismos, se pueden guardar, actualizando el registro correspondiente del cliente en la base de datos, o cancelando la edición, haciendo que el usuario regrese a la pantalla previa. El cambio de contraseña se realiza en el método 'changePassword' donde se crea un pequeño formulario con los campos 'passField' y 'repeatpassField', a diferencia de que en este método se usa la librería BCrypt [1] tanto para la generación de la sal como el hash de la contraseña, facilitando la elaboración de este método. También se hace uso del método 'goBack'.

PerfilComercioController: De la misma manera que los anteriores, se maneja la navegación dentro del perfil del comercio. Se puede observar que están presentes métodos ya conocidos y explicados previamente, tales como el método 'handleLogout' para el cierre de sesión, 'initialize' para obtener los datos de dicho usuario, 'handleDeleteProfile' para llevar a cabo la eliminación de los datos que identifican al comercio, la edición de los datos identificativos mediante el método 'handleEditButton' con la creación de un formulario junto a los campos que se requirieron en el registro salvo el NIF, 'goBack' para regresar a la pantalla anterior y 'changePassword' para el cambio de contraseña.

ProductosComercioController: En esta clase, se lleva a cabo el manejo de los productos que el comercio ha creado. Por tanto, en el método 'initialize' se cargan todos los productos pertenecientes a ese comercio, donde primero se obtiene y almacena el NIF del comercio, y seguidamente se almacenan todos los productos pertenecientes a ese comercio en una variable local creada. El encargado de la recopilación de los productos es el método 'getProductos', que obtiene los productos de dicho comercio, para que una vez recopilados, se cree un botón con el nombre de cada producto. Con el objetivo de que cada botón creado sea funcional, se ha creado el método 'mostrarDatos', mostrando los datos del producto que se ha consultado. En este método también puede llevarse a cabo la edición de los datos del producto, así como su eliminación de la plataforma y por lo tanto de la base de datos. El método 'handleAddProduct' crea un formulario con el que rellenar los datos pertenecientes a un producto, creando las variables necesarias para llevarlo a cabo, pudiendo guardar o cancelar la creación de este. El método 'searchProduct' devuelve el producto tecleado en el campo 'productField', el cual, si la variable found es falsa, se devuelve una alerta mostrando el error. También se encuentra el método 'goBack' para regresar a la pantalla anterior.

ProductosComercioDetalladoController: Esta clase se encarga de la visualización de los productos por parte del cliente una vez este ha accedido al comercio. El método 'initialize' de esta clase lleva a cabo la carga de todos los productos de dicho comercio, que es llevado a cabo por el método 'loadProducts', almacenando los productos en la variable 'productos'. El método 'searchProducts' es el encargado de realizar la búsqueda del producto que se ha tecleado en la base de datos, y que este se devuelva. Una vez devuelto, se tiene la oportunidad de agregar al carrito del cliente específico, que se crea si su carrito está vacío, y finalmente se agrega el producto al mismo. Se muestra una alerta si el producto es agregado correctamente o una alerta de error en caso de que se haya producido algún error a la hora de agregar el producto seleccionado. Si el producto ya ha sido añadido previamente al carrito, se muestra un mensaje informando sobre el suceso. En el método 'handleGoBack' se diferencia entre los cuatro usuarios, que depende de qué variable booleana esté activa, ya que podrían ocasionarse accesos no autorizados.

RegistroClienteController: En esta clase se realizan los métodos para llevar el registro del cliente. Se hace uso de un generador de números aleatorios en el método 'generateRandomNumber' para la generación del id del cliente. En el método 'handleSaveButtonAction' se lleva a cabo el guardado de los datos del formulario de registro del cliente una vez se han rellenado los campos que se requieren para la causa, donde se comprueba si hay algún fallo en los campos, y seguidamente se crea y almacena el registro del cliente en la base de datos. Se ha realizado el método 'esEmailValido' para verificar que el correo introducido cumple el formato exacto. De la misma manera que los anteriores, se hace uso del método 'goBack'.

RegistroMenuController: En esta clase se lleva a cabo el control del menú de registro, donde se puede elegir entre uno de los dos usuarios que el usuario anónimo puede escoger. Por ello, en los métodos 'handleClientRegisterButton' y 'handleCommerceRegisterButton' se redirige a la pantalla de registro dependiendo en cuál se pulse. Se hace uso del método 'goBack'.

SolicitudComercioController: En esta clase se lleva a cabo el formulario de solicitud que el comercio puede rellenar. Por ello, se ha llevado a cabo la realización de un formulario junto a los campos requeridos para el registro del comercio. Es por ello por lo que se realiza el método 'handleSaveButtonAction', en el que se comprueban los campos introducidos por el usuario comercio, que, en el caso de ser correctos, se lleva a cabo el registro de la solicitud a la espera de la revisión de un administrador. Se hace uso del método 'goBack'.

SolicitudesController: En esta clase se realizan los métodos necesarios para el manejo de las solicitudes del comercio desde el punto de vista del administrador. Se encuentra el método 'initialize', el cual obtiene los nombres de los comercios los cuales han realizado la solicitud, haciéndolos accesibles mediante la creación de botones, agregando los mismos a la variable 'nombres'. Estos botones serán accesibles gracias al método 'mostrarDatosComercio', donde se establece una conexión con la base de datos con el propósito de extraer la solicitud del comercio al cual se desea acceder. Una vez recopilada la información en la variable 'datos', se puede aceptar o rechazar la solicitud. Los métodos 'numeroAleatorioEnRango', 'cadenaAleatoria' y 'hashPasswordSHA256' se emplean para la creación del comercio en caso de que su solicitud se acepte. Como novedad, se puede observar las implementaciones de los métodos 'enviarCorreo' y 'enviarCorreoDenegacion', los cuales envían un correo a un correo de prueba establecido para informar al comercio tanto si se acepta como si se rechaza la solicitud. En caso positivo, se adjuntan las credenciales de este. También se implementa el método 'goBack'.

4

Implementación de la aplicación

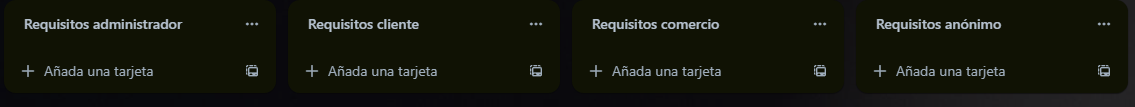
4.1 Visión general

En la implementación de una aplicación de software, se llevan a cabo una serie de pasos previos al desarrollo. Esta etapa del desarrollo es extensa debido a la variedad de pasos que conforman el desarrollo de una aplicación software, así como la extensión que abarcan. Sin embargo, se ha reducido la extensión de estos pasos atendiendo a los aspectos clave y más relevantes del sistema bajo desarrollo, centrándose en los principales apartados que se detallan a continuación.

4.2 Planificación

En esta fase, se han clasificado los requisitos previamente detallados en el software Trello [4] utilizado para la administración del proyecto. Por ello, se ha llevado a cabo la utilización del método Kanban [5] para la visualización del flujo del trabajo, donde el proceso de trabajo llevado a cabo se aclara llevando a cabo un progreso del trabajo más visible. Ya que en este proyecto se va a llevar a cabo una metodología derivada de Scrum con adaptaciones a este proyecto, se sigue por consiguiente la división de las etapas en sprints, estableciendo un periodo medio de dos semanas por cada uno.

Con el deseo de clasificar los requisitos del sistema en base a su grado de importancia, se ha diseñado una leyenda de colores, en la cual se asignan tres colores: rojo, para indicar la importancia máxima; amarillo, para denotar una importancia destacable; y verde, que señala la menor importancia. De esta manera se pretende priorizar la implementación de los requisitos en base a la relevancia en el sistema.



*Figura 20. Tablas de requisitos.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 21. Tablas Kanban utilizadas para establecer el estado de implementación de los requisitos.*

Como se puede observar en las *Figuras 20 y 21*, se ha realizado la división tal y como se ha detallado en el apartado de requisitos, escogiendo en cada sprint los requisitos que se han estimado oportunos para su implementación, haciéndose uso de tres tablas para establecer el estado de implementación de estos.

Dichos requisitos, una vez se proponían para ser implementados, se trasladaban a la tabla 'Por hacer'. Una vez su implementación comenzase, se pasaban al estado 'Haciéndose'. Finalmente, si la implementación y, por ende, el funcionamiento de dicho requisito concordaba con la descripción de este, se procedía a trasladar a la tabla del sprint correspondiente al que se encontraba el proyecto en ese momento, dándose por finalizada su implementación. A continuación, se detallan los sprints realizados en la *Figura 22.*

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

*Figura 22. Sprints del proyecto.*

*.*

En el primer Sprint se elaboraron un par de requisitos para dar los primeros pasos. La razón y objetivo de esta decisión fue interceptar los problemas que pudieran causarse durante el estudio e instalación de las tecnologías que se iban a utilizar a posteriori para la elaboración del proyecto.

Como se puede observar, el segundo sprint es el más denso de los cuatro que forman las etapas de desarrollo. Una vez comprobado que las tecnologías fueron instaladas correctamente, se propuso realizar una exhaustiva batería de requisitos para recuperar el ritmo y realizar aquellas tareas que no se agregaron en el primer sprint.

Al contrario que el segundo, este tercer sprint fue dividido. Esto se debe a que se presentaron problemas de actualización y agregación de registros en las tablas de la base de datos del sistema, por lo que fue un sprint en el que no se pudieron realizar los requisitos que se discutieron previamente, centrándose más en la resolución de los problemas encontrados. De esta manera, en el sprint 3a se realizó el requisito que muestra la imagen debido a los errores encontrados anteriormente mencionados. Una vez resuelto el problema, en el sprint 3b se desarrollaron los requisitos que no pudieron ser realizados originalmente.

En el cuarto y último sprint, podemos observar que gran parte de los requisitos restantes son del usuario anónimo salvo un par de funcionalidades que faltaban por implementar de varios actores.

4.3 Prototipado

En este proyecto se ha realizado un *mockup* (prototipo navegable a nivel de interfaz sin implementación de funcionalidades) con el propósito de dar un vistazo al primer modelo planeado de la aplicación, confirmar y actualizar los requisitos iniciales, y tener una base para la realización de las pruebas.

Este modelo nos permite obtener visualizar una posible interfaz que se ajuste al diseño propuesto y comprobar la validación de los requisitos de cara a los posibles usuarios finales. Puesto que ha sido realizado en etapas tempranas del proceso de desarrollo de software, está abierto a futuras modificaciones. Se muestran a continuación cada una de las interfaces iniciales propuestas, junto a una breve descripción, sin entrar en tecnicismos. Cabe destacar que este primer modelo está enfocado en la funcionalidad misma de la aplicación y no tanto en la idoneidad de la distribución de los elementos dentro de la interfaz.

En la *Figura 21* se representa la pantalla de inicio que el usuario visualiza al ejecutar la aplicación, en la cual se puede acceder a las distintas pantallas de registro e inicio de sesión, así como a la lista de comercios.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Figura 23. Pantalla de inicio del prototipo.*

En la pantalla de la *Figura 23* se presenta el menú de inicio de sesión de los tres tipos de usuarios que deben poseer una cuenta para iniciar sesión en la plataforma.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

*Figura 24. Pantalla de elección de inicio de sesión.*

Se puede observar en la *Figura 24* el *mockup* del inicio de sesión de cada uno de los usuarios registrados en la plataforma.

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

*Figura 25. Pantalla de inicio de sesión.*

Una vez se ingresa como el cliente, en la *Figura 25* se puede observar la pantalla principal del cliente, donde se pueden acceder a varias de sus funcionalidades implementadas, tales como el acceso a su perfil, la visualización de los comercios, y el cierre de sesión.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 26. Pantalla principal del cliente.*

Una vez el cliente iniciase sesión con sus credenciales, se muestra la interfaz de la *Figura 26* que representa a la pantalla principal del cliente teniendo capacidad para probar las diversas funcionalidades que posee este usuario en la aplicación.

Interfaz de usuario gráfica

Descripción generada automáticamente

*Figura 27. Visualización de la lista de comercios.*

El cliente puede desplegar un menú en el cual se muestran los comercios registrados en la plataforma tal y como se puede visualizar en la *Figura 27*, pudiendo acceder a cada uno de ellos pulsando el botón correspondiente al comercio que desee visitar.

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

*Figura 28. Perfil del cliente.*

Una vez se pulse el botón para acceder al perfil, en la *figura 28* se muestran las distintas opciones a las que el cliente puede acceder tal y como se han detallado en el apartado de descripción de requisitos.

Interfaz de usuario gráfica

Descripción generada automáticamente

*Figura 29. Edición del nombre de usuario del cliente.*

En la *figura 29* se muestra la pantalla donde el usuario puede editar el nombre de usuario que posee, salvando los cambios una vez introduzca un nuevo nombre de usuario. Esta acción se puede realizar con cada dato identificativo que describe al usuario, siempre que el sistema lo permita.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Figura 30. Carrito del cliente.*

En la figura 30 se muestra la interfaz del carrito que cada cliente posee, mostrando los productos que añadió anteriormente, el precio total y un botón que simula la tramitación y finalización del pedido.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Figura 31. Pantalla principal del comercio.*

Como se puede observar a continuación, la *figura 31* corresponde a la pantalla principal del usuario comercio, interfaz en la cual se puede acceder a las diversas funcionalidades implementadas tales como la visualización de los productos publicados, pudiéndose profundizar en la publicación o edición de estos, visualizar su perfil y el cierre de su sesión.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Figura 32. Prototipado de la visualización de los productos del comercio.*

En la *figura 32* se muestra la interfaz donde se almacenan los productos creados por el usuario comercio, donde se pueden crear más productos si el usuario lo desea. Al clicar en cualquiera de los productos, se muestra la pantalla de la *figura 33* en el cual puede eliminarse o editar el producto seleccionado anteriormente.

Tabla

Descripción generada automáticamente

*Figura 33. Pantalla de modificación de un producto.*

Finalmente, el usuario puede acceder a los detalles de su perfil accediendo a las pantallas correspondientes de las *figuras 34 y 35* pudiendo editar cada uno de los campos mostrados en la figura, así como cambiar las credenciales y la eliminación definitiva de su perfil del comercio.

Tabla

Descripción generada automáticamente

*Figura 34. Perfil del comercio.*

Interfaz de usuario gráfica

Descripción generada automáticamente

*Figura 35. Edición del perfil del comercio.*

En la figura 36 se muestra la interfaz de creación del nuevo producto que el comercio puede añadir a la plataforma, estando disponible en la plataforma al confirmar su creación.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

*Figura 36. Pantalla de nuevo producto.*

Como se puede observar en la *Figura 37*, esta pertenece al usuario administrador, donde se pueden apreciar los distintos apartados que corresponden a dicho usuario, tales como el manejo de solicitudes del comercio, el acceso a los datos de su perfil, y la visualización de la lista de comercios registrados en la plataforma y el cierre de su sesión, donde se pueden visualizar las características mencionadas en las *figuras 38, 39 y 40*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 37. Pantalla principal del administrador.*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Figura 38. Manejo de las solicitudes emitidas por los comercios.*

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

*Figura 39. Perfil del administrador.*

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

*Figura 40. Edición del nombre del administrador.*

Finalmente, en la *Figura 41* se muestra la pantalla dirigida al menú de registro de los usuarios cliente y comercio, los cuales redirigen a una pantalla en la cual se muestra el formulario que deben rellenar para formalizar su registro, mostrados en las *figuras 42 y 43* respectivamente.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Figura 41. Menú de registro de usuario.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 42. Pantalla de registro del cliente.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 43. Pantalla de la solicitud del comercio.*

4.4 Desarrollo de la aplicación

Una vez realizado el modelo inicial de la aplicación, se ha realizado un proyecto en Eclipse [7] para su implementación. Se ha estructurado una división típica en paquetes para una organización lo más clara posible de los archivos que se van a crear, utilizándose Java debido a que es un lenguaje de programación que ya conocía, haciendo que la curva de aprendizaje de los nuevos componentes y librerías no sea excesivamente grande.

La implementación se llevó a cabo siguiendo un enfoque de una clase controladora por cada vista creada. Esta decisión se tomó para realizar una división de las funcionalidades, evitando la agrupación de todas ellas en un mismo archivo, permitiendo una estructura más organizada y clara durante el desarrollo del proyecto.

Para el tratamiento de datos del usuario que ha accedido a la plataforma, se llevó a cabo la implementación de la clase mostrada en la *figura 44*, la cual se encarga de tratar los datos identificativos del usuario. Se realizó esta clase debido a que era necesario la obtención de sus datos para su posterior muestra en las pantallas donde fuesen necesarias.

Gráfico

Descripción generada automáticamente con confianza baja

*Figura 44. Clase SesionUsuario.*

Dada la extensión de métodos de la aplicación, se procederá a explicar aquellos más relevantes en virtud de su carácter innovador dado que se ha requerido el uso de librerías distintas a las conocidas generalmente en el aprendizaje y manejo del lenguaje Java.

Texto

Descripción generada automáticamente

*Figura 45. Método para enviar un correo en caso de aprobación de solicitud.*

En la *figura 45* podemos observar el método encargado de enviar un correo en caso de que la solicitud sea aprobada por parte del usuario administrador. Para ello, se obtiene por parámetro el nombre del comercio que ha rellenado y enviado la solicitud, el usuario y contraseña establecido por el administrador, y el correo del comercio.

En primer, se crean las variables que componen un correo electrónico. Seguidamente, se crea una variable que contiene la clave de aplicación. Esta clave consiste en una serie de letras que Google proporciona para aplicaciones de terceros y establecer conexión con el servidor SMTP.

A continuación, se crea un objeto 'Properties' para establecer las distintas propiedades que componen el envío de un correo, tales como: el servidor SMTP de Google, el remitente del correo, la clave de la aplicación, la activación de autenticación mediante usuario y clave, el establecimiento del protocolo TLS para proporcionar un canal seguro para el envío de correos, y el puerto SMTP utilizado por Google. Una vez se ha creado este objeto con todas las propiedades necesarias, se establece la sesión con estas propiedades, seguido de la creación del objeto MimeMessage para la representación del mensaje en formato MIME (Multipurpose Internet Mail Extensions) y proporcionar así funcionalidades para, en este caso, enviar el correo electrónico. Una vez se establecen todos los campos del objeto mencionado anteriormente, se establece la conexión con el servidor SMTP y se envía el mensaje. Una vez es enviado, se cierra la conexión con el servidor.

Cabe destacar que también se ha creado exactamente el mismo método en caso de rechazar la solicitud, aunque no se mostrará la figura correspondiente, pues es idéntico al de la *figura 45* salvo la variable del cuerpo del mensaje, indicando que se ha rechazado la solicitud.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Figura 46. Hasheo de la contraseña del administrador usando SHA-256.*

Como se puede observar en la *figura 46, p*ara realizar el hash de la contraseña del administrador cuando se realice el inicio de sesión, se ha utilizado la función criptográfica SHA-256 [3], tal y como se mencionó en el apartado 3.4.1. Para ello se ha hecho uso de la clase MessageDigest, encargada de realizar el hash a la contraseña para aumentar la seguridad e integridad de esta.

En primer lugar, se le añade a la contraseña un valor de sal para aumentar su seguridad. Una vez realizado, se establece la función SHA-256 [3] para el hash de la contraseña.

Seguidamente se transforma la contraseña junto con la sal a un array de bytes. Esto se debe a que es necesario convertir la cadena a un array de bytes para su posterior procesamiento en el bucle, el cual cada byte se convierte a su forma hexadecimal con un ancho de dos dígitos y almacenándolo en una variable de la clase StringBuilder que es la encargada de devolver la contraseña finalmente hasheada con el algoritmo establecido.

Texto

Descripción generada automáticamente

*Figura 47. Comprobación de credenciales del usuario cliente.*

En la *figura 47* se lleva a cabo la comprobación de las credenciales de, en este caso, el usuario cliente. Cabe destacar que este método es auxiliar, pues solamente se encarga de tratar las credenciales introducidas exclusivamente para su comprobación con la base de datos del servidor. Una vez se reciben las credenciales introducidas por el usuario, se realiza una búsqueda en la base de datos, la cual, si el usuario existe, crea la sesión de usuario junto a sus datos (la existencia de datos a nulo se debe a que esos campos no pertenecen al usuario cliente, pero sí a otros usuarios que lo requieran). A continuación, se lleva cabo el hash de la contraseña introducida por el usuario junto al valor de sal almacenado en la base de datos gracias a la librería BCrypt [1], el cual, si coincide con el hash almacenado y el usuario introducido, se procede a acceder a la plataforma de manera satisfactoria.

Texto

Descripción generada automáticamente

*Figura 48. Método para cambiar de pantalla.*

Por último, se procede a mostrar en la *figura 48* el método más utilizado en la aplicación. Este método es el encargado de la navegación por la aplicación. En esta figura, se muestra el regreso al menú de elección de registro al pulsar el botón representado con una flecha apuntando a la izquierda. Primero, se crea un objeto de la clase FXMLLoader, que es la encargada de cargar los archivos FXML utilizados para la realización de la interfaz. Una vez se carga en el objeto correspondiente, se procede a la creación de una variable de la clase Parent, que es una superclase para todos los elementos gráficos, cargando la escena en esta variable. Una vez está cargada, se crea la escena (Scene) en la que se mostrará la pantalla, pues es el contenedor principal de la pantalla, seguido de la obtención de la ventana actual en la que se encuentra el usuario para no abrir otra ventana, estableciendo la nueva escena en la pantalla actual y mostrándola finalmente.

Diagrama

Descripción generada automáticamente

Figura 49. Estructura del proyecto.

A continuación, se detallan los paquetes mostrados en la *figura 49:*

* ***images:***En este paquete se almacenan las imágenes que la aplicación muestra una vez se ejecutan, tales como las imágenes de los productos, o las mostradas en la interfaz de la aplicación.
* **mm.makery.app:** Este paquete contiene la clase Main del proyecto, el cual se ha aislado del resto del proyecto como parte de la buena práctica de un proyecto hecho en Java.
* **mm.makery.app.model:** Este paquete contiene la clase SesionUsuario, la cual se ha creado con el objetivo de almacenar la sesión del usuario activo en la plataforma.
* **mm.makery.app.style:** En esta clase se almacenan las hojas de estilo CSS del proyecto.
* **mm.makery.app.view:** En este paquete, se almacenan tanto las clases controladoras como las vistas de la aplicación.
* **mm.makery.app.test:** En este paquete se encuentra la clase realizada para los tests correspondientes al carrito.

Una vez se crearon los paquetes, nos basamos en el descrito en el apartado 4.3 con el propósito de empezar adecuadamente la implementación de la aplicación con una primera interfaz minimalista en la cual se pudiera comprobar el correcto funcionamiento de las funcionalidades implementadas en cada sprint.

4.5 Base de datos

Para el correcto funcionamiento de la aplicación, se requiere el uso de una base de datos para el almacenamiento de los datos con los que se trabajan en esta. La *Figura 43* muestra el modelo entidad-relación que se ha utilizado para la creación de esta base de datos.

Diagrama

Descripción generada automáticamente

*Figura 50. Modelo entidad-relación.*

Como se puede apreciar en la *figura 50*, se observan las siguientes tablas:

* Cliente: Tabla en la que se almacena los datos relativos al cliente, concretamente el DNI que se ha establecido como clave primaria, y los datos identificativos recogidos en el registro del cliente, a excepción de la sal creada una vez se establece la contraseña. Se ha establecido una relación uno a muchos con carrito, pues un cliente puede realizar múltiples compras y ese carrito pertenece a un cliente en específico.
* Carrito: Tabla donde se almacenan los productos añadidos al carrito por un cliente, en concreto el id del producto y el id del carrito que actúan como claves primarias, la cantidad de producto que se ha almacenado y el precio por unidad de producto, estableciéndose una relación muchos a uno con producto, pues cada entrada de la tabla carrito se relaciona con un único producto y un producto puede estar presentes en varios carritos distintos.
* Producto: Tabla en la cual se almacena la información relativa a un producto, almacenando atributos como el id del producto como clave primaria, nombre, precio, descripción, la ruta de la imagen y el id de comercio obtenido por la relación. Se establece como novedad una relación muchos a uno con comercio, pues un comercio puede tener varios productos agregados a la plataforma y ese producto pertenece a un comercio en específico.
* Solicitudcomercio: Tabla cuyo objetivo reside en el almacenamiento de las solicitudes emitidas por el comercio, estableciendo como clave primaria el NIF del comercio.
* Comercio: Tabla que almacena la información que identifica al usuario comercio estableciendo como clave primaria el NIF.
* Administrador: Tabla que almacena la información relativa al usuario administrador, donde se establece un id de administrador como clave primaria que lo identifica.

4.6 Front-end de la aplicación

El diseño de la interfaz se ha realizado gracias a SceneBuilder [8]. Se ha escogido esta aplicación debido a la interfaz sencilla e intuitiva que posee, en la cual se permite diseñar las ventanas de la aplicación sin escribir código. En la *Figura 51* se muestra la interfaz que posee una de las ventanas de la aplicación durante el proceso de creación de esta.

Interfaz de usuario gráfica

Descripción generada automáticamente

*Figura 51. Pantalla de inicio de la aplicación.*

En cada pantalla de la aplicación, se permite establecer los elementos que forman la pantalla, la clase que actúa como controlador y editar los elementos que conforman la vista al gusto del usuario, pudiendo también implementar código para modificar el estilo de los elementos.

4.7 Pruebas a la aplicación

Por último, se ha realizado una batería de pruebas para comprobar el funcionamiento de la aplicación. La realización de una batería de pruebas completa que abarque el funcionamiento total del sistema y que pruebe todos los posibles caminos del código (junto con las pruebas de caja negra) resultaría en una inversión de tiempo y esfuerzo no asumible para las restricciones temporales y de personal de este proyecto. No obstante, con el objeto de cumplir con todas las fases de un proceso de ingeniería de software, se ha llevado a cabo una batería de pruebas al carrito de compra del cliente debido a que es la que involucra una transacción económica por parte del cliente hacia el comercio, por lo que en las próximas figuras de este apartado se detallan los métodos y test realizados para la funcionalidad explicada anteriormente, donde se resume la funcionalidad de los métodos justo antes de su declaración. Aunque, el resto de las funciones de la aplicación no han gozado de una batería de pruebas automatizada, ha sido probada manualmente durante el proceso de desarrollo.

Texto

Descripción generada automáticamente

*Figura 52. Métodos realizados para las pruebas.*

Texto

Descripción generada automáticamente

*Figura 53. Métodos de prueba.*

Texto

Descripción generada automáticamente

*Figura 54. Test realizados.*

Con estos tests se ha pretendido, en la medida de lo posible, representar escenas de simulación en la cual el sistema sepa capturar y mostrar las excepciones correctamente, así como el correcto mostrado de resultados que son sumamente importantes para el funcionamiento del carrito del cliente.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Figura 55. Ejecución de los tests.*

4.8 Problemas surgidos en el desarrollo

Hoy en día es sumamente extraño que no haya ningún contratiempo durante todo el desarrollo de un producto software ni situaciones de estancamiento por problemas de las tecnologías o de la implementación del sistema. En el desarrollo de este trabajo de fin de grado no ha sido menos, ya que se han presentado algunas situaciones problemáticas las cuales han hecho que, por ejemplo, se retrasara uno de los sprints establecidos.

Durante el desarrollo de este proyecto, se encontraron problemas de actualización de registros en la base de datos, que hizo que se retrasase la implementación de la aplicación hasta encontrar una solución, ya que es imprescindible el correcto funcionamiento de las sentencias SQL utilizadas para el almacenamiento de datos. También se tuvo que remodelar el diagrama de clases realizado durante el diseño del proyecto, pues durante la implementación se vio la necesidad de actualizar el diseño inicial para ajustarse a los requisitos especificados (los cuales también sufrieron modificaciones durante el proceso de desarrollo), por lo que finalmente se elaboró el diagrama de clases que se muestra en la *Figura 18.* También se presentaron errores en la creación de ficheros FXML desde el entorno de desarrollo utilizado, donde la solución fue crear el archivo desde la carpeta del proyecto manualmente haciendo desaparecer ese error.

Durante la implementación del proyecto, se encontró un problema para la muestra de los datos del usuario que inicia sesión, pues si estos eran modificados debían actualizarse en el perfil en caso de que el usuario desease visualizar y comprobar que son correctos. Tras barajar varias opciones, se llevó a cabo la creación de una clase externa al sistema para el manejo de los datos del usuario que ha iniciado sesión en la plataforma independientemente del que sea, almacenando sus datos y pudiendo acceder a ellos siempre que se quisiera.

5

Conclusiones

5.1 Conclusiones

Para finalizar, los objetivos primordiales de este proyecto software han sido completados. Se han elaborado los procesos primordiales de un proyecto software lo más detalladamente posible, creando una aplicación totalmente funcional. Gracias a este proyecto, he descubierto nuevos entornos y tecnologías en las que he podido profundizar lo máximo posible durante el desarrollo, tanto lenguajes de programación como programas de diseño de procesos. La realización de este trabajo también me ha sido de gran ayuda para la resolución de los problemas presentados durante el desarrollo de todo el proyecto, contribuyendo a una mejora de la eficacia.

El aprendizaje de todas las tecnologías utilizadas en este proyecto supone un gran aumento de mis conocimientos informáticos de cara al mundo laboral, ya que se ha seguido y llevado a cabo un enfoque ingenieril durante todo el proceso de desarrollo de ese trabajo. Ya que tengo un gran interés por el mundo de la ciberseguridad y la seguridad de la información y, dada la importancia de los datos que se manejan en esta aplicación, no descarto seguir en un futuro con el desarrollo de la aplicación, así como su puesta en producción, añadiendo los requisitos a futuro descritos y las capas de seguridad necesarias para el correcto funcionamiento y encriptado de los datos almacenados en la base de datos.

Pese a los problemas encontrados, pienso que el resultado final cumple con las expectativas que tenía antes de empezar a desarrollarlo, ya que el objetivo de este era poner en práctica lo aprendido durante mis estudios en la universidad y utilizar tecnologías con las que no tuve oportunidad de hacerlo. En cuanto al resultado final obtenido, se ha construido una aplicación funcional y robusta que cumple con los requisitos establecidos inicialmente y que además tiene potencial para seguir creciendo gracias a los requisitos futuros ya planteados, no descartándose incluso la adición de nuevos requisitos además de los ya planteados.

5.2 Mejoras de la plataforma en un futuro

Si se tiene una visión del futuro para mejorar la aplicación, una de las mejoras que se Implementaría sería una mejora significativa de la interfaz, ya que se ha elaborado una Interfaz minimalista ya que durante el desarrollo se ha priorizado el funcionamiento de la aplicación antes que la visualización de este. Por otro lado, se desplegaría la aplicación en la web, así como la puesta en funcionamiento de la base de datos en la nube. Por último, se Implementarían los requisitos a futuro descritos ya que podría dar comienzo a un negocio, donde se podría Implementar alguna política de cobro para rentabilizar la aplicación.

Referencias

[1] Librería BCrypt:

https://en.wikipedia.org/wiki/Bcrypt

[2] Visual Paradigm

https://www.visual-paradigm.com

[3] SHA-256

https://es.wikipedia.org/wiki/SHA-2

[4] Trello

https://trello.com

[5] Kanban

https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

[6] Scrum

https://www.atlassian.com/es/agile/scrum

[7] Eclipse

https://eclipseide.org

[8] SceneBuilder

https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html

[9] Librería Javax.mail