TRABAJO FIN DE GRADO

Grado en Ingeniería Informática de Servicios y Aplicaciones

PocketMix

Monedero virtual para almacenar tus tarjetas de puntos

Autora
Bianca Claudia ILas (alumna)

Director
Mª Luisa Martín Pérez (tutora)

Escuela de Ingeniería Informática de Segovia

—

Segovia, Diciembre de 2023

Índice general

Índice de Tablas

Índice de Figuras

PoketMix

Monedero virtual para almacenar tus tarjetas de puntos.

Autora
Bianca Claudia ILas (alumna)

Directora
Mª Luisa Martín Pérez (tutora)

Monedero virtual para almacenar tus tarjetas de puntos

Bianca Claudia ILas (alumna)

Palabras clave: Android, aplicación móvil, Kotlin, monedero virtual, tarjetas, fidelización, comercios

Resumen

Este Trabajo de Fin de Grado tiene como finalidad la creación de una aplicación móvil enfocada a plataformas Android que permita el escaneo y guardado de todas nuestras tarjetas de fidelización de distintos comercios. El usuario cuenta con una herramienta intuitiva y simple que facilite todo lo posible este proceso para poder disponer de su monedero virtual. El objetivo de este proyecto se basa simplificar y agilizar la experiencia del usuario en cuanto a la utilización vía móvil de las tarjetas de puntos de las que disponga mediante su inserción y agrupación en una misma aplicación, empleable en cualquier establecimiento, cuya creación vamos a exponer.

Virtual wallet to store your point cards.

Bianca Claudia ILas (student)

Keywords: Android, mobile application, Kotlin, virtual wallet, cards, loyalty, shops

Abstract

This project verses about the creation of a mobile application focused on Android platforms that allows the scanning and saving of all our loyalty cards from different businesses.

As a starting point, the user has an intuitive and simple tool that makes this process as easy as possible to be able to have their virtual wallet.

The main objective has been to simplify and speed up the user experience using mobile phones as point cards that have their insertion and grouping in the same application. This is usable in any establishment, whose creation we are going to expose. A series of previous results has been used as a starting point to develop the whole topic discussion along this project.

Agradecimientos

Primeramente voy a dar las gracias a mi familia, a mi madre, por su apoyo incondicional, por sus noches en vela, por estar ahí siempre que la he necesitado y por sacar de mí la fuerza necesaria para llegar a este momento; a mi padre que aunque ya no esté con nosotras me dio siempre el valor necesario para conseguir mis metas; y finalmente a todas las personas que siempre han estado recorriendo este camino conmigo.

Especial agradecimiento quería darle en primer lugar a mi Universidad, sobre todo en especial a Marisa que ha aceptado dirigir este reto para mí resolviendo todas mis cuestiones y sirviendo como punto de apoyo.

Capítulo 1

Descripción del proyecto

1.1. Introducción

En la actualidad, todos estamos familiarizados con el concepto de tarjetas de fidelización, pudiendo ser físicas o electrónicas. Estas tarjetas se suelen entregar a los clientes de un negocio o de empresas de manera gratuita para crear un vínculo con ellos. De esta forma, pretenden asegurarse de que el cliente vuelva a adquirir productos pertenecientes a esa marca, establecimiento o comercio.

Las tarjetas de fidelización también se conocen como tarjetas de descuentos, promociones o de puntos para clientes y son un muy buen recurso para fidelizarlos. Esto es así, porque ofrecen un trato exclusivo a los usuarios y además, las dos partes salen ganando: la empresa obtiene información de los consumidores y de sus preferencias, lo que ayuda a mejorar las estrategias de marketing; y por otro lado, los compradores se benefician de las ventajas ofrecidas por la marca.

Por todo ello, estas tarjetas son una herramienta muy atractiva para los usuarios, porque tienen la percepción de que están comprando de forma más rentable al tener en cuenta los consumos que realizan con regularidad. Los descuentos, las rebajas, la participación en sorteos o la obtención gratuita de artículos a medida que se suman puntos son las principales ventajas de unirse a estas iniciativas. Además, las tiendas ven incrementadas las compras ya que cada vez que un usuario las realiza, lo que quiere es sacarle partido a su tarjeta de cliente.

En el informe "Fidelidad en la compra" de 2019 de Nielsen [7], indica que seis de cada diez españoles tienen al menos una tarjeta de fidelización en un establecimiento. Este artículo remarca que el 35 % de estos clientes está dado de alta en entre dos y cinco programas de fidelización, mientras que el 9 % lo está en más de seis. Muchos de estos programas dependen de tarjetas físicas, o se gestionan a través de aplicaciones separadas, propias de cada empresa. En consecuencia, la gestión de los múltiples programas de fidelización en los que un usuario está inscrito se convierte en una molestia, puesto que debe portar una gran cantidad de tarjetas o estar pendiente de múltiples plataformas para disfrutar de sus ventajas.

Como hemos comentado, con el paso del tiempo solemos acumular decenas de tarjetas de fidelización de diferentes establecimientos que al final acabamos por olvidar en casa al ocupar

demasiado espacio en nuestra cartera y solo las utilizamos en situaciones puntuales. Sin embargo, existen aplicaciones con las que podremos usar nuestras tarjetas promocionales y de fidelización desde nuestro dispositivo móvil sin tener la necesidad de llevarlas encima, facilitándonos el acceso a los descuentos y ofertas.

A pesar de que el gigante Internet facilita su utilización digital, la realidad es que no todo el mundo maneja el ordenador. En cambio, el dispositivo móvil es mucho más usado y además, los monederos digitales utilizan las capacidades inalámbricas de estos dispositivos, como Bluetooth, WiFi y señales magnéticas, para transmitir los datos de forma segura desde el dispositivo a un punto de venta diseñado para leer los datos y conectarse a través de estas señales.

Se propone pues la idea de crear una aplicación móvil que permite centralizar la gestión en todas nuestras tarjetas de fidelización en una única aplicación. Mediante esta herramienta almacenaremos todos nuestros números de socio o códigos de acceso que necesitamos mostrar en los comercios para aplicar los descuentos que nos corresponden por ser clientes y mejorar la comodidad del usuario.

1.2. Objetivos del proyecto

El objetivo principal que se quiere alcanzar en este trabajo es la creación de una aplicación Android que facilite y agilice a los usuarios de los clubes de fidelización tener todas sus tarjetas de puntos almacenadas en un único espacio virtual, vaciando así su cartera de las tarjetas físicas.

Teniendo claro que este es el objetivo principal que queremos conseguir, tendremos en cuenta otros factores que van a complementar nuestra meta y proporcionan una mejor funcionalidad para los consumidores. Estos objetivos son:

Objetivo 1	Desarrollo de una aplicación Android para la gestión centralizada de los programas de fidelización de un usuario.
Objetivo 1.1	Mejorar la seguridad de la aplicación mediante la implementación de medidas y el mantenimiento de credenciales de programas de fidelización.
Objetivo 1.2	Gestión de compras mediante programas de fidelización.
Objetivo 1.3	Gestión de ventajas de programas de fidelización.
Objetivo 2	Implementar una arquitectura de servidor escalable que permita gestionar el crecimiento en el número de usuarios, garantizando la estabilidad y disponibilidad del servicio en todo momento.
Objetivo 3	Desarrollar una interfaz de usuario intuitiva y atractiva que facilite la navegación y el uso eficiente de todas las funciones relacionadas con la gestión de programas de fidelización.
Objetivo 4	Integrar tecnologías de escaneo de códigos (QR o códigos de barras) para agilizar y simplificar el proceso de registro de compras.
Objetivo 5	Desarrollo de un servidor escalable para el acceso concurrente de múltiples usuarios.
Objetivo 6	Optimizar el rendimiento de la aplicación para garantizar una respuesta rápida y eficiente.

Por otro lado, la aplicación cuenta con un sistema actualizado que deja en manos del administrador la gestión de toda la información producida por las tarjetas de fidelización así como la interfaz y el contenido ofrecido a los usuarios para que el sistema sea robusto y amigable.

1.3. Entorno de la aplicación

Como hoy en día vivimos pegados al teléfono móvil e incluso tiene funcionalidad de monedero en la mayoría de los casos, este trabajo se realiza con la intención de reducir la tenencia de tarjetas y agruparlas todas en nuestro smartphone que siempre llevamos encima.

La aplicación está orientada al contexto de desarrollo Android, por lo que debe ser compatible con las principales versiones de este sistema operativo en el momento de su lanzamiento. En esta aplicación, la compatibilidad mínima es un sistema operativo superior o igual a Android 5.

Pero este trabajo se presenta como una primera aproximación, con la idea de seguir avanzando y adaptándose a las necesidades que puedan ir surgiendo en la utilización de las tarjeta tanto de los usuarios que se benefician de ellas, como de las empresas que las ofertan.

Permite un aprendizaje ágil e intuitivo, robustez frente situaciones anómalas, facilidad en su uso por parte del usuario y garantía de la integridad de la información.

A continuación, se muestra el diagrama de Ishikawa que detalla la funcionalidad y características principales de la aplicación:

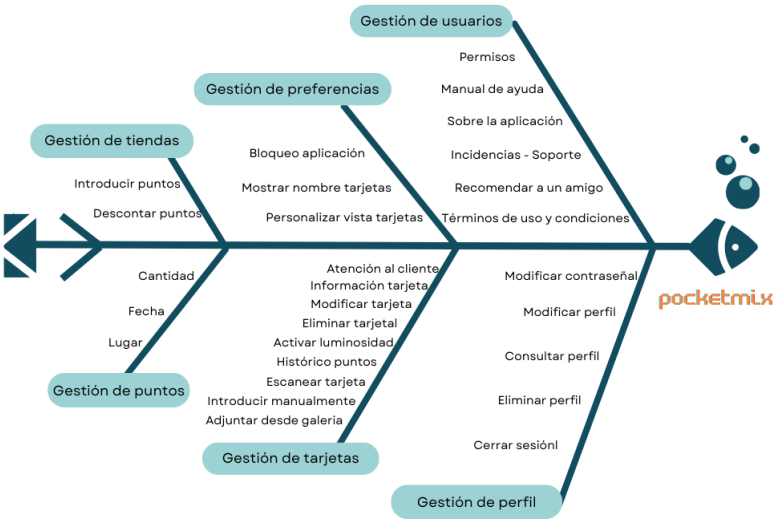

Figura 1.1: Diagrama de Ishikawa.

Como se puede observar en la figura anterior, la aplicación Android consta de un servidor externo, denominado PocketMix. En la parte superior se encuentra definida la gestión con los usuarios: configuración de su cuenta, modificación del perfil, permisos, preferencias...

- **Gestión de usuarios**: engloba las características y funcionalidades referentes a los usuarios a la hora de gestionar sus tarjetas, como también sus preferencias, tales como filtrar sus tarjetas fidelizadas.

Por otro lado, en la parte inferior, se definen todas las características básicas destinadas a la parte administrativa de la aplicación:

- **Gestión de tarjetas**: engloba la funcionalidad y las características necesarias para añadir, modificar, consultar, borrar, ordenar tarjetas y el historial de puntos.

Por último, con un servidor externo:

- **API PocketMix**: gestiona los puntos, las tarjetas, las tiendas y el contacto con ellas (web, teléfono, email).

Capítulo 2

Estado del arte

Las funcionalidades de nuestra plataforma están enfocadas en el escaneo de diferentes tarjetas de afiliación para los establecimientos y el beneficio de los clientes que las utilizan.

Para afrontar correctamente el desarrollo de nuestra aplicación, llevamos a cabo una investigación de otros entornos con características similares. De esta manera, estudiaremos qué propuestas y herramientas hay ya en el mercado actual y analizaremos sus aspectos más relevantes.

En este caso se puede observar una escasez de herramientas en el dominio que nos encontramos en cuanto a la gestión de las tarjetas de fidelización en nuestro dispositivo. Tras la búsqueda de varias aplicaciones que comentaremos en el siguiente apartado, la que utilizaremos como referencia porque puede acercarse a nuestro objetivo es Stocard, que detallaremos a continuación y que tomaremos como ejemplo para el desarrollo de nuestro proyecto.

2.1. Stocard

Figura 2.1: Logo Stocard.

Aplicación de origen alemán líder del mercado de carteras digitales en Europa cuyo objetivo principal es guardar todas tus tarjetas de fidelización en tu smartphone [22].

Por otro lado, muestra tu saldo actual de puntos, tiendas cerca de ti y también las mejores ofertas, cupones y catálogos de tus tiendas favoritas mejorando la experiencia con el usuario y atrayendo a nuevos públicos gracias a su diseño de alto nivel.

Recientemente ha realizado una renovación mediante la integración con *Klarna*, que es un servicio de pagos bancarios en línea, lo que le brinda la oportunidad de añadir la funcionalidad de comprar y pagar y de esta manera ser más segura y sencilla para los clientes.

Stocard está disponible para dispositivos iOS y Android además de disponer de un portal web.

Funcionalidades:

- Para que los clientes visualicen las ofertas de diferentes establecimientos cuenta con un diseño innovador de estados.
- Añadir tarjetas: mediante el escaneo, manualmente o desde un archivo sin necesidad de hacer un registro previo del usuario.
- Visualizar información de la tarjeta añadida: notas, fotos, últimas ofertas del catálogo y tiendas cercanas si activas la ubicación.
- Consultar ofertas de un catálogo en función de tus preferencias y de tus tarjetas añadidas de diferentes establecimientos.
- Permite darte de alta en la aplicación para no perder nunca tus tarjetas añadidas, mediante una cuenta Apple, Google o por email. Así como iniciar sesión con Apple, Google, email y Facebook.
- Compartir mediante un enlace para recomendar la app a un amigo y recibir puntos para el beneficio de ambas partes.
- Asistente de tarjeta, consiste en mostrar la tarjeta desde tu pantalla de bloqueo cuando te encuentres en tu tienda favorita. Necesario acceso a las notificaciones y a la ubicación.
- Abarca un ámbito internacional, dispone de diferentes regiones.
- Avanzado:
  - Bloqueo con Face ID.
  - Ordenar las tarjetas por *nombre, más usadas o utilizado recientemente.*
  - Eliminar cuenta.
- Otros:
  - Menú de ayuda para nuevos usuarios que no estén familiarizados con la aplicación, contiene diferentes apartados tales como: *información Stocard, utilizar Stocard, códigos de barras y escaneo, ofertas y catálogos, cuenta Stocard, tarjetas de regalo y pagar con Stocard y gestionar su cuenta.*
  - Contactar con soporte, se abre una pestaña nueva de email con un formulario intuitivo, para redactar una incidencia y enviársela al soporte de Stocard.

2.2. Yudonpay

Figura 2.2: Logo Yudonpay.

Era una aplicación gratuita creada en España en 2017, disponible para Android e iOS, enfocada en gestionar y guardar tarjetas de puntos en un mismo lugar y presente en cinco países latinoamericanos: Argentina, Chile, Colombia, México y Perú.

Funcionalidades:

- Almacenar diferentes tarjetas de puntos y establecimientos.
- Proporcionar información en tiempo real sobre tus puntos, cómo y en qué puedes gastarlos.
- Registro a través de Facebook, Google o con un correo electrónico en la aplicación.

2.3. FidMe

Figura 2.3: Logo FidMe.

Es una aplicación nacida en Francia en 2010 para guardar todas nuestras tarjetas de fidelización desde nuestro smartphone sin necesidad de llevarlas encima, capaz de almacenar y detectar tarjetas de comercios distintos [23]. Tras registrarnos en la aplicación, ingresamos una a una nuestras tarjetas que serán almacenadas a nuestra cuenta vinculada. Disponible para Android e iOS además de disponer de un portal web.

Funcionalidades:

- Registro obligatorio mediante un perfil de Facebook, correo electrónico, Google o Apple.
- Añadir tarjetas mediante escaneo o introduciendo los datos manualmente. Por cada tarjeta incluida, obtienes una bonificación de puntos en su programa *Happy Fids*.

- Permite organizar los tickets de compra escaneándolos o adjuntándolos desde un archivo, habilitando además el análisis de esta información para realizar informes anuales de compras en las principales marcas y tiendas.
- Recompensas por las compras ganando puntos gracias a la importación de los tickets para canjearlos por dinero o regalos.
- Notificaciones a través de mensajería para recibir información del historial de las tareas desempeñadas por el usuario.
- Perfil - mi aplicación:
  - Apartado *Mi Cuenta* donde te da la opción de añadir más información a tu perfil y una imagen.
  - Información de reembolso.
  - Mi centro de patrocinio.
  - Gestión de preferencias:
    - Personalizar la visualización de las tarjetas.
    - Mostrar nombres de los comercios.
- Ayuda a través de mail, gmail, outlook o yahoo para contactar con el equipo FidMe.
- Transparencia total:
  - Información sobre las condiciones generales de uso y aviso legal.
  - Protección de datos.
    - Carta de confidencialidad.
    - Información sobre la protección de datos de la aplicación.
    - Administración de la configuración de datos.
    - Extracción de datos.
    - Eliminación de la cuenta.

2.4. Mobile-pocket

Figura 2.4: Logo Mobile-pocket.

Es una aplicación desarrollada en 2001 por Bluesource para almacenar las tarjetas de fidelización en un único espacio [24]. Disponible para iOS y Android y cuenta con un Portal web.

Funcionalidades:

- Registro e inicio de sesión mediante correo o con una cuneta personal de Facebook, Google o Apple.
- Visualizar códigos de barras o QR para mostrarlos al pasar por caja.
- Añadir tarjeta: escaneo, manualmente y opción de activar flash para dar más luminosidad a la hora de escanear nuestra tarjeta.
- Visualizar catálogos de ofertas de diferentes comercios.
- Permisos para controlar el acceso, a la notificaciones, ubicación y a la cámara.
- Ordenar tarjetas alfabéticamente o de manera personalizada.
- Filtro de País.
- Apariencia del diseño de la aplicación.
- Modificar icono de la aplicación según nuestra preferencia.
- Menú de ayuda.
- Aceptar términos y condiciones.
- Recomendar a un amigo la aplicación.
- Información sobre la aplicación, de la versión, identificación.
- Opción de seguimiento en las redes sociales.

2.5. Comparativas de mercado frente PocketMix

Una vez estudiadas las funcionalidades de las aplicaciones similares a la desarrollada, compararemos con nuestra aplicación las características de cada una de ellas para analizar cómo de competente puede ser en el mercado. En la siguiente tabla detallaremos esta comparativa:

	Stocard	Yudonpay	FidMe	mobile-pocket	PocketMix
Escanear tarjeta	✓	✓	✓	✓	✓
Escanear tickets	×	×	✓	×	×
Introducir tarjeta manualmente	✓	✓	✓	✓	✓
Adjuntar tarjeta desde archivos	✓	×	×	×	✓
Adjuntar ticket desde archivos	×	×	✓	×	×
Activar flash	×	×	×	✓	✓
Modificar tarjeta	✓	✓	✓	✓	✓
Eliminar tarjeta	✓	✓	✓	✓	✓
Asistente de tarjeta	✓	×	×	×	×
Notas	✓	×	×	✓	✓
Tarjetas favoritas	×	×	×	×	✓
Comprar y pagar	✓	×	×	×	×
Canjear puntos	✓	×	✓	×	×
Puntos acumulados	✓	×	✓	×	✓
Visualizar promociones y ofertas	✓	✓	×	✓	×
Registro y acceso usuario	✓	×	✓	✓	✓
Modificar cuenta de usuario	✓	×	✓	✓	✓
Eliminar cuenta	✓	×	✓	✓	✓
Seguimiento en redes sociales	×	×	✓	✓	×
Ordenar tarjetas	✓	✓	✓	✓	✓
Preferencias visualizar tarjetas	×	×	✓	×	✓
Filtrar por país	✓	×	×	✓	×
Modificar diseño de la aplicación	×	×	×	✓	×
Modificar icono de la aplicación	×	×	×	✓	×
Funcionamiento en IOS-Android	✓	✓	✓	✓	×
Geolocalizar ofertas	✓	×	×	✓	×
Moneda virtual propia	✓	×	✓	×	×
Menú Ayuda	✓	✓	✓	✓	✓
Gestión de permisos de usuario	✓	×	✓	✓	✓
Términos de uso de la aplicación	✓	✓	✓	✓	×
Recomendar a un amigo	✓	×	×	✓	✓
Información de la aplicación	✓	✓	✓	✓	✓
Portal web	✓	×	✓	✓	✓
Cifrado de la aplicación	✓	×	✓	✓	✓

Tabla 2.1: Comparativa de mercado.

Capítulo 3

Planificación

A continuación, realizaremos un análisis de la metodología de desarrollo empleada en la realización del proyecto así como la planificación, los costes y estimaciones iniciales. Finalmente, detallaremos el coste final de todas teniendo en cuenta todas las restricciones que han afectado al desarrollo del sistema.

3.1. Metodología de trabajo

Para este proyecto en concreto he decidido optar por la **metodología iterativa e incremental** tras haber analizado las metodologías y paradigmas que se utilizan hoy en día en procesos de desarrollo software. La metodología incremental es una metodología de desarrollo de software que se va construyendo de manera progresiva. En cada etapa incremental se agrega una nueva funcionalidad, lo que permite ver resultados de una forma más rápida en comparación con el modelo en cascada.

La metodología iterativa, en conjunto con la incremental, se convierte en un enfoque poderoso para el desarrollo de software. La iteratividad implica la repetición de ciclos de desarrollo, cada uno de los cuales incluye la planificación, diseño, implementación y evaluación de una parte del sistema. Esta repetición de fases permite una retroalimentación continua, lo que posibilita ajustes y mejoras a medida que avanza el proyecto. En cada iteración, se priorizan y abordan aspectos específicos del software, lo que facilita la adaptación a cambios en los requisitos o la identificación temprana de posibles desafíos. La combinación de ambas metodologías brinda la flexibilidad necesaria para responder eficientemente a la evolución de los requerimientos del cliente o a nuevas perspectivas surgidas durante el desarrollo.

La iteratividad no solo acelera la obtención de resultados tangibles, sino que también promueve una mayor involucración de los stakeholders, quienes pueden proporcionar comentarios valiosos a lo largo de todo el proceso. Este enfoque iterativo e incremental minimiza los riesgos asociados con posibles desviaciones en los objetivos del proyecto y favorece la entrega de un producto final más alineado con las expectativas y necesidades del usuario.

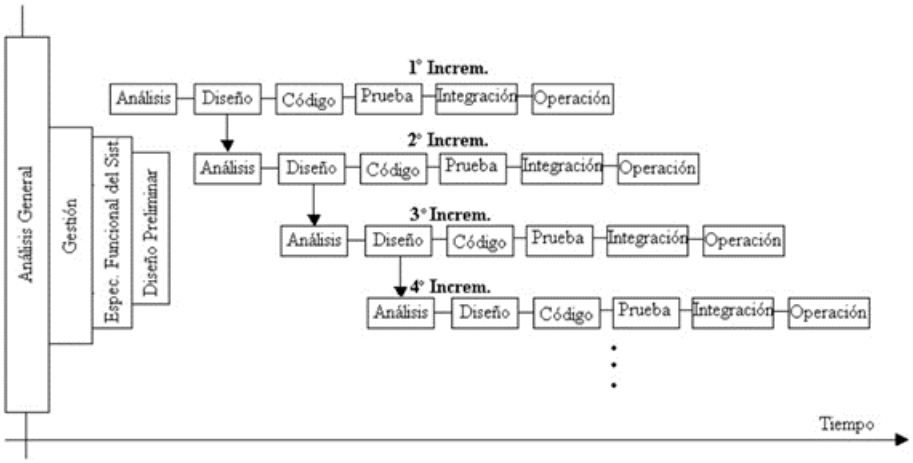

Figura 3.1: Metodología incremental. [25]

En la imagen 3.1, podemos observar **secuencias en cascada retroalimentada** en forma escalonada mientras se avanza en el tiempo. Cada secuencia en cascada provoca un incremento que, a su vez, contiene muchas funciones suplementarias, conocidas o no, sin finalizar. Por simplicidad, en la figura se muestra como secuencial puro.

Bajo este modelo, los **incrementos** son partes funcionales más pequeñas y reutilizables. Cada incremento concluye con la actividad de *operación y mantenimiento* que es donde se produce la entrega del producto parcial al cliente. Hay que tener en cuenta que un incremento no necesariamente se iniciará durante la fase de diseño del anterior, puede ser posterior o incluso producirse antes, en cualquier tiempo de la etapa previa.

Las consecuencias generadas de esta metodología, conllevan una serie de ventajas en el desarrollo, aunque también puede acarrear ciertos inconvenientes.
El modelo es útil para el desarrollo software donde observemos que hay áreas bastante bien definidas a cubrir, con suficiente independencia como para ser implementadas en etapas sucesivas. El enfoque incremental resulta también muy útil cuando se dispone de baja plantilla de personal para llevar a cabo el proyecto o si no hay disponible fecha límite por lo que se entregan versiones parciales que proporcionan al usuario funcionalidad básica.

Esta metodología presenta características que son particularmente apropiadas para el desarrollo de un proyecto como el que se describe. En particular, podemos identificar las siguientes **ventajas** [26] [27]:

- Es apropiado para productos que estén formados por componentes bien definidos e independientes, que permitan su desarrollo sucesivo.
- Permite el desarrollo de productos software con entregas parciales frecuentes incluso con equipos de desarrollo pequeños.

- Reduce el impacto en los costos por que en caso de alterar o rehacer los requisitos, solo afecta una parte del sistema.
- Control y seguimiento, se puede ir probando el software fácilmente durante las diferentes iteraciones.
- Pruebas continuas testables, al entregarse las iteraciones críticas al principio, estas pasan por un mayor número de pruebas, logrando que no se encuentren errores ni fallos en las partes más importantes del software.

Por otro lado, también presenta una serie de **inconvenientes** que deberán ser tomados en cuenta durante el desarrollo de este proyecto, entre los cuales se cuentan:

- No es recomendable para casos de sistemas de alto nivel de seguridad, de procesamiento distribuido, o de alto índice de riesgos.
- No garantiza el éxito del proyecto por sí solo debido a que las iteraciones pueden ser costosas.
- Requiere de un cliente comprometido a lo largo de todo el desarrollo del proyecto.

Como ya hemos comentado, cada etapa está formada por una serie de iteraciones para finalizar el desarrollo del proyecto y entregar un producto completo. Podemos diferenciar **6 fases**:

- **Análisis**: se examinan y comprenden los requisitos y objetivos del proyecto. Se identifican las necesidades del usuario, se definen los alcances y limitaciones, y se establece una base clara para el desarrollo.
- **Diseño**: se enfoca en la creación de una estructura detallada del sistema. Define arquitecturas, elaboración de diagramas y se planifica la interacción entre los componentes teniendo como objetivo una representación visual y funcional del sistema.
- **Desarrollo**: lleva a cabo la codificación del software conforme a los diseños establecidos con especificaciones técnicas y funcionales para el funcionamiento del sistema.
- **Integración de componentes**: combina los componentes desarrollados para formar una versión inicial del sistema completo.
- **Validación y pruebas**: se realizan pruebas para verificar que el sistema cumple con los requisitos establecidos. La validación garantiza la calidad del software y su conformidad con las expectativas del usuario.
- **Documentación**: se completa la documentación técnica y de usuario con manuales, especificaciones técnicas, y cualquier información necesaria para comprender, mantener y utilizar eficientemente el sistema desarrollado.

.

Para el desarrollo de este proyecto se han considerado un total de **10 iteraciones** que vienen detalladas en el siguiente apartado, en la figura 3.2.

3.2. Planificación temporal inicial de proyecto

Para la planificación de este proyecto se han tenido en consideración una serie de aspectos que repercuten de forma directa en la duración y estimación temporal del mismo.

Dentro de las distintas etapas en las que se encuentra dividido el proyecto pueden surgir distintos problemas que afecten considerablemente a la planificación inicial programada. Uno de estos problemas es la gestión de retrasos o avances temporales, el cual provoca un incremento o decremento en la duración del proyecto. Este tipo de problemas pueden originarse debido al tiempo destinado por el equipo de trabajo (más concretamente el desarrollador) al aprendizaje y formación requerido en las distintas herramientas y lenguajes utilizados en el desarrollo del proyecto. Es por ello que el tiempo destinado a las fases de implementación e integración de componentes son especialmente susceptibles a sufrir ciertas modificaciones.

Para intentar paliar este problema, la solución que se ha aplicado es la de sobreestimar la duración de estas tareas para cada una de las etapas, que consigue a su vez establecer y mantener controlada la fecha de finalización de proyecto, lo que proporciona una estimación más precisa en el caso de que este tipo de problema se encontrara presente. Si este periodo de formación y aprendizaje no fuera necesario por parte del equipo de trabajo, la duración del proyecto se vería reducida, estableciendo la fecha de finalización de este antes de lo previsto.

Por otro lado, puede darse la situación de que el aumento de tiempo destinado a estas etapas no sea suficiente para cubrir dichas necesidades de formación y que, a pesar de contar con esa ampliación de tiempo, la duración total del proyecto aumente y la fecha de finalización se retrase a lo estimado. Para solucionar este supuesto y que la duración y finalización del proyecto no se vean modificadas, los miembros del equipo de trabajo que se vean afectados deberán aumentar (de forma extra) el número de horas dedicadas a la tarea en cuestión, para así evitar el retraso que se originaría en las etapas y tareas posteriores del desarrollo.

Una vez establecida esta condición, para este proyecto se ha seguido el criterio de aumentar el tiempo destinado a la realización de las tareas de implementación e integración de componentes, debido a que la formación y el aprendizaje básico del desarrollador no formará parte del proyecto al considerar que este cuenta con experiencia previa. Aun así, y de forma preventiva, se ha tenido en cuenta en la estimación temporal.

3.2.1. Gestión de conflictos temporales

Las estimaciones del tiempo y el esfuerzo que supondría el proceso de aprendizaje de los lenguajes y programas necesarios para llevar a cabo el proyecto fueron insuficientes, lo que provocó retrasos a lo largo de todo el proyecto. Este aprendizaje se basó en la realización de formación adicional para adquirir las competencias necesarias para asegurar la calidad del resultado del proyecto, lo que supuso un tiempo superior al esperado. Además, la necesidad de compaginar la realización del proyecto con la jornada laboral favoreció la aparición de imprevistos que alteraron la planificación inicial en algunos momentos.

3.2.2. Aspectos generales de la planificación inicial

Para la planificación de esta aplicación se han tenido en cuenta una serie de aspectos que repercuten de forma directa con la duración y estimación temporal del proyecto:

- Se consideran los fines de semana y días festivos como laborables, exceptuando algunos de descanso.
- Los días laborales se prevé una dedicación de tres horas diarias.
- La formación y familiarización con las distintas aplicaciones y lenguajes, queda reflejada directamente en las etapas de cada desarrollo.
- En la etapa de desarrollo de la aplicación, así como las pruebas realizadas, han supuesto un aumento en la complejidad del proyecto con la finalidad de satisfacer y cumplir con las necesidades finales del cliente.

Por motivos laborales, decidimos poner el foco en acabarlo en un periodo razonable de un año y tres meses. A continuación, se detalla las etapas que se han seguido para la elaboración del proyecto:

ID	Descripción	Inicio	Fin	Días trabajados
	Inicio proyecto pocketMix	**01/07/2022**	**14/08/2023**	**142**
Etapa 1	Análisis	01/07/2022	18/08/2022	16
Etapa 2	Añadir tarjetas	19/08/2022	17/10/2022	20
Etapa 3	Gestionar tarjetas	18/10/2022	02/12/2022	15
Etapa 4	Información tarjetas	05/12/2022	02/01/2023	11
Etapa 5	Gestión usuarios	03/01/2023	13/02/2023	15
Etapa 6	Sistema puntos	14/02/2023	29/03/2023	16
Etapa 7	Recomendar a un amigo	30/03/2023	25/04/2023	9
Etapa 8	API PoketMix	26/04/2023	14/06/2023	18
Etapa 9	Administración datos	15/06/2023	06/07/2023	8
Etapa 10	Testing	07/07/2023	14/08/2023	14

Tabla 3.1: Etapas de planificación inicial de proyecto

A continuación mostramos el resumen final completo del proyecto con un **diagrama de Gantt [37]** en el que se detallan las distintas etapas.

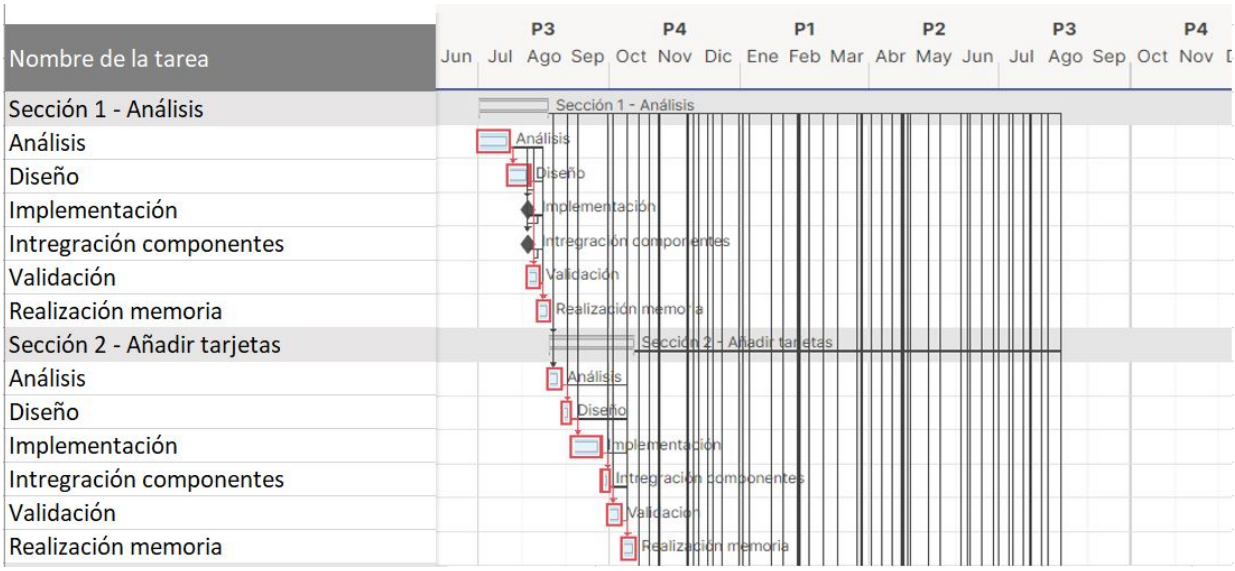

Figura 3.2: Diagrama de Gantt: Analisis - Añadir tarjetas.

Figura 3.3: Diagrama de Gantt: Gestionar tarjetas - Información tarjetas.

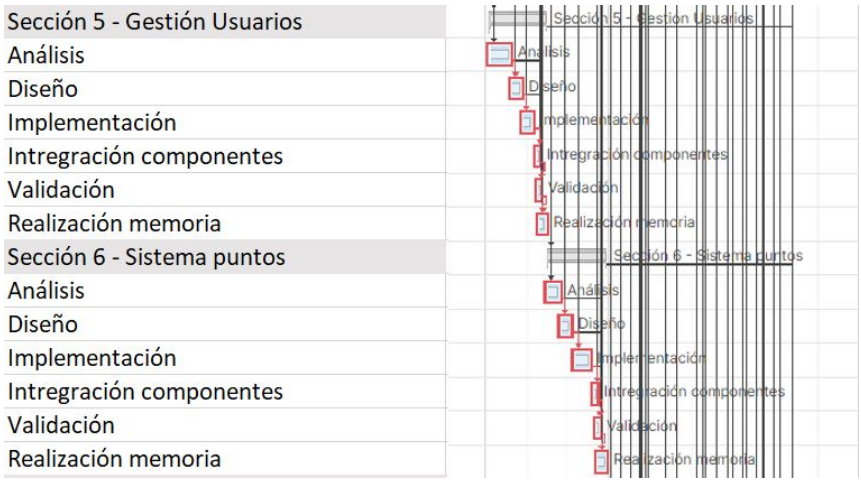

Figura 3.4: Diagrama de Gantt: Gestión usuarios - Sistema puntos.

Figura 3.5: Diagrama de Gantt: Recomendar amigo - API PocketMix.

Figura 3.6: Diagrama de Gantt: Administración datos - Testing.

3.3. Presupuesto económico

En este apartado, vamos a realizar el cálculo preliminar de los presupuestos del desarrollo de proyecto y posterior del mismo.

3.3.1. Presupuesto inicial del proyecto

Llevaremos a cabo los cálculos en función de la planificación temporal descrita en el Diagrama de Gantt.

- Costes del personal

Estos costes hacen alusión a los recursos humanos que se han utilizado durante todo el proceso para llevar a cabo la aplicación. En el caso de nuestro proyecto, durante las diferentes etapas de desarrollo, quien se encarga de gestionar y hacerse cargo de ellas es el mismo creador, aunque desempeñe un rol diferente en cada una de ellas. En la tabla 3.2, mostraremos el tiempo que ha dedicado cada rol en cada etapa correspondiente, el salario base de cada uno de ellos, el tiempo que cobra por hora y finalmente el coste final que nos supone dicho rol en nuestro proyecto:

- **Jefe de Proyecto**: persona encargada de planificar, ejecutar y monitorizar las acciones que forman parte de un proceso. Debe avanzar en la misma línea del proceso que dirige y aplicar en cada una de sus fases los tipos de liderazgo más oportunos.
- **Analista**: persona que analiza los datos y los transforma en información útil para la empresa para tomar decisiones sobre bases más seguras.
- **Diseñador UX/UI**: profesional que busca la forma fácil, intuitiva y amigable de solucionar un problema en función de los gustos, necesidades y opiniones de los usuarios de un producto o servicio. Nos referimos, por tanto, al encargado de diseñar la experiencia de un cliente de manera óptima.

- **Desarrollador/Programador**: crean todo lo que el usuario final ve y con lo que interactúa además de ser los responsables de los procesos y funcionalidades que están detrás.
- **Tester**: persona encargada de planificar, realizar pruebas, identificar errores y generar informes que evalúan el rendimiento general del software.

Roles	Sueldo €	Sueldo € / hora	Tiempo (horas)	Coste total (€)
Jefe de proyecto	3.000 €/mes	24,50 €/h	100	2.450 €
Analista	2.100 €/mes	18,15 €/h	120	2.220 €
Diseñador gráfico	1.500 €/mes	11,46 €/h	130	1.490 €
Programador	2.000 €/mes	15,10 €/h	180	2.718 €
Tester	1.900 €/mes	13,00 €/h	70	910 €
Total	-	-	**600**	**9.788 €**

Tabla 3.2: Costes total inicial del personal.

Para el cálculo de estos costes para la estimación, al desconocer el porcentaje de trabajo realizado por cada uno de los roles, se ha decidido utilizar una tasa de horas de trabajo correspondiente a la media de salario entre todos los miembros de trabajo previamente identificados (véase tabla 3.2).

$$\textbf{Sueldo Medio} = \sum sueldos = 3.000 + 2.100 + 1.500 + 2.000 + 1.900 = 10.500/mes : 5 =$$

$$\textbf{2.100 €/mes = 17,5 €/hora}$$

Teniendo en cuenta que la jornada laboral de los trabajadores, descrita para este proyecto, es de 30 horas semanales.

- Costes hardware

Para realizar el cálculo se ha tenido en cuenta la amortización en base al tiempo de vida útil estimado para cada uno de ellos, junto con su coste de adquisición. Con este procedimiento se obtiene la cuantía real del gasto que supone la utilización de estos componentes en el proyecto. Al ser prácticamente semejantes en duración la planificación y la estimación se ha redondeado a 7 meses el periodo de amortización para ambos cálculos. En la siguiente tabla se pueden reflejar los resultados obtenidos:

Componentes	Precio	Vida útil	Porcentaje uso	Coste real
Ordenador portátil	1.850 €	8 años	60 %	129,50 €
Ratón	25 €	5 años	55 %	2,75 €
Pantalla	180 €	7 años	50 %	14,40 €
Disco duro externo	70 €	6 años	25 %	6,30 €
Dispositivo móvil	620 €	4 años	70 %	89,90 €

Tabla 3.3: Coste total inicial hardware.

Se han tenido en cuenta 7 meses de uso. Esto es, hemos hecho una comparativa con respecto a la vida útil de los componentes y hemos tenido en cuenta cuál sería el porcentaje de uso dentro de nuestro proyecto. Por ejemplo, para el ordenador portátil, al tener una vida útil de 8 años en total, hemos considerado un desgaste del 7 % (al considerar 7 meses de desarrollo).

$$\textbf{Coste hardware total} = \sum componentes = 129,50 + 2,75 + 14,40 + 6,30 + 89,90 = \textbf{242,85 €}$$

- Costes software

Para obtener la cuantía real del gasto que supone la utilización de los componentes software para este proyecto, se ha considerado emplear todo el software gratuito posible para reducir gastos. En la siguiente tabla se reflejan los datos obtenidos:

Componentes	Precio	Vida útil	Porcentaje uso	Coste real
Windows 10 Pro	250 €	5 años	60 %	150 €
Smartsheet	6 €/mes	-	90 %	66 €
Overleaf	Gratuito	-	50 %	0 €
Adobe Reader	Gratuito	-	5 %	0 €
Android Studio	Gratuito	-	80 %	0 €
IntelliJ IDEA	Gratuito	-	50 %	0 €
Visual Studio Code	Gratuito	-	30 %	0 €
Firebase	Gratuito	-	20 %	0 €
Git	Gratuito	-	70 %	0 €
MySQL Workbench	Gratuito	-	25 %	0 €
Draw.io	Gratuito	-	10 %	0 €
Lucidchart	Gratuito	-	10 %	0 €

Tabla 3.4: Costes total inicial software.

$$\textbf{Coste software total} = \sum componentes = 150 + 66 = \textbf{316 €}$$

- Otros costes

Teniendo en cuenta las mismas consideraciones que en los apartados anteriores, se reflejan los gastos secundarios en la siguiente tabla:

Otros gastos	Coste unidad	Coste real
Costes eléctricos	35 €/mes	245 €
Conexión Internet (Fibra 600 MB) + dispositivo móvil	45 €/mes	315 €

Tabla 3.5: Costes total otros gatos.

$$\textbf{Otros costes} = 245 + 315 = \textbf{560 €}$$

3.4. Estimación del esfuerzo

En este apartado llevaremos a cabo la estimación inicial de los costes y duración del proyecto. El objetivo principal es conseguir una reducción de costes, mejorar la calidad y aumentar los niveles de servicio y para ello se van a seguir los distintos modelos que expondremos a continuación: método de Albretch con puntos de función y el método COCOMO.

A continuación, tras realizar las estimaciones mediante los métodos comentados, las compararemos entre sí y más adelante con la planificación obtenida para la exposición del coste real del proyecto.

3.4.1. Estimación por puntos de función: Método Albretch

El método de Albrecht consiste en hacer el cálculo de los puntos de función de un sistema realizando tres estapas:

1. Identificación de los componentes necesarios para el cálculo.
2. Calcular los **puntos de función no ajustados** (PFNA).
3. Ajustar los puntos de función mediante el cálculo del **factor de ajuste** (FA).

En primer lugar, se enumeran todos los componentes de cada tipo (entradas externas, salidas externas, ficheros lógicos de datos internos, ficheros lógicos de datos de interfaz y consultas externas); seguidamente, se evalúa individualmente la complejidad de cada uno de ellos, utilizando unas tablas ya establecidas que proporcionan el factor de complejidad de cada componente individual (**ALTO, MEDIO, BAJO**).

3.4.2. Entradas de usuario

Son todos los datos de control de usuario que entran en la aplicación y añaden o cambian información en un fichero lógico de datos interno.

Descripción	Complejidad
Registro nuevo usuario	Baja
Modificar perfil	Baja
Login	Baja
Buscador de tarjetas por nombre	Baja
Añadir nuevas tarjetas de fidelización	Media
Modificar tarjeta	Baja
Seleccionar tarjeta favorita	Baja
Introducir tarjeta manualmente	Baja
Introducir pin bloqueo	Baja
Añadir puntos	Alta

Tabla 3.6: Entradas de usuario

3.4.3. Salidas externas

Son todos aquellos ficheros lógicos de datos de control de usuario que salen de la aplicación.

Descripción	Complejidad
Información de la cuenta	Baja
Información tarjeta	Baja
Información de la aplicación	Baja
Información puntos	Baja
Listado tarjetas favoritas	Baja
Listado ordenación tarjetas	Baja
Notificación gestión perfil	Baja
Notificación gestión tarjetas	Baja
Notificación de añadir manualmente tarjetas	Baja
Código QR	Baja
Código de barras	Baja
Correo electrónico (soporte y recomendar a)	Baja

Tabla 3.7: Salidas externas

3.4.4. Consultas externas

Son entradas de usuario u otra aplicación que generan una salida inmediata. Son consecuencia de una búsqueda y no una actualización de un fichero lógico de datos interno.

Descripción	Complejidad
Consultar tarjetas favoritas	Baja
Consultar puntos	Baja
Consultar listado preferencias	Baja
Consultar perfil	Baja
Consultar detalles tarjetas	Baja
Eliminación contenido cuenta	Baja
Eliminación tarjeta	Baja
API PoketMix	Media

Tabla 3.8: Consultas externas

3.4.5. Ficheros lógicos internos

Son aquellos ficheros lógicos de datos o información de control interna que se generan, son usados y mantiene la aplicación.

Descripción	Complejidad
Preferencias del usuario	Baja
Formulario de registro	Baja

Tabla 3.9: Ficheros lógicos internos

3.4.6. Ficheros lógicos externos

Son aquellos ficheros lógicos de datos compartidos con otra aplicación, recibidos o enviados a ella.

Descripción	Complejidad
Firebase	Media
API PoketMix	Media

Tabla 3.10: Ficheros lógicos externos

Una vez establecidas las complejidades de cada sistema, calculamos los puntos de función sin ajustar (PFNA) en base a la siguiente tabla:

Dominio	Complejidad	Peso por complejidad	Nº total de funciones	Total
Entradas de usuario	Baja	x3	8	24
	Media	x4	1	4
	Alta	x6	1	6
Salidas externas	Baja	x4	12	48
	Media	x5	0	0
	Alta	x7	0	0
Consultas externas	Baja	x3	7	21
	Media	x4	1	4
	Alta	x6	0	0
Ficheros lógicos internos	Baja	x7	2	14
	Media	x10	0	0
	Alta	x15	0	0
Ficheros lógicos externos	Baja	x5	0	0
	Media	x7	2	14
	Alta	x10	0	0
Puntos de función sin ajustar (PFNA)				**135**

Tabla 3.11: Equivalencia de los PFNA

Una vez definidos los puntos de función sin ajustar, calcularemos el factor de ajuste que se obtiene a partir de asignar un grado de complejidad del 0 al 5, donde 0 es el mínimo índice y 5 es el máximo.

Factores de complejidad	0-5
Comunicación de datos	4
Prestaciones (Rendimiento)	3
Velocidad (Frecuencia) de las transacciones	3
Diseño de la eficiencia del usuario final	4
Complejidad del proceso lógico interno	3
Facilidad de instalación	1
Localizaciones múltiples	1
Funciones distribuidas	3
Gran uso de la configuración	3
Entrada online de datos	4
Actualizaciones online de datos	3
Reusabilidad	1
Facilidad de operación	3
Facilidad de cambio	2
Factor de Complejidad Total	**38**

Tabla 3.12: Ficheros lógicos externos

A continuación calculamos el factor de ajuste (FA) aplicando la siguiente fórmula que se basa en la suma total de los factores de complejidad (FC):

$$\textbf{FA} = 0,65 + (0,01 * FC) = 0,65 + (0,01 * 38) = \textbf{1,03}$$

A partir de este valor, se calculan los puntos de función ajustados (PFA):

$$\textbf{PFA} = PFNA * FA = 135 * 1,03 = \textbf{139}$$

Por último, para finalizar la estimación calculamos la duración del proyecto junto con el esfuerzo personal en horas. Establecemos la equivalencia puntos de horas por puntos de función contando con que 1 mes de esfuerzo (21 días laborables) equivale a 13 puntos de función.

$$\textbf{Duración de proyecto estimada} = 139PFA/13 = \textbf{10 meses}$$

3.5. Estimación por COCOMO

El Modelo Constructivo de Costes es una jerarquía de modelos de estimación de costes software que incluye submodelos **básico, intermedio y avanzado**. Obtendremos el esfuerzo y el tiempo empleado en el desarrollo del sistema en función de las líneas de código que hemos calculado en la estimación por puntos de función.

De las tres subcategorías de desarrollo en este modelo, hemos optado por el intermedio ya que tiene en cuenta el entorno de trabajo, tiene una especificación de requisitos completa y calcula el esfuerzo que se emplea en desarrollar el proyecto en función del tamaño de las propiedades y teniendo en cuenta un conjunto de factores.

Lenguaje	Número de líneas
Ensamblador	320
Macroensamblador	213
C	150
Fortran	106
Cobol	106
Pascal	91
Cobol ANSI	91
Basic	91
RPG	80
PL / I	80
Ada	71
Basic ANSI/Quick/Turbo	64
Java	53
Visual C++	34
Foxpro 2,5	34
Visual Bsic	32
Delphi	29
C++	29
Visual Cobol	20
Clipper	19
Power builder	16
Hoja de Cálculo	6

Tabla 3.13: Relación líneas de código / Puntos de función.

En la tabla no aparece reflejado el lenguaje Kotlin, pero como este está basado en el lenguaje Java utilizamos su valor correspondiente.

$$\textbf{LDC} = PFA * (Lineas/PF) = 160,6 * 53 = \textbf{8.511,8}$$

Una vez obtenidas las líneas de código y teniendo en cuenta que el desarrollo de la aplicación se realiza en un entorno constante con una estimación por debajo de las 50 KLDC, utilizaremos el modelo orgánico de estimación ya que es el más apropiado. Para ello, es necesario aplicar un factor de esfuerzo basado en 15 características.

CONDUCTORES DE COSTE	VALORACIÓN					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extr. alto
Fiabilidad requerida del software	0,75	0,88		1,15	1,40	-
Tamaño de la base de datos	-	0,94	1,00	1,08	1,16	-
Complejidad del producto	0,70	0,85	1,00	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	1,00	1,11	1,30	1,66
Restricciones del almacenamiento principal	-	-	1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	1,00	1,15	1,30	-
Tiempo de respuesta del ordenador	-	0,87	1,00	1,07	1,15	-
Capacidad del analista	1,46	1,19	1,00	0,86	0,71	-
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	-
Capacidad de los programadores	1,42	1,17	1,00	0,86	0,70	-
Experiencia en S.O. utilizado	1,21	1,10	1,00	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1,00	0,95	-	-
Prácticas de programación modernas	1,24	1,10	1,00	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1,00	0,91	0,83	-
Limitaciones de planificación del proyecto	1,23	1,08	1,00	1,04	1,10	-

Tabla 3.14: Ponderación factores de esfuerzo

Seguidamente veremos el factor aplicado a cada una de ellas:

	Valores de los factores	
Fiabilidad Software	Alto	1,15
Tamaño de la BDD	Nominal	1
Complejidad del software	Nominal	1
Restricciones en tiempo de ejecución	Alto	1,11
Restricciones de almacenamiento principal	Nominal	1
Volatilidad de la máquina virtual	Nominal	1
Tiempo de respuesta del ordenador	Alto	1,07
Calidad del analista	Alto	0,86
Experiencia en la aplicación	Nominal	1
Capacidad de los programadores	Muy Alto	0,7
Experiencia en el sistema operativo utilizado	Nominal	1
Experiencia en el lenguaje de programación	Alto	0,95
Prácticas de programación modernas	Nominal	1
Utilización de herramientas software	Alto	0,91
Limitaciones de planificación del proyecto	Bajo	1,08

Tabla 3.15: Relación factores de coste genérico

Para obtener la variable m_x, que es un un multiplicador dependiente de los 15 atributos anteriores, haremos el producto de los valores de los factores de la tabla que acabamos de ver para obtener la estimación del tiempo de desarrollo y el esfuerzo del proyecto.

	Básico		Intermedio	
Modo	a_i	b_i	a_i	b_i
Orgánico	2.4	1.05	3.2	1.05
Semiencajado	3.0	1.12	3.0	1.12
Empotrado	3.6	1.2	2.8	1.2

Figura 3.7: Modelos de desarrollo COCOMO Intermedio.

$$m_x = 1,15 * 1,11 * 1,07 * 0,86 * 0,7 * 0,95 * 0,91 * 1,08 = \mathbf{0,7676}$$

$$E = a_i S^b m_x = 3,2 * 9.094,8^{1,05} * 0,7676 = \mathbf{24,944\ persona/mes}$$

$$T_{DEV} = c * E^d = 2,5 * E^{0,38} = \mathbf{8,8\ meses}$$

Para finalizar, calcularemos el esfuerzo nominal (N) requerido para poder llevar a cabo el proyecto:

$$N = E/T_{DEV} = 24,944/8,8 = \mathbf{3,118}$$

Conclusión: Se requiere de tres personas a jornada completa para desempeñar el desarrollo del proyecto en 8,8 meses.

3.6. Comparativas

Tras calcular las distintas estimaciones, realizaremos una comparativa entre ellas para analizar cual de ellas es la que mejor se ajusta a nuestro proyecto:

Modelo estimación	Personas	Tiempo
Puntos de función	1	10 meses
COCOMO	3	9 meses

Tabla 3.16: Comparativa de estimaciones (simulada).

Teniendo en cuenta que el proyecto ha sido realizado por una única persona, la traducción final de los datos obtenidos en la tabla de arriba serían los siguientes:

Modelo estimación	Personas	Tiempo
Puntos de función	1	10 meses
COCOMO	1	27 meses

Tabla 3.17: Comparativa de estimaciones (real).

Analizando los resultados y comparándolo con la planificación temporal obtenida del siguiente apartado, la que mejor se ajusta y más precisa es la estimación del **método de Albretch** ya que, considerando 90 horas de trabajo mensual, nos saldría:

$$\mathbf{Horas\ trabajadas} = 10 meses * 90 h/mes = \mathbf{900\ horas}$$

$$\mathbf{Coste\ personal} = 900 horas * 17,5 euros/h = \mathbf{15.750\ €}$$

3.7. Presupuesto final

Modelo	Tiempo de vida	Costes	Coste final
Planificación inicial	7 meses	▪ Coste personal = 9.788 € ▪ Coste hardware = 242,85 € ▪ Coste software = 316 € ▪ Otros gastos = 560 €	**10.906,85 €**
Estimación puntos función	10 meses	▪ Coste personal = 15.750 € ▪ Coste hardware = 242,85 € ▪ Coste software = 316 € ▪ Otros gastos = 560 €	**16.870 €**

Tabla 3.18: Presupuesto final

3.8. Balance temporal y económico

En esta parte, se realiza una evaluación integral del proyecto, que implica analizar los contrastes temporales y económicos entre lo que realmente sucede y lo que se había planificado. La evaluación temporal se lleva a cabo al concluir cada iteración, mientras que la evaluación económica se realiza de manera global al finalizar el proyecto.

3.8.1. Balance temporal

Para este proyecto, se han realizado 10 etapas:

- Primera *Etapa* - **Análisis**: 1 jul - 18 ago con un total de 16 días trabajados.
- Segunda *Etapa* - **Añadir tarjetas**: 19 ago - 17 nov con un total de 20 días trabajados.
- Tercera *Etapa* - **Gestionar tarjetas**: 18 nov - 2 dic con un total de 15 días trabajados.
- Cuarta *Etapa* - **Información tarjetas**: 5 dic - 2 ene con un total de 11 días trabajados.
- Quinta *Etapa* - **Gestión usuarios**: 3 ene - 13 feb con un total de 15 días trabajados.
- Sexta *Etapa* - **Sistema puntos**: 14 feb - 29 mar con un total de 16 días trabajados.
- Séptima *Etapa* - **Recomendación**: 30 mar - 25 abr con un total de 9 días trabajados.
- Octava *Etapa* - **API PoketMix**: 26 abr - 14 jun con un total de 18 días trabajados.
- Novena *Etapa* - **Administración datos**: 15 jun - 6 jul con un total de 8 días trabajados.
- Décima *Etapa* - **Testing** : 7 jul - 14 ago con un total de 14 días trabajados.

.

En este apartado, presentamos el rendimiento temporal efectivo del estudiante encargado de lograr el producto objetivo de este proyecto en cada una de las tareas planificadas para cada iteración. Esto permitirá visualizar posibles retrasos y obstáculos que hayan impactado el progreso y la secuencia normal de cada etapa.

El retraso significativo observado en las primeras etapas y durante la implementación del proyecto se debe principalmente a dos factores clave. En primer lugar, la necesidad de participar en una formación continua ha consumido parte del tiempo planificado, ya que implica dedicar recursos temporales al desarrollo de habilidades y conocimientos específicos para poder desarrollar el proyecto. En segundo lugar, el hecho de haber estado trabajando simultáneamente ha generado ciertos obstáculos, ya que la atención y el esfuerzo se han dividido entre las responsabilidades laborales y la ejecución del proyecto, afectando así la fluidez y la rapidez en el avance planificado.

Además, es importante señalar que se han registrado demoras adicionales debido a la toma de vacaciones durante periodos claves, como Semana Santa, el mes de agosto, las festividades navideñas y otros festivos predeterminados. Estas interrupciones vacacionales han contribuido a la extensión de los plazos previstos, ya que durante estos períodos se ha reducido la disponibilidad de recursos humanos, lo que ha afectado directamente la continuidad y eficiencia de las tareas planificadas. La combinación de la formación continua, el trabajo concurrente y las pausas vacacionales ha generado un contexto complejo que ha impactado notablemente en el cronograma y desarrollo del proyecto.

3.8.2. Balance económico

Los gastos iniciales de software permanecen sin alteraciones, dado que se emplean herramientas de software de código abierto.

Como indicamos en la revisión temporal, la duración del proyecto se ha extendido notablemente, principalmente debido a la participación en formación continua y al hecho de trabajar simultáneamente en otras responsabilidades. Esta prolongación del tiempo de ejecución ha conllevado ajustes en el presupuesto destinado a hardware y en los costos asociados al personal, ya que implica la incorporación de recursos adicionales durante un periodo extra de trabajo, excluyendo los días libres correspondientes a festividades y vacaciones.

Por tanto, la tablas correspondiente al personal 3.2, hardware 3.3 y otros gastos 3.5, se ve alterada en el presupuesto inicial planificado, lo que supone una variación de cuatro meses más.

Roles	Sueldo €	Sueldo € / hora	Tiempo (horas)	Coste total (€)
Jefe de proyecto	3.000 €/mes	24,50 €/h	150	3.675 €
Analista	2.100 €/mes	18,15 €/h	180	3.267 €
Diseñador gráfico	1.500 €/mes	11,46 €/h	200	2.292 €
Programador	2.000 €/mes	15,10 €/h	250	3.775 €
Tester	1.900 €/mes	13,00 €/h	100	1.300 €
Total	-	-	**880**	**14.309 €**

Tabla 3.19: Balance económico del personal.

Componentes	Precio	Vida útil	Porcentaje uso	Coste real
Ordenador portátil	1.850 €	8 años	60 %	212 €
Ratón	25 €	5 años	55 %	4,58 €
Pantalla	180 €	7 años	50 %	23,57 €
Disco duro externo	70 €	6 años	25 %	2 €
Dispositivo móvil	620 €	4 años	70 %	142 €
Total	-	-	-	**384,15**

Tabla 3.20: Balance económico hardware.

Otros gastos	Coste unidad	Coste real
Costes eléctricos	35 €/mes	385 €
Conexión Internet (Fibra 600 MB) + dispositivo móvil	45 €/mes	495 €
Total	-	**880**

Tabla 3.21: Balance económico otros gastos.

A continuación, en la siguiente tabla, vemos cuál ha sido el total del balance económico:

Modelo	Tiempo de vida	Costes	Coste final
Balance económico	11 meses	▪ Coste personal = 14.309 € ▪ Coste hardware = 384,15 € ▪ Coste software = 316 € ▪ Otros gastos = 880 €	**15.889,15 €**

Tabla 3.22: Balance económico final

Por tanto, globalmente, la diferencia entre la planificación inicial y la el coste real ha sido:

$$\textbf{Diferencia coste} = 15.889,15 - 10.906,85 = 4.982,30$$

Capítulo 4

Análisis del sistema

En este bloque se detallarán los diferentes aspectos que han sido considerado para realizar el diseño y desarrollo del sistema, desde todos los requisitos que satisfacen el sistema, junto con las especificaciones y diagramas que los representan.

4.1. Actores del sistema

Los actores son todos aquellos roles que interactúan con el sistema, pueden ser tanto personas físicas como sistemas software externos al nuestro. Para este proyecto se han localizado lo siguientes:

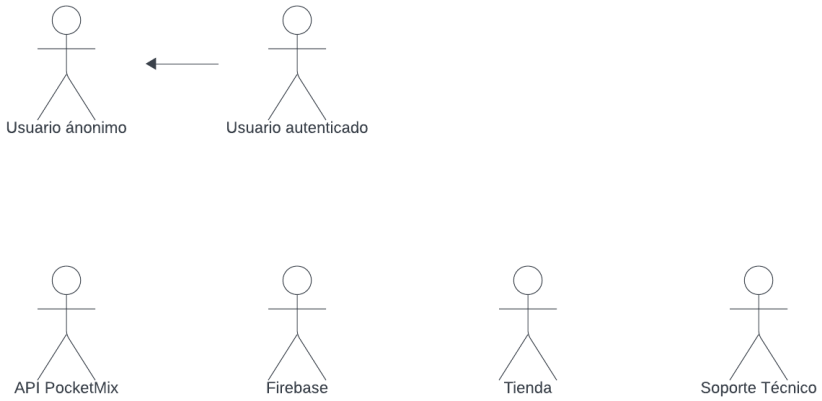

Figura 4.1: Actores del sistema

.

ID	Nombre	Descripción
ACT_01	Usuario anónimo	Usuario que accede a nuestra aplicación sin registrarse previamente.
ACT_02	Usuario Autenticado	Usuario que inicia sesión en nuestra aplicación.
ACT_03	API Pocket Mix	Servicio externo que nos proporciona toda la información necesaria de las tarjetas de fidelización de cada usuario autenticado.
ACT_04	Firebase	Servicio en la nube donde se almacenan las credenciales de los usuarios registrados en nuestra aplicación.
ACT_05	Tienda	Servicio que se encarga de introducir y actualizar los puntos desde la web para las tarjetas.
ACT_06	Soporte técnico	Servicio que realiza una simulación de ayuda para gestionar incidencias realizadas con la aplicación.

Tabla 4.1: Descripción de los actores del sistema

4.2. Requisitos de usuario

Son todas aquellas características y funcionalidades que deberán satisfacer las necesidades del usuario final. Para ello, los organizaremos en función del actor que interviene con la aplicación, organizando los requisitos de la siguiente forma:

Usuarios anónimos		
ID	Nombre	Descripción
RU_01	Registrar	El usuario anónimo se registra en nuestra aplicación.
RU_02	Recomendar	El usuario anónimo recibe enlace de la invitación de un amigo.

Tabla 4.2: Requisitos de usuario pertenecientes a los usuarios no registrados

Usuarios autenticados		
ID	Nombre	Descripción
RU_03	Iniciar sesión	El usuario identificado inicia sesión en nuestra aplicación.
RU_04	Cerrar sesión	El usuario autenticado puede cerrar la sesión en la aplicación.
RU_05	Consultar perfil	El usuario autenticado puede consultar la información de su cuenta.
RU_06	Modificar perfil	El usuario autenticado puede modificar la información de su perfil como, por ejemplo: foto, nombre...
RU_07	Eliminar perfil	El usuario autenticado puede eliminar su cuenta de nuestra aplicación.
RU_08	Añadir tarjeta	El usuario autenticado puede registrar una tarjeta mediante tres formas; escaneando, adjuntando desde una foto o manualmente.
RU_09	Modificar tarjeta	El usuario autenticado puede modificar una tarjeta que haya introducido previamente.
RU_10	Eliminar tarjeta	El usuario autenticado puede eliminar una tarjeta añadida en su listado de tarjetas disponible.
RU_11	Consultar tarjeta	El usuario autenticado puede consultar la información de cada tarjeta, así como el historial de puntos de cada una de ellas.
RU_12	Consultar puntos	El usuario autenticado puede consultar los puntos acumulados en la aplicación de su tarjeta de fidelización.
RU_13	Añadir tarjeta favorita	El usuario autenticado puede añadir tarjetas a favoritos.
RU_14	Consultar tarjetas favoritas	El usuario autenticado puede consultar sus tarjetas favoritas.
RU_15	Visualizar tarjetas	El usuario autenticado puede visualizar el listado de todas sus tarjetas guardadas, tanto favoritas como no.
RU_16	Modificar contraseña	El usuario autenticado puede modificar su contraseña.
RU_17	Habilitar pin	El usuario autenticado puede activar un código de bloqueo para proporcionar seguridad de la aplicación.
RU_18	Ordenar tarjetas	El usuario autenticado puede ordenar sus tarjetas de diferentes formas: más utilizada, añadida recientemente, utilizada recientemente o por orden alfabético.
RU_19	Buscar tarjeta	El usuario autenticado puede buscar una tarjeta mediante la barra búsqueda introduciendo su nombre.
RU_20	Mostrar nombres	El usuario autenticado puede habilitar que aparezca o no el nombre de la tarjeta.
RU_21	Aceptar términos y condiciones	El usuario autenticado para utilizar la aplicación, debe aceptar los términos y condiciones descritos.
RU_22	Atención al cliente	El usuario autenticado puede ver información relacionada con la tienda: teléfono, email, contacto.
RU_23	Consultar manual ayuda	El usuario autenticado puede consultar el manual de ayuda de la aplicación.
RU_24	Recomendar a un amigo	El usuario autenticado puede recomendar la aplicación a un amigo.
RU_25	Activar luminosidad automática	El usuario autenticado puede activar la luminosidad automática para facilitar la lectura al pasar por caja.
RU_26	Gestionar permisos	El usuario autenticado puede configurar la aplicación para controlar los permisos de las notificaciones y del acceso a la cámara.
RU_27	Contactar con soporte	El usuario autenticado puede reportar una incidencia al soporte técnico.
RU_28	Personalizar vista	El usuario puede configurar la vista de las tarjetas.
RU_29	Consultar información acerca de PocketMix	El usuario puede consultar información de la aplicación.

Tabla 4.3: Requisitos de usuario pertenecientes a los usuarios registrados

Usuarios tienda		
ID	Nombre	Descripción
RU_30	Añadir/actualizar puntos.	El usuario puede añadir y actualizar los puntos de las tarjetas de los usuarios fidelizados.

Tabla 4.4: Requisitos de usuario pertenecientes a los usuarios tienda

4.2.1. Modelo de Casos de Uso

A continuación, vamos a representar los requisitos de usuario anteriormente descritos con la ayuda del diagrama de casos de uso. Éstos pretenden describir la interacción del usuario con nuestra aplicación, además de representar los procesos de intercambio de información.

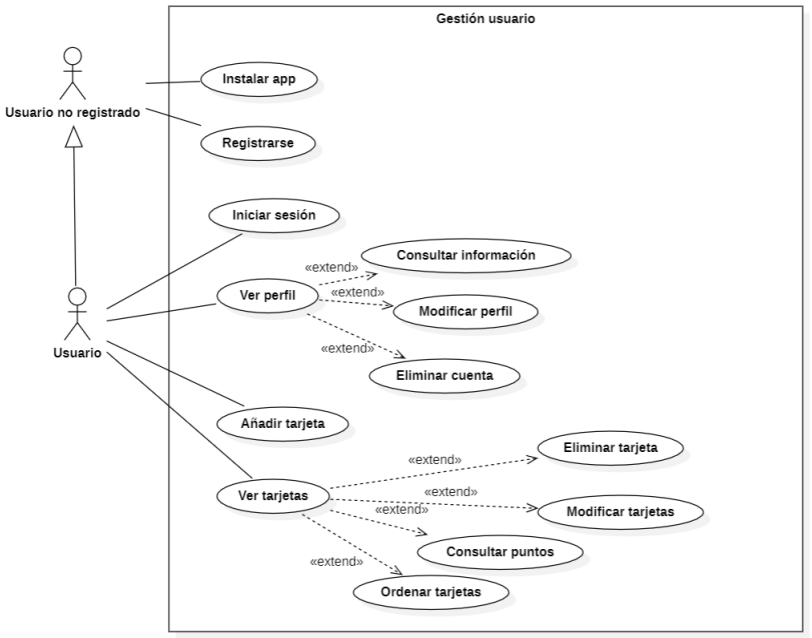

Figura 4.2: Casos de uso - gestión usuarios.

En el anterior diagrama podemos observar la gestión del usuario que hace referencia tanto al inicio de sesión como al registro en la aplicación. Se muestra una visión genérica que va a ser detallada y completa en el siguiente diagrama.

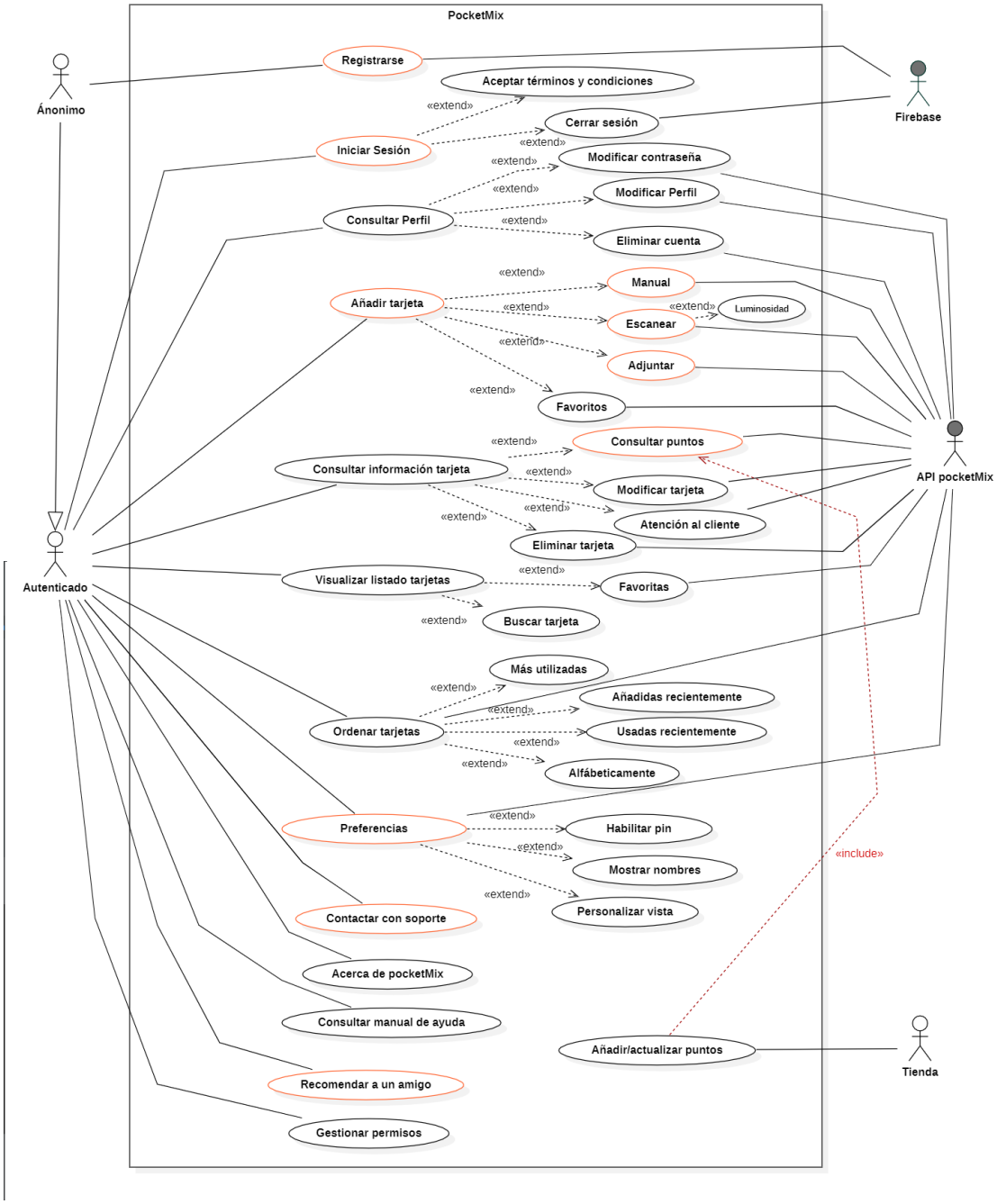

Figura 4.3: Casos de uso - diagrama genérico.

Con estos diagramas de la primera toma de contacto de los usuarios con la aplicación, pasaremos a detallarlos en profundidad.

4.2.2. Especificación de Casos de Uso

Una vez expuestos los casos de uso, para no extender innecesariamente la memoria técnica del proyecto, se muestra el análisis detallado de alguno de los casos de uso identificados más importantes para comprender con mayor precisión el funcionamiento de nuestra aplicación.

Identificador	**CU_01 - Registrarse**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	Firebase Auth
Obj. Asociados		Req. Asociados	RU_01, RU_02
Descripción	El usuario puede crear una nueva cuenta tanto con un email normal o a través del correo de Google para darse de alta en nuestra plataforma.		
Precondiciones	1. El usuario debe disponer de un dispositivo Android y conexión a internet.		
Secuencia Normal	1. El usuario selecciona con qué método quiere crear una cuenta nueva: email o servicios de Google. 2. El sistema muestra al usuario los campos que debe completar del formulario de registro. 3. El usuario rellena los campos requeridos con sus datos y los envía al sistema. 4. El sistema envía los datos introducidos por el usuario al servidor de Firebase. 5. El servidor comprueba que los datos sean correctos y los registra. 6. El sistema genera una nueva cuenta de usuario.		
Secuencia Alternativa	El usuario puede seguir los pasos mencionados con anterioridad utilizando los servicios de Google para registrase.		
Excepciones en la secuencia	▪ Ya existe una cuenta de usuario similar, el sistema te redirige al paso 2 de la secuencia normal. ▪ Los datos de acceso sean erroneos, el sistema te redirige al paso 2 de la secuencia normal .		
Postcondiciones	El sistema envía correctamente la información del registro al servidor de Firebase.		
Importancia	Alta	Frecuencia	Alta

Tabla 4.5: Casos de Uso perteneciente a Registrarse

Identificador	**CU_03 - Iniciar Sesión**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	Firebase Auth
Obj. Asociados		Req. Asociados	RU_01
Descripción	El usuario registrado se autentica en el sistema a través de su email y su contraseña.		
Precondiciones	1. El usuario debe estar registrado previamente en el sistema. 2. El usuario debe disponer de una conexión a internet para que pueda iniciar sesión en nuestro sistema.		
Secuencia Normal	1. El usuario selecciona con que método quiere iniciar sesión si con el email o con los servicios de Google. 2. El sistema muestra al usuario por pantalla el formulario para que introduza sus credenciales. 3. El usuario rellena los campos requeridos con sus credenciales. 4. El sistema envía los datos introducidos por el usuario al servidor de Firebase. 5. El servidor comprueba la existencia de ese usuario y devuelve la respuesta al sistema para que permita establecer la conexión. 6. El sistema redirige al usuario a la ventana principal de la aplicación.		
Secuencia Alternativa	El usuario puede seguir los pasos mencionados con anterioridad utilizando los servicios de Google para iniciar sesión.		
Excepciones en la secuencia	▪ 4.1. En caso de que el email introducido por el usuario no se encuentre registrado en el sistema, se pide al usuario que introduzca un email válido. ▪ 4.2. En caso de que la contraseña introducida por el usuario se pide que vuelva a introducir de nuevo la contraseña válida.		
Postcondiciones	El usuario tiene acceso a su cuenta registrada en nuestra aplicación.		
Importancia	Alta	Frecuencia	Alta

Tabla 4.6: Casos de Uso perteneciente a Iniciar Sesión

Identificador	**CU_08 - Añadir tarjeta**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	API externa PocketMix
Obj. Asociados		Req. Asociados	RU_08, RU_13
Descripción	El usuario puede añadir una tarjeta de diferentes maneras; escaneando, adjuntando desde una foto o manualmente.		
Precondiciones	El usuario debe aceptar los permisos para que la aplicación pueda acceder a su cámara y a su galeria de fotos.		
Secuencia Normal	1. El usuario selecciona añadir tarjeta desde su pagina principal. 2. El sistema le redirige a una nueva ventana y muestra las opciones a elegir por el usuario para introducir la tarjeta; ya sea escanear, adjuntar desde una foto o manualmente. 3. El usuario elige la opción de entre las ofrecidas por el sistema y en función de eso hará lo siguiente: 4. Escanear tarjeta: ▪ El usuario selecciona escanear código y puede habilitar la opción de activar la luminosidad para obtener un escaneo más óptimo. ▪ El sistema envía un mensaje para que el usuario acepte los permisos para acceder a la cámara. ▪ El usuario acepta los permisos. ▪ El sistema habilita el escaner. ▪ El usuario escanea la tarjeta. ▪ El sistema recoge el escaneo y lo envía al servidor PocketMix para que este se encargue de almacenarlo. ▪ El sistema muestra un mensaje al usuario por pantalla "tarjeta añadida correctamente", y muestra en la ventana principal la tarjeta añadida por el usuario.		

Continúa en la página siguiente.

Identificador	**CU_08 - Añadir tarjeta**
Secuencia Normal	5. Adjuntar tarjeta desde foto: ▪ El usuario selecciona adjuntar desde fotografía. ▪ El sistema le solicita permisos al usuario para acceder a su galería. ▪ El usuario acepta los permisos. ▪ El sistema abre la galería de fotos del usuario. ▪ El usuario selecciona la foto de la tarjeta a importar. ▪ El sistema recibe la foto seleccionada por el usuario y la envía al servidor PocketMix para que lo almacene. ▪ El sistema muestra mensaje al usuario "Tarjeta añadida correctamente". ▪ El sistema muestra la tarjeta adjuntada en el listado de tarjetas que tiene el usuario en la ventana principal. 6. Introducir manualmente: ▪ El usuario selecciona introducir manualmente la tarjeta. ▪ El sistema le redirige a una nueva ventana y le muestra el campo para que seleccione el tipo de código que va a introducir en el siguiente campo; QR o Código de barras, junto con el campo introducir código y el nombre del establecimiento. ▪ El usuario introduce los campos solicitados. ▪ El sistema válida los campos y los envía al servidor PocketMix para que sean almacenados. ▪ El sistema muestra mensaje por pantalla al usuario "Tarjeta añadida correctamente". ▪ el sistema muestra la tarjeta en el listado de la ventana principal del usuario.
Secuencia Alternativa	

Continúa en la página siguiente.

Identificador	**CU_08 - Añadir tarjeta**		
Excepciones en la secuencia	▪ 4.1. Error al abrir el escaner, el sistema vuelve a solicitar al usuario que acepte los permisos. ▪ 4.2. Error al intentar abrir la galería de fotos e importar la imagen. ▪ 4.3. Error al no seleccionar un tipo de código y al no introducir un valor válido en el campo número código barras.		
Postcondiciones	El sistema añade correctamente la tarjeta en la lista del usuario.		
Importancia	Alta	Frecuencia	Alta

Tabla 4.7: Caso de uso perteneciente a Añadir Tarjeta

Identificador	**CU_12 - Consultar Puntos**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	API externa PocketMix
Obj. Asociados		Req. Asociados	RU_08, RU_11, RU_12
Descripción	El usuario accede a través de la información de las tarjetas para consultar sus puntos.		
Precondiciones	El usuario debe tener una tarjeta registrada en la aplicación y haberla utilizado previamente en diferentes comercios para que se actualice el historial de puntos.		
Secuencia Normal	1. El usuario selecciona una tarjeta desde la página principal. 2. El sistema le redirige a una nueva ventana donde muestra la información de la tarjeta y las opciones a elegir por el usuario. 3. El usuario elige la opción puntos de entre las ofrecidas por el sistema. 4. El sistema le redirige a una nueva ventana que es el historial de puntos de la tarjeta: ubicación, fecha y la cantidad que ha ganado o perdido.		
Secuencia Alternativa			

Continúa en la página siguiente.

Identificador	**CU_12 - Consultar Puntos**		
Excepciones en la secuencia			
Postcondiciones			
Importancia	Media	Frecuencia	Media

Tabla 4.8: Caso de uso perteneciente a Consultar Puntos

Identificador	**CU_20 - Preferencias - Mostrar nombres**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	
Obj. Asociados		Req. Asociados	RU_20
Descripción	El usuario puede configurar la previsualización de sus tarjetas de la ventana principal.		
Precondiciones	El usuario debe tener un listado de tarjetas en la aplicación.		
Secuencia Normal	1. El usuario selecciona la opción de preferencias ubicada en el menú de su cuenta. 2. El sistema le redirige a una nueva ventana y muestra la opción mostrar una o dos columnas. 3. El usuario selecciona la opción mostrar en una o dos columnas la vista de las tarjetas en función de sus preferencias. 4. El usuario vuelve a la página principal para ver cómo ha quedado la configuración personalizada de las tarjetas.		
Secuencia Alternativa	-		
Excepciones en la secuencia			
Postcondiciones	El sistema actualiza correctamente los parámetros elegidos por el usuario.		
Importancia	Baja	Frecuencia	Baja

Tabla 4.9: Caso de uso perteneciente a Mostrar nombres

Identificador	**CU_24 - Recomendar a un amigo**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	API PoketMix externa
Obj. Asociados		Req. Asociados	RU_01, RU_02, RU_03
Descripción	El usuario puede compartir la aplicación con su amigos a través de un enlace de descarga.		
Precondiciones	▪ El usuario autenticado debe tener instalada nuestra aplicación en su dispositivo. ▪ El usuario no registrado debe disponer de un dispositivo Android.		
Secuencia Normal	1. El usuario selecciona la opción recomendar a un amigo ubicada en el menú cuenta. 2. El usuario le habilita la opción por la que quiere compartir el enlace de descarga. 3. El usuario elige una de esas opciones. 4. El sistema envía el enlace al destinatario.		
Secuencia Alternativa			
Excepciones en la secuencia	Error al enviar el enlace. El sistema te pide repetir los pasos de la secuencia normal.		
Postcondiciones	El sistema envía correctamente el enlace de descarga al destinatario.		
Importancia	Baja	Frecuencia	Baja

Tabla 4.10: Caso de uso perteneciente a Recomendar a un amigo

Identificador	**CU_27 - Contactar con soporte**		
Versión Actual	1.0	Autor	Bianca Claudia Ilas
Actor Principal	Usuario Autenticado	Actores Implicados	
Obj. Asociados		Req. Asociados	RU_03, RU_08, RU_27

Continúa en la página siguiente.

Identificador	**CU_27 - Contactar con soporte**		
Descripción	El usuario puede ponerse en contacto con el soporte técnico de la aplicación para abrir una incidencia.		
Precondiciones	▪ El usuario autenticado debe tener instalada nuestra aplicación en su dispositivo. ▪ Al usuario le ocurre una anomalía con la aplicación.		
Secuencia Normal	1. El usuario selecciona la opción contactar con soporte ubicada dentro del menú cuenta. 2. El sistema le habilita la opción de ponerse en contacto con soporte a través del email. 3. El usuario selecciona la opción email. 4. El sistema le redirige a la pasarela del email. 5. El usuario redacta la incidencia dentro del campo habilitado y la envía a soporte.		
Secuencia Alternativa			
Excepciones en la secuencia	Error al enviar el email. El sistema te pide repetir los pasos de la secuencia normal.		
Postcondiciones	Se envía correctamente la incidencia a soporte.		
Importancia	Media	Frecuencia	Baja

Tabla 4.11: Caso de uso perteneciente a Contactar con soporte

4.3. Reglas de negocio

Las reglas de negocio son importantes para el proyecto ya que definen una serie de normas y especificaciones concretas en su desarrollo. A continuación, mostramos una tabla con las reglas de negocio definidas:

ID	Descripción
RN_01	Nuestro proyecto debe cumplir con los estándares definidos en la Ley Orgánica de Protección de Datos (LOPD).
RN_02	En la aplicación solo se permite una cuenta de usuario por email.
RN_03	Las contraseñas tienen un cifrado en SHA256 cuando se almacenan en la base de datos.
RN_04	Las contraseñas deben cumplir con las siguientes características. ▪ Longitud mínima y máxima entre 8 y 16 caracteres. ▪ Mínimo una letra mayúscula y una minúscula. ▪ Mínimo un carácter especial (?,!,@,-, etc).
RN_05	Para habilitar la opción de modificar contraseña, el usuario previamente debe estar registrado con una cuenta común de correo electrónico que no sea de Google.
RN_06	Solo pueden hacer uso de la aplicación los usuarios registrados previamente en el sistema.
RN_07	No se permite introducir más de una tarjeta perteneciente a un mismo comercio.
RN_08	Cada número de tarjeta es único por usuario y es intrasferible a otros usuarios.
RN_09	La aplicación abrirá el escáner de la cámara solo si el usuario le concede los permisos necesarios previamente.
RN_10	La aplicación deberá solicitar permisos al usuario para acceder a la galería del dispositivo.
RN_11	La aplicación se ha desarrollado para dispositivos Android a partir de una versión 5.
RN_12	Las tarjetas de fidelización solo pertenecen a establecimientos de ámbito nacional.
RN_13	La actualización de puntos en las tarjetas de los usuarios en las tiendas solo es posible a través de la página web designada para este propósito. Esta plataforma actúa como un simulador interno, facilitando la gestión de puntos a nivel API de cada comercio.
RN_14	El usuario solamente puede modificar su contraseña si está previamente registrado con su email, no con los servicios de Google.
RN_15	Solamente se podrá introducir un pin de bloqueo si se habilita previamente esta opción.
RN_16	El pin de bloqueo debe tener un formato numérico de 4 dígitos.
RN_17	En la barra de búsqueda el usuario puede introducir hasta tres caracteres para obtener recomendaciones de búsqueda por tarjetas.
RN_18	La aplicación no está publicada en los servicios de Google.
RN_19	Si la versión de Android es obsoleta, una vez aceptados los permisos, no podrás volver a conceder los mismos.
RN_20	Los formularios para abrir una incidencia vienen predefinidos con un asunto y con el destinatario indicado para que el usuario solamente tenga que redactar el problema.

Tabla 4.12: Reglas de negocio

4.4. Requisitos funcionales

Los requisitos funcionales abarcan toda la funcionalidad necesaria para la implementación del proyecto.

ID	Descripción
RF_01	El sistema mostrará los campos necesarios para que el usuario pueda introducir sus datos tanto para el registro como para el inicio de sesión.
RF_02	El sistema mostrará la respuesta de los valores introducidos por el usuario a la hora de registrarse o iniciar sesión.
RF_03	El sistema es capaz de validar los campos del formulario previamente rellenados por el usuario.
RF_04	El sistema es capaz obtener la información, recuperada de la base de datos, para mostrársela al usuario.
RF_05	El sistema es capaz de comunicarse correctamente con el servidor externo para solicitar la información de las tarjetas referidas por el usuario.
RF_06	El sistema es capaz de insertar cualquier registro nuevo en la base de datos.
RF_07	El sistema introduce una a una las tarjetas al listado del usuario.
RF_08	El sistema permite visualizar las tarjetas en la ventana de no favoritas solamente aquellas que no lo sean.
RF_09	El sistema permite visualizar las tarjetas favoritas en la ventana de favoritas.
RF_10	El sistema es capaz de actualizar los campos en la base de datos con la nueva información y modifica por el usuario.
RF_11	El sistema será capaz de eliminar una tarjeta de la base de datos y refrescar la página.
RF_12	El sistema elimina las tarjetas del listado de las tarjetas.
RF_13	El sistema es capaz de añadir tarjetas favoritas y almacenarlas en la base de datos.
RF_14	El sistema es capaz de mantener la sesión iniciada del usuario autenticado.
RF_15	El sistema es capaz de mostrar un listado de las tarjetas ordenadas según las preferencias del usuario.
RF_16	El sistema cuenta con una barra de búsqueda para que el usuario puede buscar sus tarjetas de forma más rápida.
RF_17	El sistema muestra la información del perfil del usuario autenticado.
RF_18	El sistema muestra la información de las tarjetas y el historial de puntos de las mismas.
RF_19	El sistema muestra correctamente la información proporcionada por la API externa para mostrar el historial de puntos.
RF_20	El sistema es capaz de eliminar la cuenta del usuario registrado en Firebase Auth.
RF_21	El sistema es capaz de eliminar todas las tarjetas asociadas al usuario eliminado.
RF_22	El sistema notifica al usuario si está seguro de que quiere eliminar su cuenta de forma permanente de nuestra aplicación.

Continúa en la página siguiente.

ID	Descripción
RF_23	El sistema actualiza correctamente la contraseña modificada por el usuario.
RF_24	El sistema solicita permisos de acceso a la cámara al usuario para poder escanear una tarjeta.
RF_25	El sistema solicita el pin de bloqueo al usuario en caso de que este haya habilitado esta opción previamente.
RF_26	El sistema almacena correctamente el pin de bloqueo, introducido por el usuario, en la base de datos y lo valida.
RF_27	El sistema valida el pin y en caso de error muestra un mensaje para introducir los dígitos correctos.
RF_28	El sistema solicita permisos para acceder a la galería de fotos del usuario.
RF_29	El sistema es capaz de generar tanto un código QR como un código de barras en función de lo que el usuario introduce manualmente al registrar la tarjeta.
RF_30	El sistema solicita al usuario que introduzca un nombre del establecimiento de la tarjeta introducida manualmente.
RF_31	El sistema muestra la información de las tarjetas proporcionadas por la API PocketMix.
RF_32	El sistema abre un formulario de solicitud en el email en el que está el enlace de la descarga.
RF_33	El sistema envía este enlace al amigo indicado por el usuario.
RF_34	El sistema solicita confirmación al usuario para eliminar una tarjeta.
RF_35	El sistema permite al usuario autenticado la posibilidad de cerrar sesión.
RF_36	El sistema es capaz de activar la luminosidad automática a la hora de escanear una tarjeta del usuario.
RF_37	El sistema muestra por pantalla con un check los permisos que tiene activados el usuario.
RF_38	El sistema muestra al usuario el manual de ayuda.
RF_39	El sistema abre un formulario dentro de un email para que el usuario pueda redactar una incidencia y enviarla al soporte técnico.
RF_40	El sistema envía la incidencia a soporte.
RF_41	El sistema es capaz de guardar la configuración en función de las preferencias del usuario.
RF_42	El sistema muestra el listado de las tarjetas, incluyendo su nombre o no en función de la preferencia del usuario.
RF_43	El sistema muestra el listado de las tarjetas en una columna o en dos en función de las preferencias del usuario.
RF_44	El sistema muestra la información de la aplicación.
RF_45	El sistema notificará cualquier error controlado dentro de la aplicación que sea producido por el usuario.
RF_46	El sistema valida todos los campos de cualquier formulario que haya cumplimentado el usuario previamente.

Tabla 4.13: Requisitos funcionales

4.5. Requisitos no funcionales

Los requisitos no funcionales son las restricciones o requisitos impuestos al sistema. Especifican el atributo de calidad del software y se organizan según su categoría, tal y como describimos a continuación:

4.5.1. Usabilidad

Tratan de conseguir un buen nivel de utilización y aceptación por parte del usuario final.

ID	Descripción
RNF_USA_01	La aplicación será de manejo fácil y cuenta con una interfaz amigable.
RNF_USA_02	La aplicación es intuitiva para que el usuario de adapte y se sienta cómodo navegando a través de ella.
RNF_USA_03	La aplicación cuenta con un manual de ayuda sencillo para el usuario.
RNF_USA_04	La aplicación es compatible con la mayoría de los dispositivos Android, a partir de la versión 6.
RNF_USA_05	La aplicación será eficiente en el uso. Es decir, el usuario final aprenderá a manejarse en ella sacando el mayor grado de productividad posible.
RNF_USA_06	La aplicación presenta una tasa baja en errores. Es decir, la gestión de errores cometidas por el usuario final o por el sistema es baja, debido a que será capaz de volver al estado normal de funcionamiento sin complejidad.
RNF_USA_07	La aplicación permite, en el 70 % de las veces, que con un máximo de 5 clicks sea suficiente para que el usuario llegue a la información deseada.
RNF_USA_08	La aplicación permite a los nuevos usuarios, familiarizarse con ella en un tiempo máximo de 15 minutos.

Tabla 4.14: Requisitos no funcionales - Usabilidad

4.5.2. Eficiencia

Estos requisitos se encargan del establecimiento de métricas referentes a las salidas del sistema debidas a las acciones de los usuarios.

ID	Descripción
RNF_EFI_01	La aplicación tendrá un tiempo de respuesta de forma eficiente inferior a 3 segundos en las acciones realizadas dentro de la aplicación, una vez se ha iniciado sesión y una alta mejora en tiempo de procesamiento.
RNF_EFI_02	Tanto el inicio de sesión como en el de cerrar sesión, no debe tardar más de 10 segundos.

Continúa en la página siguiente.

ID	Descripción
RNF_EFI_03	La aplicación soporta el uso simultáneo y el acceso no limitado de los usuarios finales.
RNF_EFI_04	La aplicación no tiene límite para el acceso diario de los usuarios.
RNF_EFI_05	Toda respuesta a cualquier transacción realizada, tanto por el usuario como por el sistema, no excederá un tiempo de respuesta superior a 15 segundos.
RNF_EFI_06	La probabilidad de que la aplicación falle es inferior al 1 %.

Tabla 4.15: Requisitos no funcionales - Eficiencia

4.5.3. Mantenibilidad

Estos requisitos definen aspectos relacionados con la simplicidad de aumentar la funcionalidad, modificar o corregir errores en un sistema software.

ID	Descripción
RNF_MAN_01	La aplicación será capaz de restaurarse en la correción de errores.
RNF_MAN_02	La aplicación se adaptará en la evolución de los cambios para que satisfaga nuevos requisitos.

Tabla 4.16: Requisitos no funcionales - Mantenibilidad

4.5.4. Seguridad

Estos requisitos se encargan de proteger el sistema frente a posibles ataques y llevar a cabo las acciones de seguridad requeridas.

ID	Descripción
RNF_SEG_01	En caso de una brecha de seguridad, la aplicación se detendrá hasta que el administrador resuelva el problema.
RNF_SEG_02	Las transacciones realizadas con el servidor serán siempre cifradas y realizadas bajo el protocolo HTTP.
RNF_SEG_03	En la aplicación solo se permite una cuenta de usuario por email.
RNF_SEG_04	Las contraseñas tienen un cifrado en SHA256 cuando se almacenan en la base de datos.
RNF_SEG_05	Las contraseñas deben cumplir con las siguientes características. ▪ Longitud mínima y máxima entre 8 y 16 caracteres. ▪ Mínimo una letra mayúscula y una minúscula. ▪ Mínimo un carácter especial (?,!,@,-, etc). .
RNF_SEG_06	Los usuarios inician sesión en la aplicación a través de sus credenciales.

Continúa en la página siguiente.

ID	Descripción
RNF_SEG_07	Se atribuye en toda la aplicación la Ley Orgánica de Protección de Datos (LOPD), ISO 27001, ISO 27017, ISO 27018, SOC1, SOC2, SOC3.
RNF_SEG_08	La aplicación cierra sesión tras un tiempo inactivo de 5 minutos.
RNF_SEG_09	El sistema permite un máximo 3 intentos de validación.

Tabla 4.17: Requisitos no funcionales - Seguridad

4.5.5. Disponibilidad

Estos requisitos se encargan de la operatividad de un sistema.

ID	Descripción
RNF_DSP_01	La aplicación estará disponible 24 horas al día durante los 7 días de la semana.
RNF_DSP_02	La disponibilidad de la aplicación será de un 99 % cada vez que el usuario intente acceder.

Tabla 4.18: Requisitos no funcionales - Disponibilidad

4.5.6. Portabilidad

Estos requisitos se encargan de la capacidad que tiene el sistema de ser transferido de forma efectiva y eficiente en un entorno hardware, software, operacional o de utilización a otro entorno.

ID	Descripción
RNF_POR_01	La aplicación se adapta a diferentes entornos sin aplicaciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado.
RNF_POR_02	La aplicacion es de fácil instalación.
RNF_POR_03	El sistema podrá coexistir con otro software independiente.

Tabla 4.19: Requisitos no funcionales - Portabilidad

4.5.7. Implementación

Estos requisitos se encargan del proceso de desarrollo de las tecnologías y del software además de establecer los lenguajes utilizados.

ID	Descripción
RNF_IMP_01	La aplicación se desarrollará utilizando el IDE de Android Studio.
RNF_IMP_02	Como lenguaje se ha optado por Kotlin ya que es de código abierto y admite la programación funcional y orientada a objetos.
RNF_IMP_03	En la implementación de la API externa PocketMix se ha utilizado el IDE IntelliJ Idea y se ha utilizado Spring como lenguaje de programación.

Continúa en la página siguiente.

ID	Descripción
RNF_IMP_04	Para las páginas web se ha utilizado el Visual Studio Code: HTML, CSS, JavaScript.

Tabla 4.20: Requisitos no funcionales - Implementación

4.6. Requisitos de información

Estos requisitos se encargan de describir las características relacionadas con los datos y su gestión por parte del sistema.

ID	Descripción
RI_01	El sistema enviará correctamente toda la información de los usuarios nuevos registrados al servidor de Firebase Auth para que este pueda almacenarlos.
RI_02	El sistema enviará correctamente la información relacionada de las tarjetas al servidor externo PocketMix para que este las almacene correctamente en la base de datos de MySQL.
RI_03	El sistema habilita un formulario con los campos requeridos para que el usuario envíe el enlace de la descarga de la aplicación a un amigo.
RI_04	El sistema habilita un formulario predefinido para redactar incidencias y enviarlo a soporte técnico.
RI_05	El sistema almacena la configuración en función de las preferencias del usuario.
RI_06	El sistema se encargará de guardar la información de las tarjetas de fidelización asignadas a los usuarios en la aplicación.
RI_07	El sistema se encargará de guardar toda la información de los usuarios registrados en nuestra aplicación.

Tabla 4.21: Requisitos de información

4.7. Modelo Entidad - Relación

En el siguiente diagrama reflejaremos los requisitos anteriormente explicados, con la ayuda de entidades y relaciones, para poder visualizarlos de manera más ordenada.

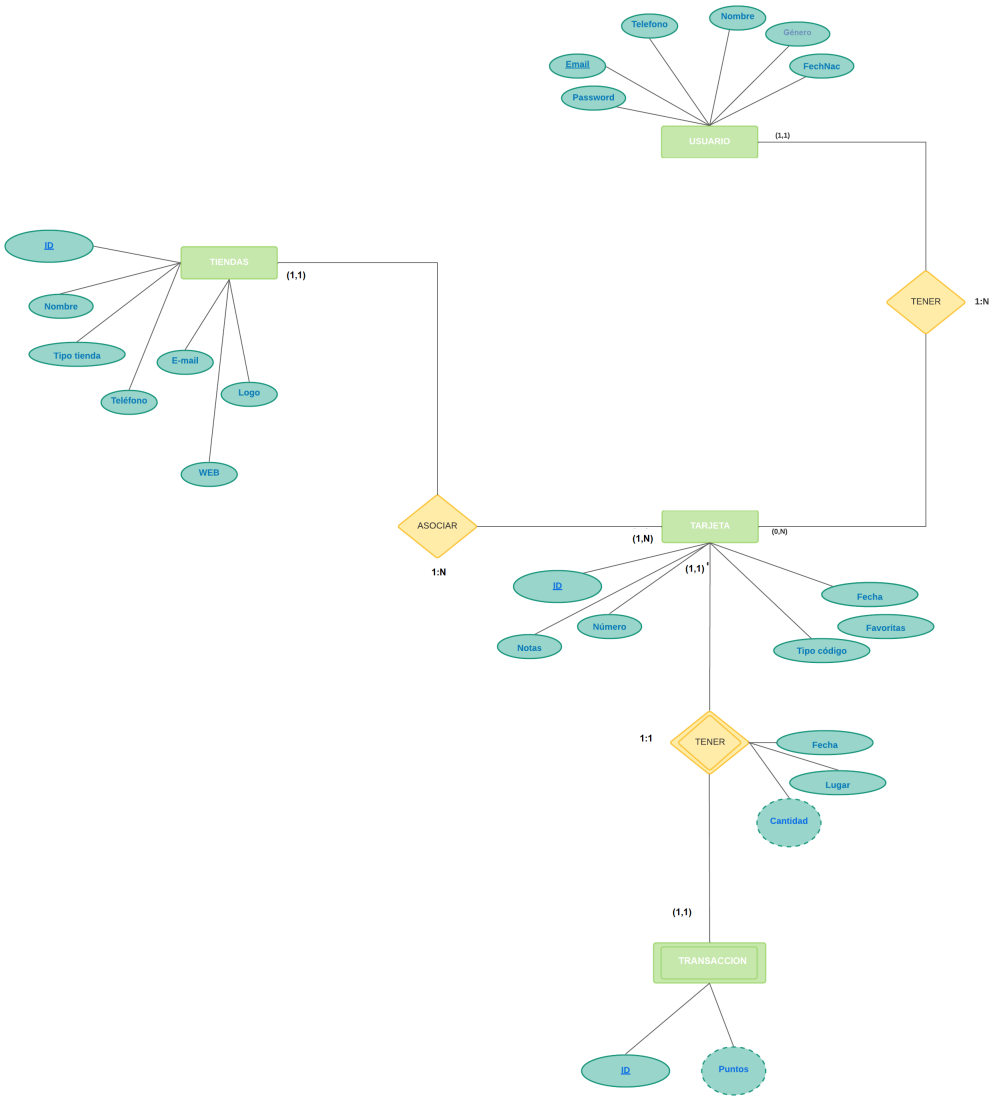

Figura 4.4: Modelo Entidad-Relación.

Capítulo 5

Diseño del Sistema

5.1. Arquitectura

La arquitectura es un esquema que presenta una estructura general del sistema para determinar cómo interactúan sus diferentes componentes. Esto es, cómo se relacionan los componentes lógicos entre sí dentro de la aplicación. Utiliza la abstracción para la representación de sistemas complejos.

5.1.1. Arquitectura lógica

A continuación, se explicará con la ayuda de la siguiente imagen, la arquitectura lógica correspondiente al sistema definido para nuestra aplicación.

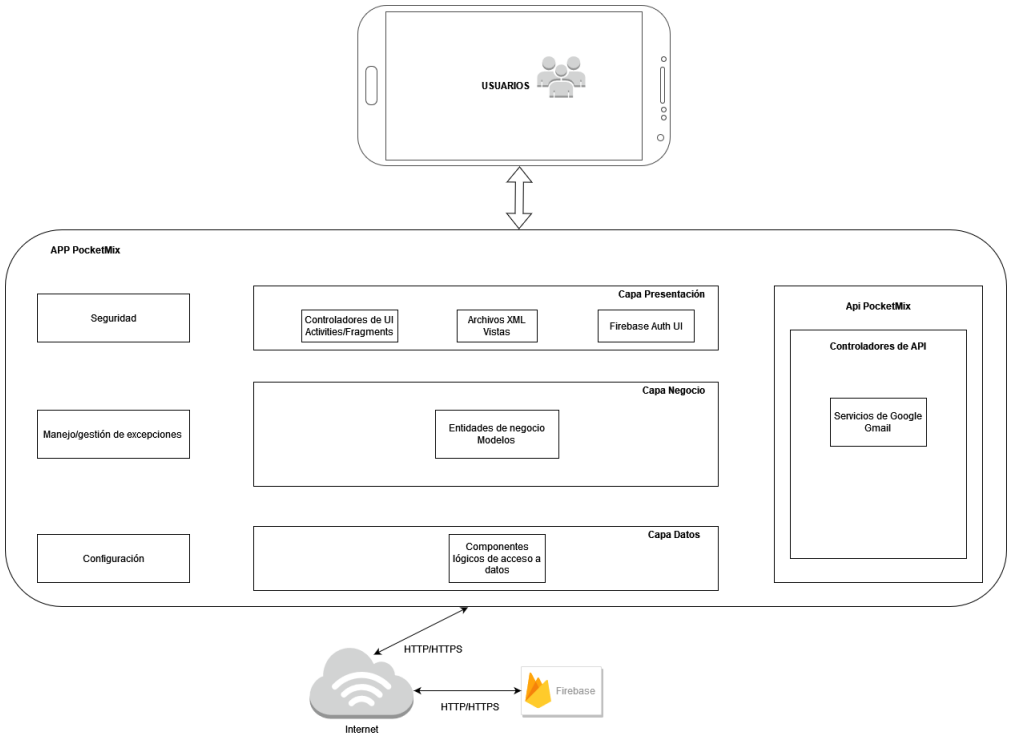

Figura 5.1: Arquitectura lógica.

Como podemos observar, este tipo de arquitectura está basada en el **modelo de tres capas**: *capa de presentación, capa de negocio y capa de datos*. Además, existe otra cuarta capa que identifica a los *agentes de servicios* externos que forman parte de la aplicación y a su vez se relacionan con las tres capas principales. Por último, disponemos de componentes de servicios comunes a las capas mencionadas, encargadas del *manejo y gestión de excepciones, configuración y seguridad* de nuestro sistema.

1. **Capa de presentación**: es la interfaz de usuario y la capa de comunicación de la aplicación, utilizada por el usuario final. Su objetivo principal es mostrar y recopilar información. Los niveles de presentación web normalmente se desarrollan utilizando lenguajes de etiqueta. Los controladores de esta capa sirven de conexión con el resto de niveles.

2. **Capa de negocio**: es el núcleo de la aplicación ya que procesa toda la información recopilada en la capa de presentación utilizando un conjunto específico de reglas de negocio y lógica empresarial. También puede añadir, suprimir o modificar datos en la siguiente capa mediante llamadas de APIs.

3. **Capa de datos**: se almacena y gestiona la información procesada por la aplicación. El nivel de presentación y el nivel de datos no pueden comunicarse directamente entre sí, favoreciendo así el aislamiento de los datos.

4. **Agentes de servicios**: aíslan la necesidad de llamar a diversos servicios adicionales. Los controladores de interfaz se encargan de la solicitud de estos servicios que operan conjuntamente con las tres capas principales.

5.1.2. Arquitectura física

Representa la manera en la que se implementa la tecnología empleada mediante todos los componentes físicos que intervienen en un sistema para su correcto funcionamiento y despliegue.

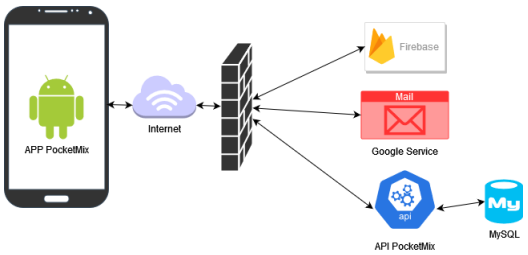

Figura 5.2: Arquitectura física.

Se define a través de un patrón cliente-servidor. El cliente será el propio dispositivo móvil del que dispone el usuario; el servidor estará compuesto por elementos diferentes:

- **Google Services**: se accede con las propias APIs de los servicios de Gmail.
- **API PocketMix**: se accede a través de una librería denominada Retro Fit de Android.
- **Firebase**: se accede a ella y a sus servicios a través de las API de Firebase, Cloud Firestore y Firebase Auth. Almacena los datos que están relacionados con los usuarios registrados.

Para dotar a la aplicación de seguridad y filtrar las peticiones de entrada y salida realizadas del lado del cliente, las conexiones y las comunicaciones están protegidas mediante un *firewall*.

5.2. Modelos de diseño

A continuación, para conocer en detalle y adentrarnos en el diseño de nuestra aplicación, mostraremos una serie de diagramas y representaciones que nos ayudarán a comprender nuestro sistema.

5.2.1. Diagrama de actividades

Estos diagramas nos van a describir lo que debe suceder en el sistema que vamos a modelar para mostrarnos comportamientos que interactúan en varios casos de uso. Se adjuntan los diagramas de la funcionalidad principal que tiene la aplicación, es decir, añadir tarjetas mediante tres opciones posibles.

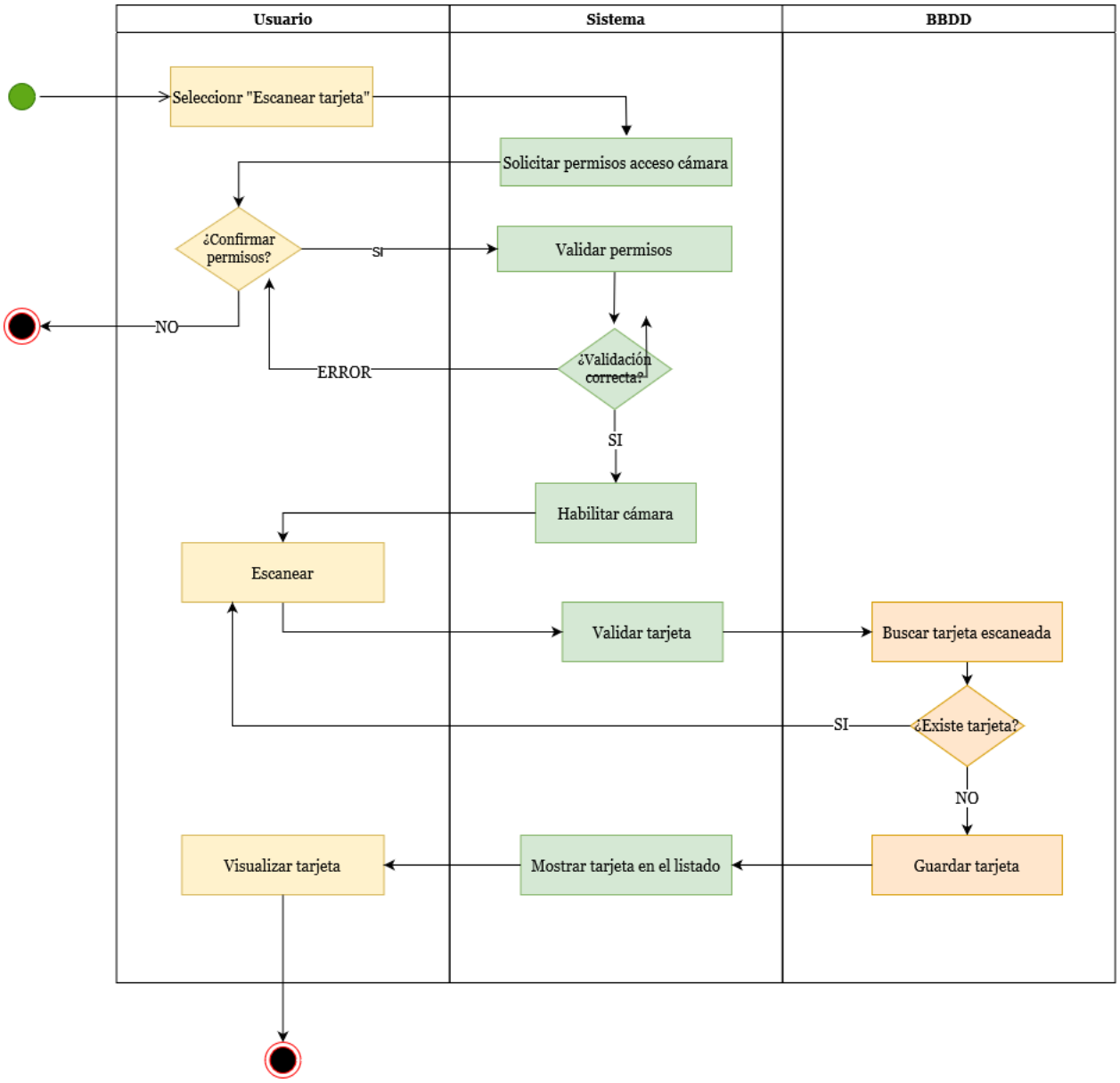

Figura 5.3: Diagrama de actividades - escanear tarjetas.

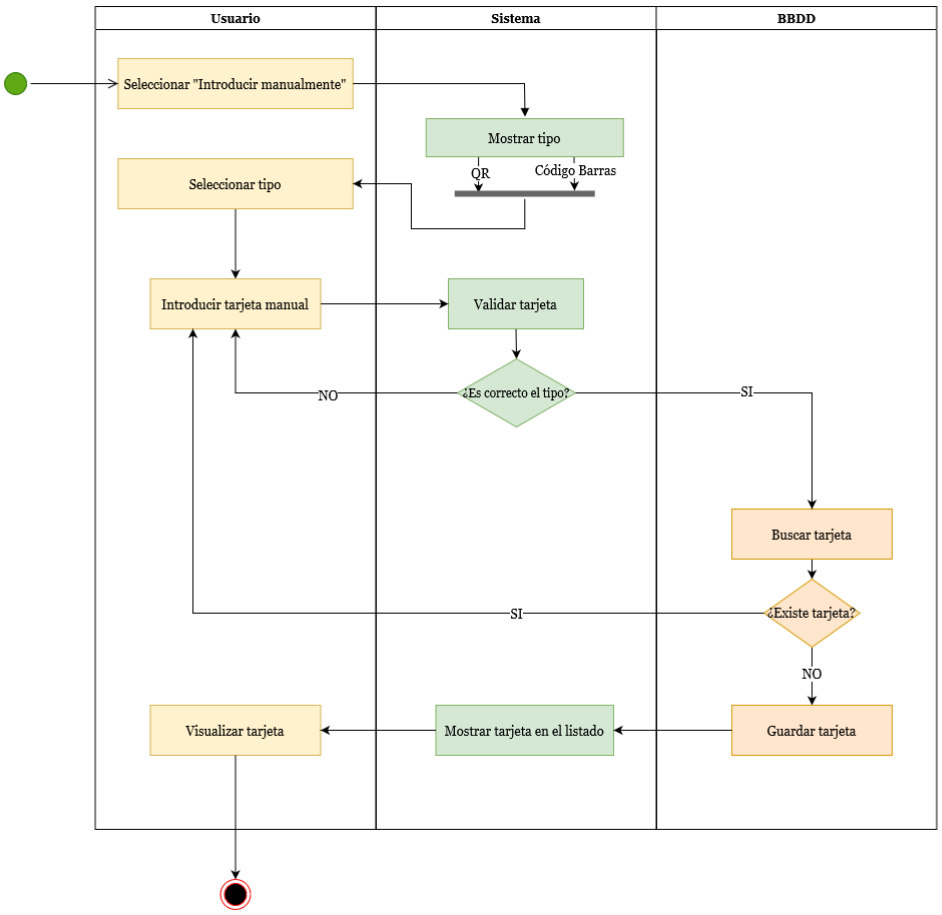

Figura 5.4: Diagrama de actividades - añadir tarjeta manualmente.

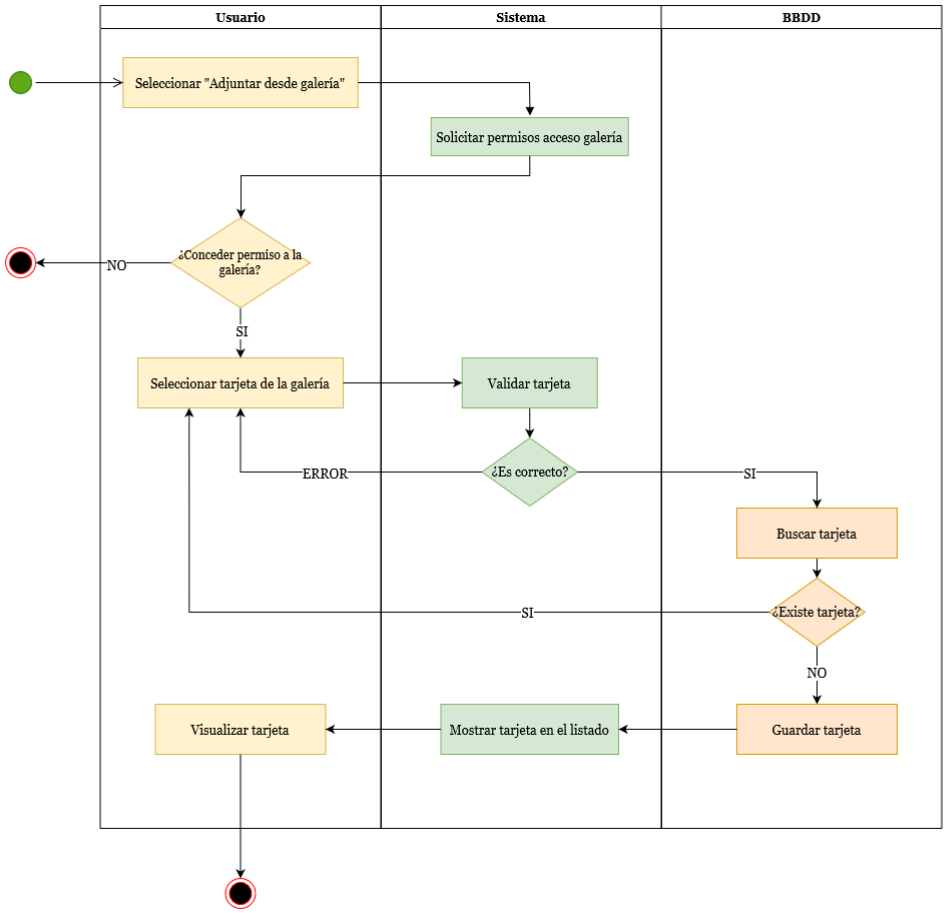

Figura 5.5: Diagrama de actividades - adjuntar desde galería.

5.2.2. Diagrama de clases

Están basados en el modelo de programación orientado a objetos y describen la estructura del sistema compuesto por **clases** y las **relaciones** existentes entre ellas.

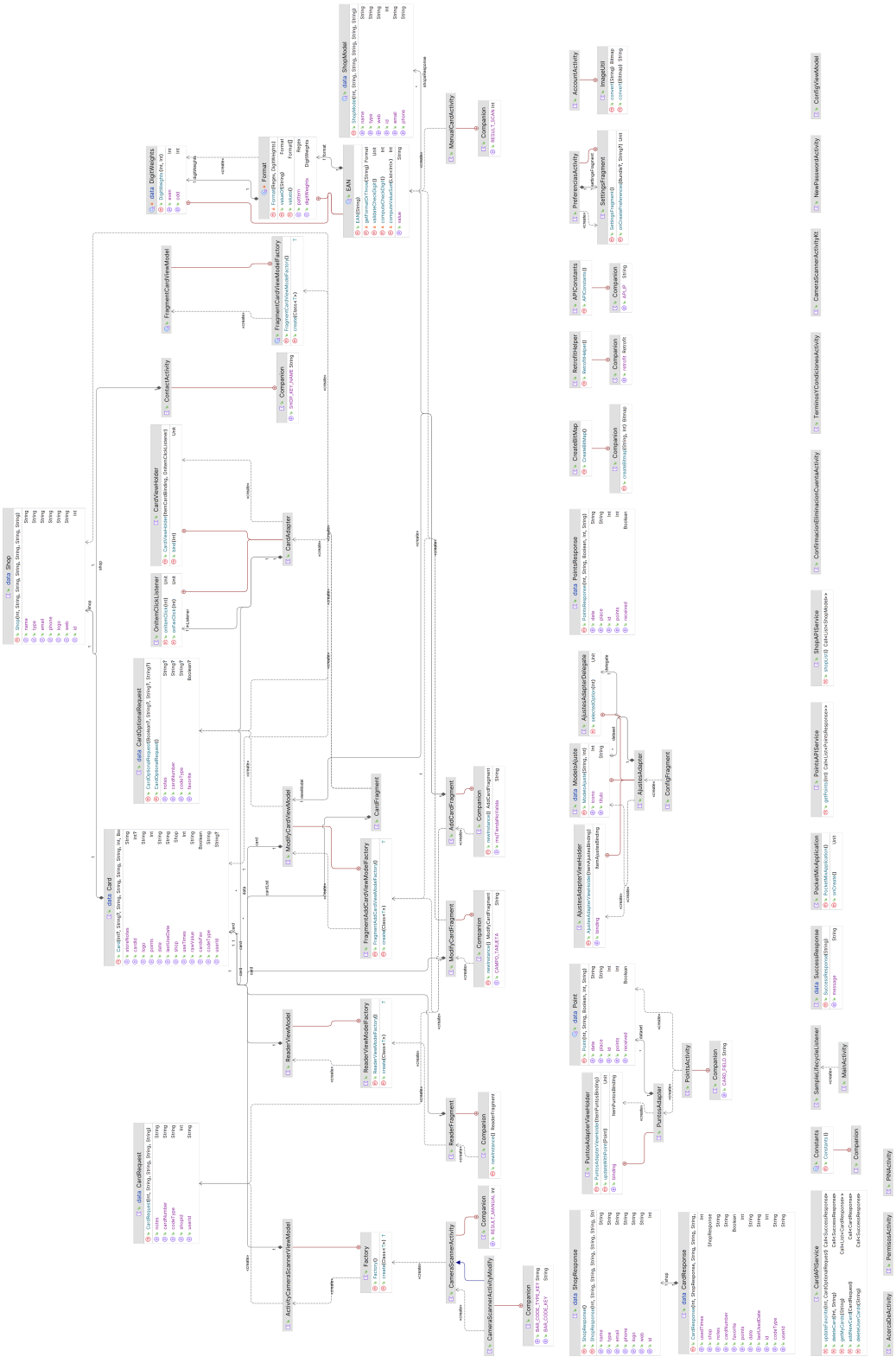

Figura 5.6: Diagrama de clases.

5.2.3. Diagrama de secuencia

Muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Contiene detalles de implementación del escenario, donde se incluyen las clases y los mensajes intercambiados.

En estos diagramas hemos intentado reflejar otras funcionalidades no descritas con anterioridad en el apartado anterior.

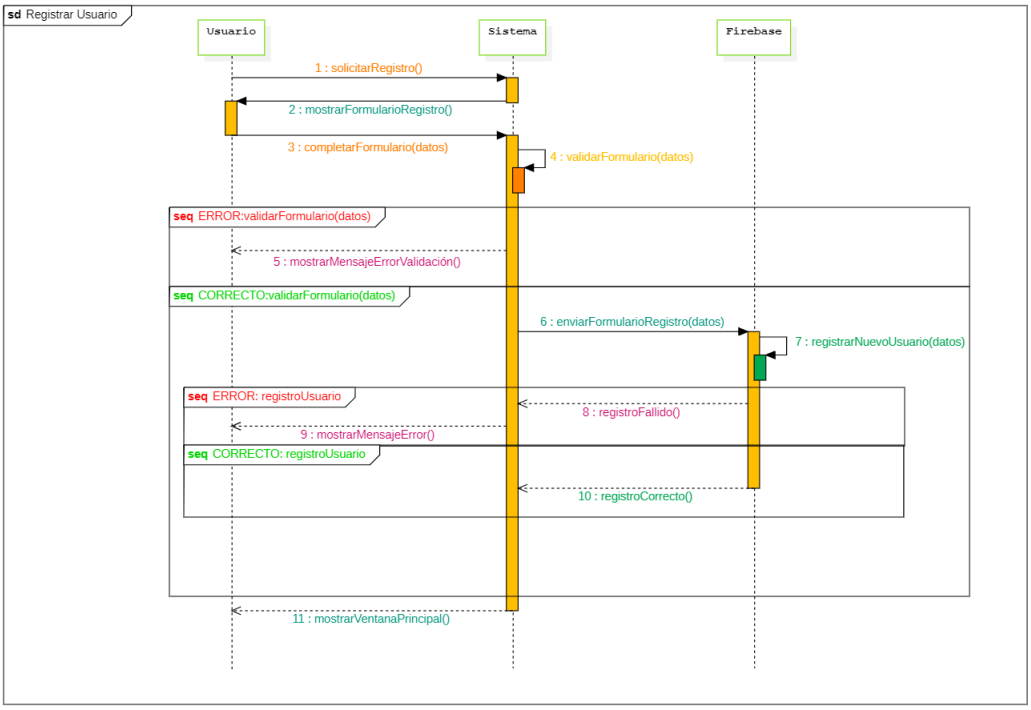

Figura 5.7: Diagrama de secuencia - registrar.

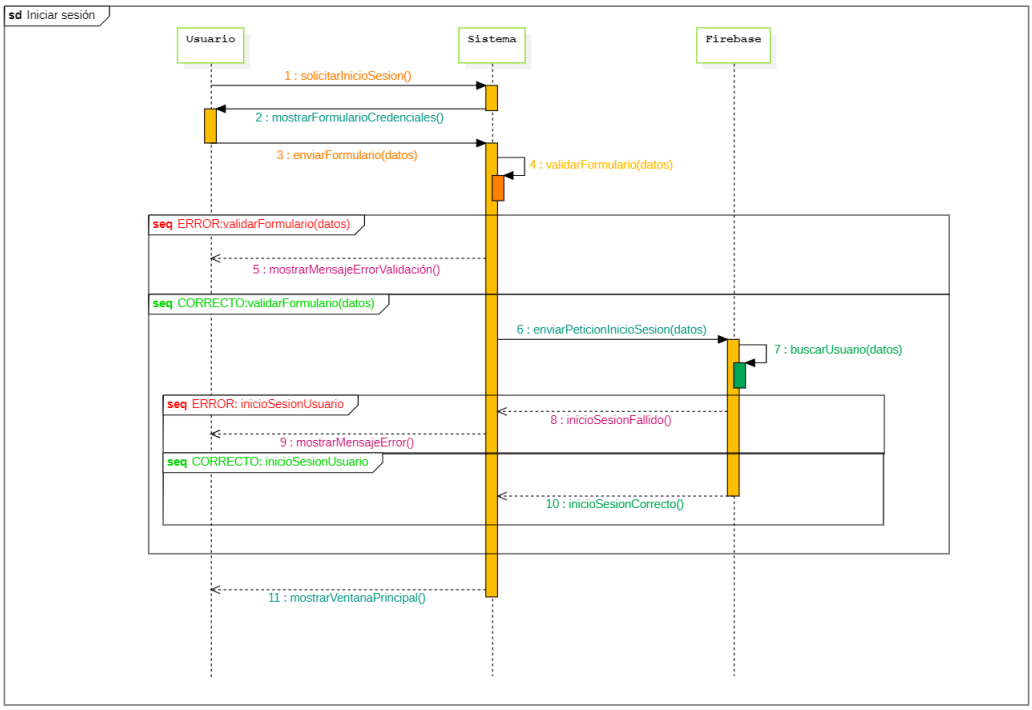

Figura 5.8: Diagrama de secuencia - inicio sesión.

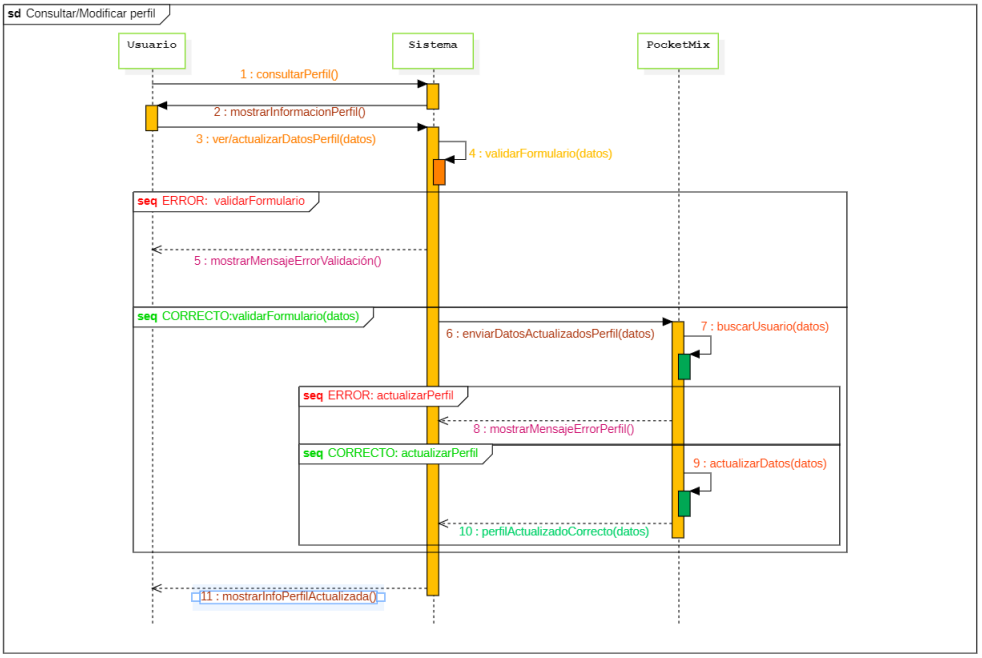

Figura 5.9: Diagrama de secuencia - consultar/modificar perfil.

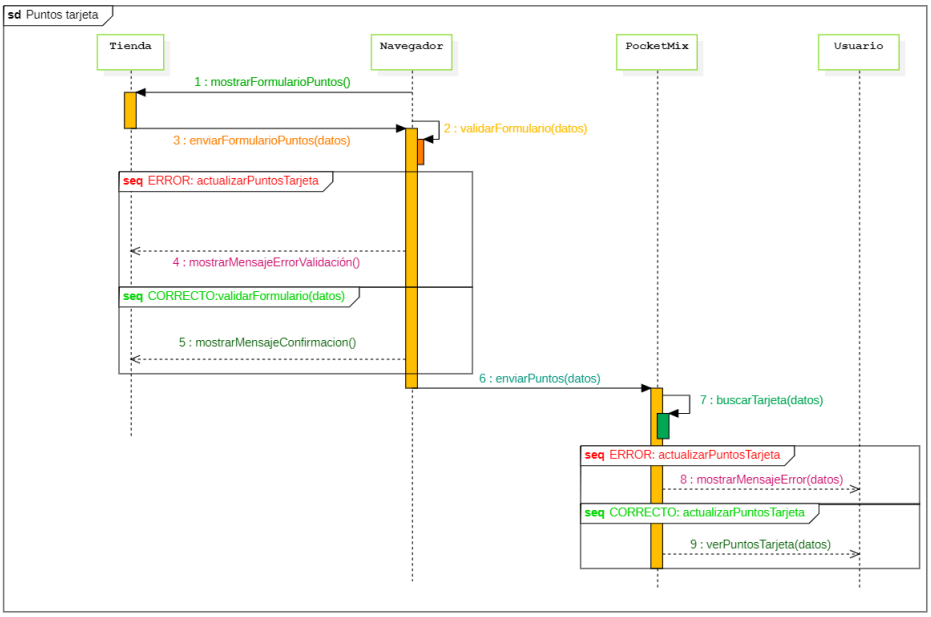

Figura 5.10: Diagrama de secuencia - puntos tarjeta.

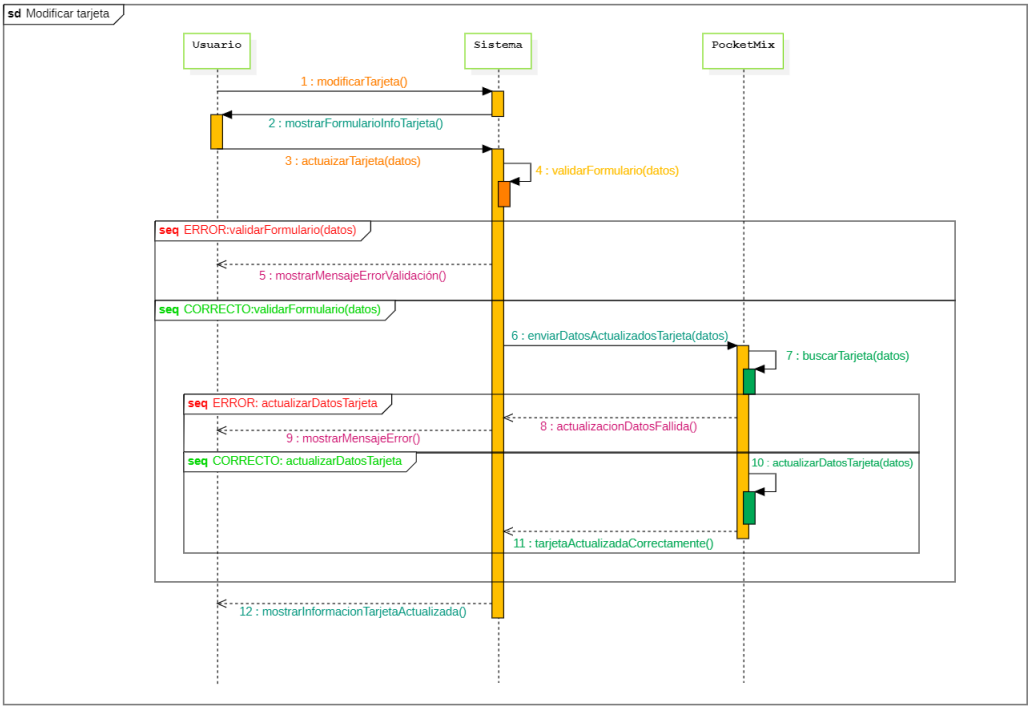

Figura 5.11: Diagrama de secuencia - modificar tarjeta.

5.2.4. Modelo lógico de datos

Para entender la estructura y el diseño de la base de datos, haremos hincapié en el modelo de programación utilizado: **ORM**. Este mapeo de objeto-relacional consiste en transformar las tablas de una base de datos en entidades, lo que permite al programador abstenerse de interactuar de manera directa con la base de datos, centrándolo en el desarrollo de la aplicación. De esta forma, facilitamos el trabajo haciéndolo más eficaz y rápido y proporcionamos más seguridad en la capa de acceso a datos contra posibles ataques.

Cloud Firestore pertenece a la plataforma **Firebase** y se define por ser una base de datos no relacional (NoSQL). Es decir, guarda los documentos en pares *clave-valor* que, a su vez, están organizados en colecciones. Por lo tanto, podemos decir que es un producto que te permite alojar los datos de tu aplicación en la nube con el objetivo de realizar una sincronización en tiempo real de estos en todos tus dispositivos.

Es necesaria la integración de Cloud Firestore con **Firebase Authentication** [21], para crear nuevos registros utilizando este Cloud Firestore como paso auxiliar.

Es decir, por un lado se ha utilizado Firebase Auth junto con Cloud Firestore para guardar los usuarios de la aplicación y por otro lado, tenemos un backend para proveer o almacenar los datos de las tarjetas, tiendas y puntos. Para el API, hemos utilizado la base de datos MySQL que es relacional y se maneja a través del ORM de Spring Data.

Por tanto, la estructura de los datos va a quedar definida tal y como se muestra en la imagen que viene a continuación:

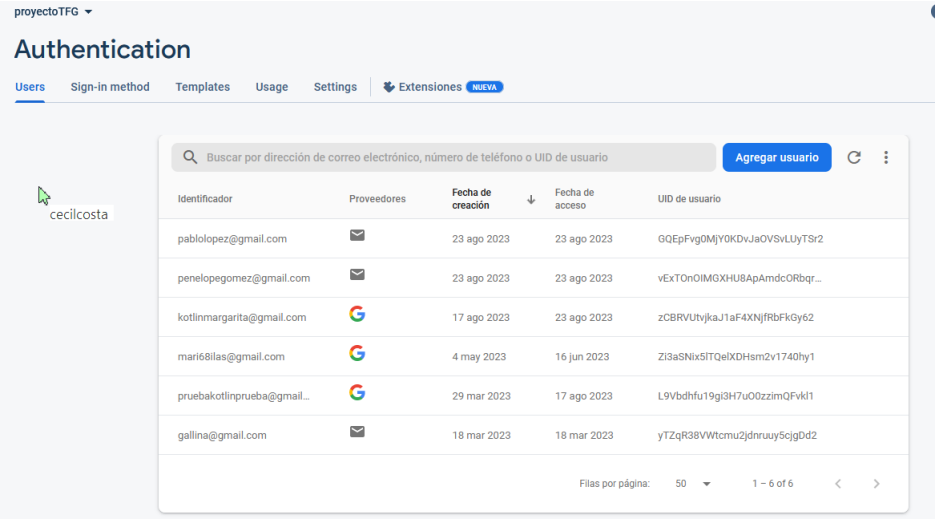

Figura 5.12: Firebase Authentication.

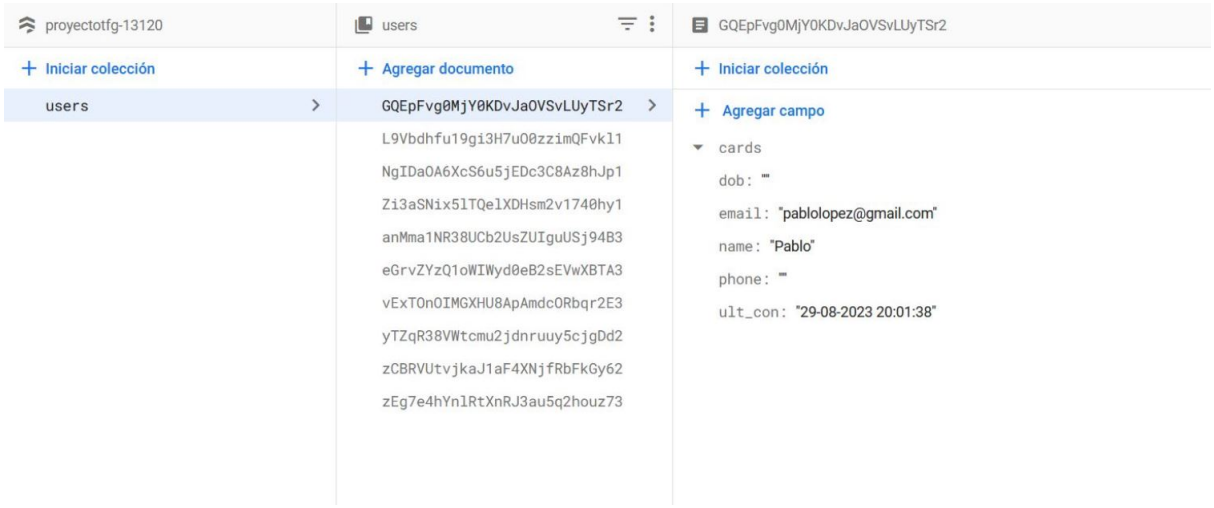

Figura 5.13: Firestore Database.

- **Iniciar colección**: aquí es donde tenemos la colección *"users"* que contiene documentos con los campos adicionales de los usuarios.
- **Agregar documento**: la aplicación tiene asignado un identificador que es un número proporcionado por Firebase Auth para dotarle de unicidad.
- **Agregar campo**: los datos personales de cada usuario se almacenan mediante pares clave-valor que abarcan una gran variedad de tipos de dato para guardar esa información.

Una vez explicados estos aspectos, detallamos el modelo lógico que nos permite comprender las relaciones que existen entre entidades, atributos e identificadores. Esto se ha hecho de manera automática gracias a la API que se ha diseñado que gestiona toda la parte de la aplicación que no tiene cabida en Firebase.

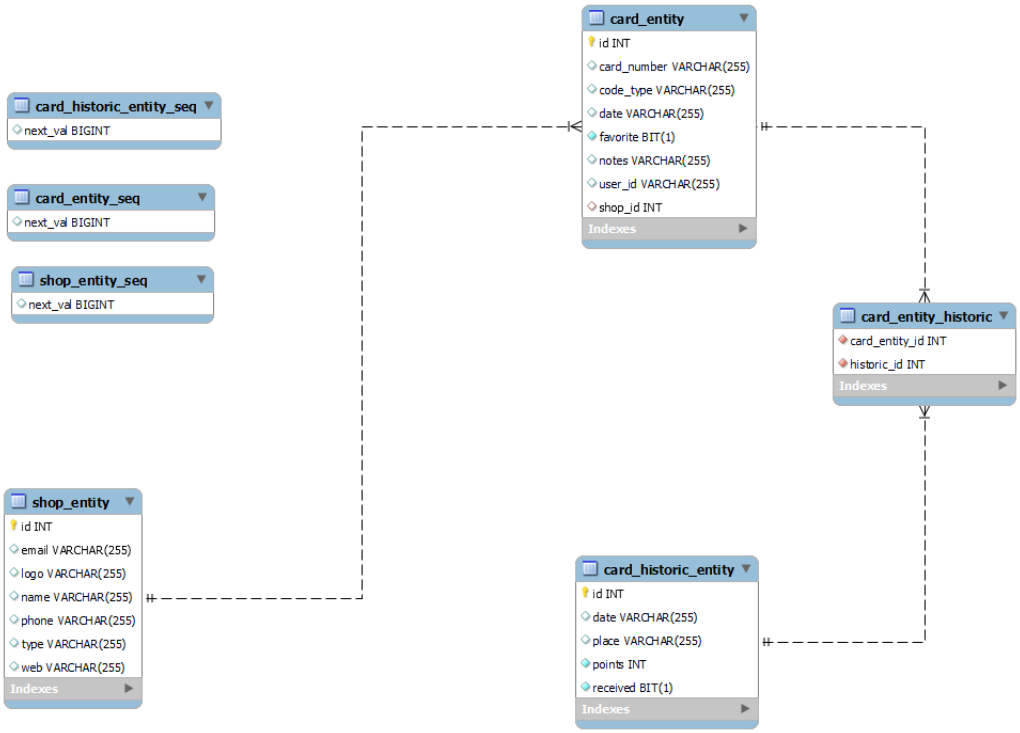

Figura 5.14: Modelo Lógico de datos.

5.3. Diseño de interfaces

5.3.1. Tecnologías y elementos utilizados

Para el diseño de la interfaz de la aplicación se ha hecho uso de los elementos técnicos que se mencionan seguidamente:

Material Design

Es una guía completa para el diseño visual, interactivo y de movimiento en plataformas y dispositivos. Fue lanzada por Google I/O en 2014 con el objetivo de unificar criterios estéticos y funcionales basados en procesos y experiencias en torno al diseño de Google para la creación de la interfaz de sistemas operativos.

Las herramientas se centran en las diferentes etapas del proceso de diseño de interfaces. Por otro lado, agilizan la forma en la que los equipos trabajan, diseñan y construyen aplicaciones conjuntamente: ejemplos de diseño, plataformas donde compartir proyectos y recibir feedback del equipo, iconos, visor interactivo para previsualizar el trabajo, etc.

Para más información consultar:

https://material.io/develop/android

Constraint Layouts

Permite crear diseños difíciles sin tener la necesidad de utilizar layouts anidados y así evitar problemas de memoria y eficiencia en los dispositivos.

Un constraint se puede crear en relación al contenedor ("parent") o a otra vista con respecto a una línea de guía denominada **guideline**.

Para más información consultar:

https://developer.android.com/training/constraint-layout?hl=es

http://www.androidcurso.com/index.php/881

Recycler View

La vista *RecyclerView* visualiza una lista o cuadrícula deslizable de varios elementos, donde cada elemento puede definirse mediante un layout. Su utilización es algo compleja, pero muy potente. Dentro del API de Android encontramos las vistas *ListView* y *GridView* nos ofrece una alternativa a RecyclerView. Esta última no ha sido añadida a ningún API si no que se añade en una librería de compatibilidad. A pesar de que resulta algo más compleja de manejar, es más recomendable su uso.

Para más información consultar:

https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=es-419

http://www.androidcurso.com/index.php/691

Linear Layouts

Es un contenedor que nos permite incluir listas en fila ya sea en horizontal o en vertical. Es sencillo y práctico.

Para más información consultar:

https://developer.android.com/develop/ui/views/layout/linear

Frame Layouts

Se trata del layout mas básico de todos y su uso suele ser para añadir un solo hijo o para vistas muy sencillas.

Para más información consultar:

https://cursokotlin.com/framelayout-diseno-de-layouts

https://developer.android.com/reference/android/widget/FrameLayout

5.3.2. Interfaces

En la siguiente sección se describen las vistas que se muestran al usuario en cada una de la funcionalidades de la aplicación.

Como interfaz designamos, la conexión física y funcional que se establece entre dos dispositivos o sistemas que funcionan independientemente uno del otro. Es decir, sirve para realizar la comunicación entre una persona y un ordenador para mostrar al usuario las funcionalidades de la aplicación.

Se ha intentado que nuestra aplicación cuente con un diseño amigable para el usuario de forma que sea sencillo y se sienta cómodo al navegar dentro de ella.

Ventana inicial, registro e inicio de sesión

- **Descripción**: pantalla de registro y de autenticación que se muestra al usuario al abrir la aplicación.
- **Activación**: al iniciar la aplicación en el dispositivo.
- **Eventos**:
  1. Pulsando el botón de *Acceder con email* o pulsando el botón *Registrarte con Google* el usuario se registra o inicia sesión en la aplicación y se le redirige a la pantalla principal.
  2. Pulsando el botón de atrás de nuestros dispositivos, finalizamos la ejecución de la aplicación.
- **Diseño**:

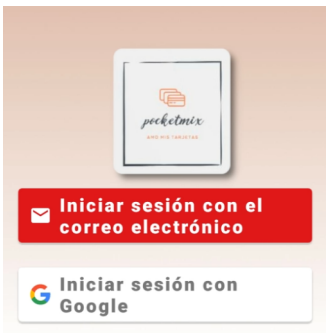

Figura 5.15: Ventana inicial, registro e inicio de sesión

Pantalla principal

- **Descripción**: pantalla que permite visualizar el listado de todas nuestras tarjetas, tanto las favoritas como las que no lo son. Además, es desde aquí desde donde podemos añadir nuevas tarjetas, buscarlas u ordenarlas.
- **Activación**: pantalla principal de la aplicación.
- **Eventos**:
  1. Pulsando el botón *buscar* introduces el nombre del comercio para localizarlo en tu listado de tarjetas.

2. Pulsando en el icono de menú de *ordenación* puedes clasificar las tarjetas a tu gusto.
3. Si pinchamos sobre alguna de las tarjetas del listado, accedemos a la información y otras funcionalidades relacionadas con ellas.
4. El botón de *añadir tarjeta* nos redirige a otra pantalla para completar el registro con la información de la nueva tarjeta.

- **Diseño**:

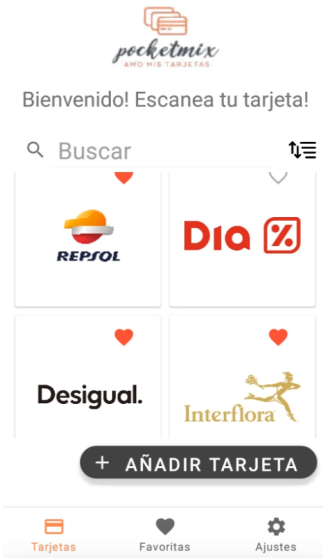

Figura 5.16: Pantalla principal

Tarjetas favoritas

- **Descripción**: pantalla para visualizar el listado de tarjetas favoritas.
- **Activación**: opción *favoritas* ubicada en la parte inferior.
- **Eventos**: mismas funcionalidades que la **Pantalla principal**.
- **Diseño**:

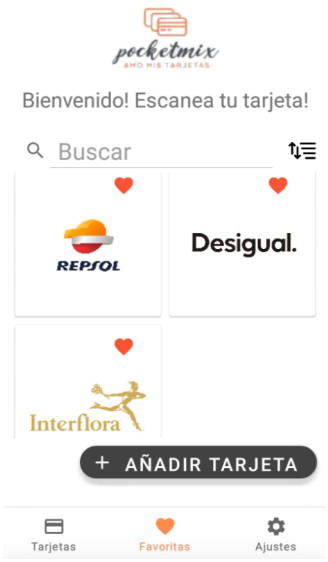

Figura 5.17: Tarjetas favoritas

Ajustes

- **Descripción**: pantalla para configurar los diferentes parámetros y funcionalidades de la aplicación.
- **Activación**: seleccionando la opción *ajustes* ubicada en la parte inferior derecha de la pantalla.
- **Eventos**:
  1. Seleccionando *Cuenta* accedes a la información y funcionalidades del perfil de usuario.
  2. Pinchando sobre *Preferencias* puedes configurar diferentes parámetros relacionados con las tarjetas o activar el bloqueo pin.
  3. Pulsar sobre *Recomendar a un amigo* accede al correo para remitir un enlace de descarga y compartir la aplicación con otros posibles usuarios.
  4. Seleccionando *Permisos* visualizamos la gestión de permisos de la aplicación.
  5. Seleccionar *Ayuda* redirige a una página web donde viene descrito el manual de ayuda.
  6. A través de *Soporte* se habilita el correo electrónico con una plantilla predefinida para redactar una incidencia a soporte.
  7. Haciendo clic en *Acerca de PocketMix* accedes a la información de la aplicación.
  8. Pinchando sobre *Cerrar sesión* finaliza la sesión del usuario.
- **Diseño**:

Figura 5.18: Ajustes

Añadir tarjeta

- **Descripción**: pantalla donde se habilita un formulario para cumplimentarlo con la información correspondiente a una nueva tarjeta de fidelización.
- **Activación**: pulsando sobre el botón *Añadir tarjeta*.
- **Eventos**:
  1. Rellenar el campo *Nombre del comercio* de tipo obligatorio.
  2. Rellenar el campo opcional *Notas*.
  3. Pulsando uno de estos botones, se elige la forma en la que se va a añadir la tarjeta: manual, escaneada o adjunta.
- **Diseño**:

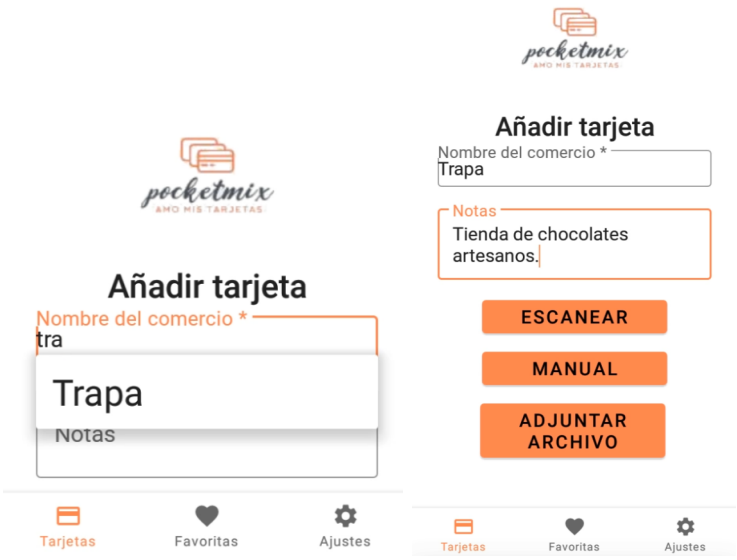

Figura 5.19: Añadir tarjeta

Figura 5.20: Añadir tarjeta manualmente

Consultar tarjeta

- **Descripción**: pantalla donde se muestra el código de barras de la tarjeta junto con su número y nombre identificativos. También podemos ver otras funcionalidades relacionadas con dicha tarjeta.
- **Activación**: haciendo clic sobre la tarjeta dentro del listado principal de tarjetas del usuario.
- **Eventos**:

1. En primer lugar, para que el usuario consulte sus puntos, la tienda previamente los tiene que añadir tal y como se muestra a continuación:

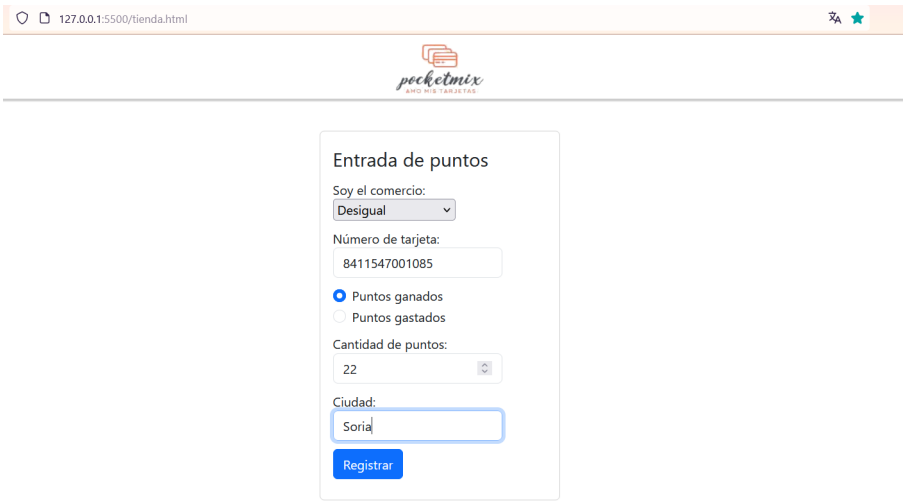

Figura 5.21: Añadir puntos

2. Pulsando sobre el botón *Puntos*, podemos ver tantos los puntos descontados como los obtenidos de la tarjeta de fidelización del usuario.

Figura 5.22: Puntos tarjeta

3. Haciendo clic en *Modificar tarjeta* se abre una pantalla para rellenar los campos, tales como: número de tarjeta (introducir manualmente o escaneándola), seleccionar el tipo de tarjeta (QR o código de barras), añadir notas. Una vez cumplimentados estos campos, se puede escoger el botón *Actualizar tarjeta* o *Cancelar la actualización.*

Figura 5.23: Modificar tarjeta

4. Pinchando sobre *Atención al cliente* las diferentes opciones de contacto del comercio.

Figura 5.24: Atención al cliente

5. El botón *Eliminar tarjeta*, se abre un pop-up donde se le pregunta al usuario si realmente desea eliminar la tarjeta y este debe confirmarlo pulsando en los botones de *Sí* o *No*.

Figura 5.25: Eliminar tarjeta

- **Diseño**:

Figura 5.26: Consultar tarjeta

<u>Mi cuenta</u>

- **Descripción**: en esta pantalla se puede consultar la información del perfil del usuario registrado, así como modificar la información añadida.
- **Activación**: seleccionando la opción *Cuenta* que se encuentra dentro de los ajustes de la aplicación.
- **Eventos**:

1. Permite añadir o cambiar la foto del perfil.
2. En la parte superior, se muestra el email del usuario y la última conexión.
3. Rellenar o modificar los campos Nombre, Fecha de Nacimiento, Teléfono y Género.
4. Como podemos observar en las imágenes, si la cuenta no es de un usuario de gmail, se habilita la opción de *Modificar contraseña*.
5. Si seleccionamos el botón *Cerrar sesión* el usuario sale de la aplicación.
6. Pulsando sobre el botón *Eliminar cuenta* se elimina la cuenta de usuario de forma permanente de la aplicación.

- **Diseño**:

Figura 5.27: Mi cuenta

Eliminar cuenta

- **Descripción**: pantalla para dar de baja la cuenta de usuario de nuestra aplicación.
- **Activación**: seleccionando el botón *Eliminar cuenta* ubicado dentro del apartado *Mi cuenta* que, a su vez, lo encontramos dentro de *Ajustes*.
- **Eventos**:
  1. Pulsando el botón *Eliminar cuenta* se elimina la cuenta de forma segura y permanente.
  2. Seleccionando la opción *Mantener cuenta* se conserva la cuenta actual del usuario.
- **Diseño**:

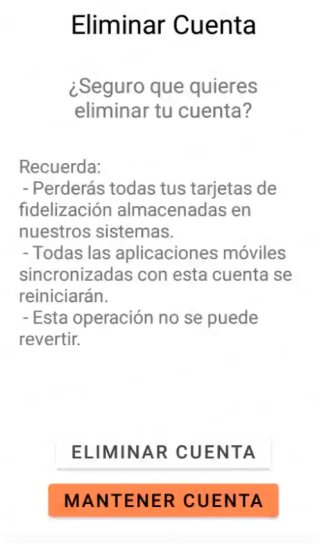

Figura 5.28: Eliminar cuenta

Preferencias

- **Descripción**: pantalla donde se muestra la previsualización de las tarjetas en función de los gustos del usuario pudiendo varias los parámetros: mostrar o no el nombre de las tarjetas, visualizar la vista de tarjeta en una o dos columnas o habilitar un pin de bloqueo para la aplicación.
- **Activación**: pulsando sobre la opción *Preferencias* ubicada dentro de *Ajustes*.
- **Eventos**:
  1. Habilitar la opción *Mostrar nombre tarjetas* pulsando el botón de activación.
  2. Si pulsamos sobre *Personalizar vista tarjetas* se habilita un pop-up para seleccionar entre dos opciones y que aparezcan en una o dos columnas.

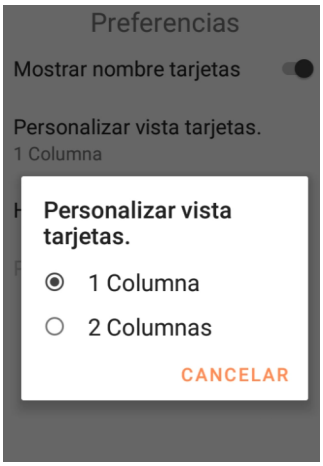

Figura 5.29: Personalizar vista tarjetas

3. Activando la opción *Habilitar bloqueo de pantalla* aparece un pop-up para añadir un pin o cancelar el evento.

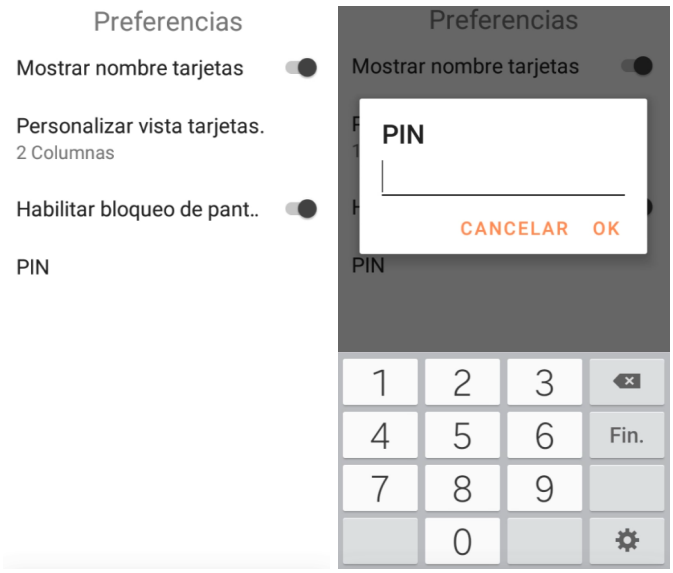

Figura 5.30: Habilitar bloqueo de pantalla

Recomendar a un amigo

- **Descripción**: permite al usuario compartir la aplicación con otros posibles usuarios.
- **Activación**: pulsando la opción *Recomendar a un amigo* que se encuentra dentro de *Ajustes*.
- **Eventos**:
  1. Se activan *recomendados* y, de entra las opciones, el usuario puede elegir enviarlo por gmail, por correo electrónico o sms.

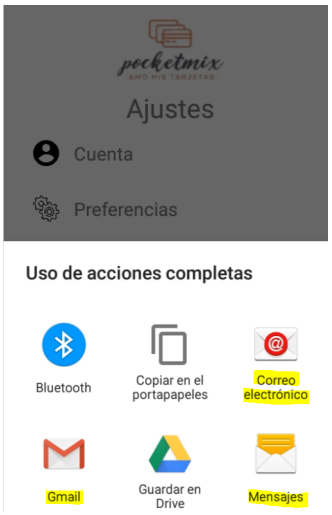

Figura 5.31: Recomendados

2. Seleccionando la opción a través del correo común, se abre el correo electrónico y se muestra un texto predefinido para ayudar con el envío del mensaje de recomendación.

- **Diseño**:

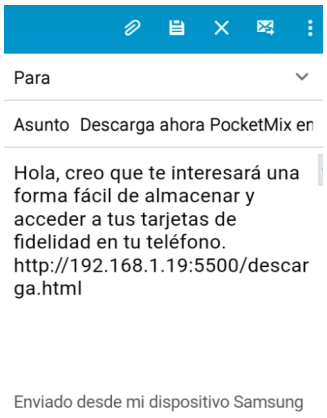

Figura 5.32: Recomendar a un amigo

Permisos

- **Descripción**: en esta pantalla se gestiona todo todo lo relacionado con los permisos de la cámara y el acceso a la galería del dispositivo móvil.
- **Activación**: pulsando sobre la opción *Permisos* dentro de *Ajustes*.
- **Eventos**:

1. Seleccionando el botón de *configuración de la aplicación*, se redirige a los ajustes del dispositivo móvil para habilitar los permisos requeridos.

- **Diseño**: Se muestra con un check, ✓, si están o no habilitados los accesos de cámara y galería de fotos.

Figura 5.33: Permisos

Ayuda

- **Descripción**: permite al usuario consultar el manual ayuda cuando le surja un problema con la aplicación.
- **Activación**: pulsando la opción *Ayuda* dentro de *Ajustes*.
- **Eventos**:
  1. Abre el navegador web donde se muestran las diferentes opciones de ayuda.
- **Diseño**:

Figura 5.34: Ayuda

Soporte

- **Descripción**: permite al usuario redactar una incidencia al soporte técnico que haya podido surgirle con la aplicación mediante diferentes vías: email o sms.
- **Activación**: pulsando la opción *Soporte* que se encuentra dentro de *Ajustes*.
- **Eventos**:
  1. Se activan *recomendados* y, de entra las opciones, el usuario puede elegir enviarlo por gmail, por correo electrónico o sms.

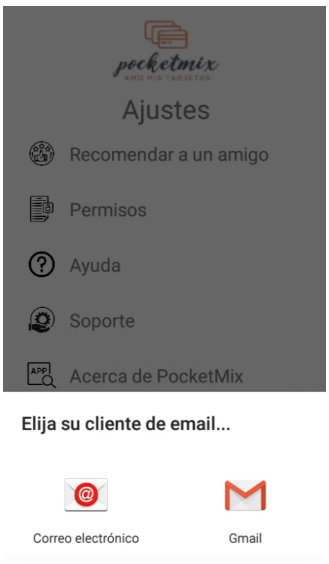

Figura 5.35: Soporte

2. Seleccionando la opción a través del correo común, se abre el correo electrónico y se muestra un texto predefinido para ayudar con la incidencia y de esta forma, soporte lo intenta solucionar con la mayor brevedad posible.

- **Diseño**:

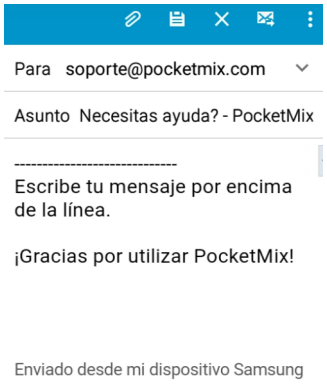

Figura 5.36: Soporte

Acerca de PocketMix

- **Descripción**: permite al usuario consultar la información acerca de la aplicación tales como la versión y los derechos de autor.
- **Activación**: pulsando la opción *Acerca de PocketMix* que se encuentra dentro de *Ajustes*.
- **Eventos**: -
- **Diseño**:

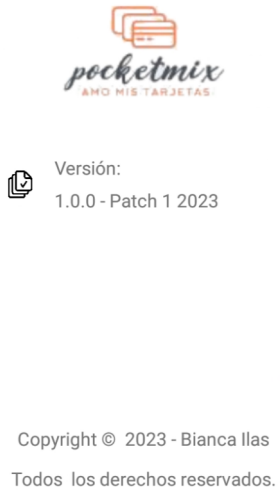

Figura 5.37: Acerca de PocketMix

Capítulo 6

Implementación

6.1. Requerimientos Hardware y Software

Para poner en marcha la aplicación, evitando error y con la mayor fluidez posible, se deberá contar con unos requisitos previos para utilizar nuestra aplicación.

Software

- **Sistema operativo**: Android 5 o superior.
- Otras opciones para compartir de aplicación: email o sms.
- Navegador.

Hardware

- **RAM**: mínimo 1GB.
- **CPU**: Quad-Core 1,2GHz o superior
- **Memoria interna**: mínimo 100MB libres.
- **Conectividad**: Wifi o datos móviles para contar con una conexión a Internet estable.
- Compatibilidad con los servicios de Google.

6.2. Herramientas empleadas

6.2.1. Herramientas para frontend

Android Studio

Es el entorno de desarrollo integrado (IDE) oficial creado por Google que se usa en el desarrollo de apps para Android. Este software proporciona herramientas avanzadas para la codificación, depuración, pruebas y diseño de aplicaciones Android, junto con un emulador de dispositivo para

probar aplicaciones en diferentes configuraciones. Está basado en el potente editor de código y las herramientas para desarrolladores de IntelliJ IDEA y ofrece más funciones que mejoran la productividad:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android.
- Integración con GitHub [19] y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra.
- Variedad de marcos de trabajo y herramientas de prueba.

Visual Studio Code [11]

Es un editor de código fuente altamente popular y ampliamente utilizado desarrollado por Microsoft. VS Code es un editor de código ligero y altamente personalizable y es conocido por su versatilidad y extensibilidad, lo que permite a los desarrolladores personalizarlo para satisfacer sus necesidades específicas a través de una amplia variedad de extensiones disponibles en su marketplace.

- JavaScript [6]: es un lenguaje de programación usado para aplicaciones web, que se ejecuta en los navegadores, manipula el contenido de las páginas, es dinámico y ampliamente utilizado.
- HTML [6]: es un lenguaje de marcas utilizado para crear páginas web. Define la estructura y contenido de una página a través de etiquetas. Facilita la creación de hipervínculos, formularios y contenido multimedia.
- CSS [6]: lenguaje de diseño gráfico que se encarga de la apariencia y la creación de la presentación de un documento escrito en un lenguaje de marcas.
- Bootstrap [13]: es un marco de diseño web que facilita la creación de aplicaciones web atractivas y dinámicas mediante componentes y estilos predefinidos. Ayuda a diseñar páginas web de manera rápida y sencilla.

Firebase

Es una plataforma de desarrollo de aplicaciones en la nube creada por Google que ofrece una variedad de servicios, como autenticación, almacenamiento, base de datos y análisis, para simplificar el desarrollo de aplicaciones móviles y web.

6.2.2. Herramientas de soporte

Overleaf [38]

Es una plataforma en línea para editar documentos *LaTeX* siendo éste un sistema de composición de documentos. Se ha utilizado para la realización y elaboración de la memoria de este trabajo.

Smartsheet [37]

Plataforma de gestión de proyectos y colaboración en línea para llevar a cabo la realización del diagrama de Grant para nuestro proyecto.

Adobe Reader DC

Herramienta desarrollada por Adobe Systems utilizada para la gestión y el manejo de archivos en formato PDF. En este proyecto se ha utlizado para la creación, lectura y manejo de este tipo de archivos.

Draw.io

Herramienta de creación de diagramas en línea con la que hemos diseñado los diagramas y figuras empleados en este proyecto.

Canva

Es una plataforma en línea para la creación y edición de diseños gráficos y documentos visuales. Se ha empleado en este trabajo para la elaboración de diagramas.

GIMP - GNU Image Manipulation Program

Es un programa de edición de imágenes de código abierto y gratuito que ofrece herramientas para retocar, editar y crear imágenes.

6.3. Tecnologías utilizadas para backend

IntelliJ IDEA

Es un entorno de desarrollo integrado (IDE) creado por JetBrains, que se utiliza principalmente para el desarrollo de aplicaciones de software en diversos lenguajes de programación, incluyendo Java, Kotlin, Groovy y muchos otros. Este IDE se destaca por su potente conjunto de herramientas de desarrollo, que incluye resaltado de sintaxis inteligente, completado automático de código, refactorización de código, depuración avanzada y análisis estático. Además, es conocido por su capacidad de integración con numerosos marcos y bibliotecas, lo que facilita la creación de aplicaciones de alta calidad en múltiples plataformas.

Kotlin [2]

Es un lenguaje de programación de código abierto desarrollado por JetBrains que se ejecuta en la máquina virtual de Java (JVM). Es conocido por su concisión, seguridad y facilidad de lectura, y se utiliza ampliamente para el desarrollo de aplicaciones Android, así como para aplicaciones de servidor y otros tipos de desarrollo de software. Kotlin se ha convertido en un lenguaje de programación popular y está respaldado por Google como un lenguaje oficial para el desarrollo de aplicaciones.

Spring [5] [8] [9]

Es un marco de desarrollo en Java que simplifica la creación de aplicaciones empresariales al gestionar la inversión de control y la programación orientada a aspectos. Facilita la construcción de aplicaciones escalables y modulares.

- Spring Data: es un subproyecto de la familia Spring centrado en simplificar el acceso a datos en aplicaciones Java. Proporciona una capa de abstracción que facilita la interacción con diferentes tecnologías de almacenamiento de datos, incluyendo bases de datos relacionales y NoSQL. Además, reduce la cantidad de código repetitivo al ofrecer métodos y consultas comunes para acceder y manipular datos.
- Spring Boot: es un framework de desarrollo de aplicaciones en Java utilizado para crear aplicaciones web y microservicios de forma rápida y simple. Simplifica la configuración y el desarrollo al proporcionar una serie de características preconfiguradas, como un servidor web integrado, gestión de dependencias y configuración automática.

MySQL

Es un sistema de gestión de bases de datos relacionales de código abierto. Es ampliamente utilizado en el mundo de la tecnología debido a su rapidez, fiabilidad y facilidad de uso. Almacena datos de manera estructurada y se integra con varios lenguajes de programación.

Con el uso en conjunto de Spring Boot, Spring Data y MySQL, los desarrolladores pueden crear aplicaciones Java de manera más rápida y eficiente, gestionar el acceso a datos de manera sencilla y utilizar una base de datos fiable y ampliamente utilizada como MySQL para almacenar información. Esta combinación es especialmente común en el desarrollo de aplicaciones empresariales y en la implementación de microservicios.

6.4. Librerías y dependencias

Firebase Auth

'com.google.firebase:firebase-firestore-ktx:24.4.0'

Es una dependencia que proporciona funcionalidades de autenticación segura para aplicaciones móviles y web a través de Firebase. Se puede utilizar para gestionar la autenticación de usuarios de manera eficiente en tu proyecto.

Firebase Firestore

'com.google.firebase:firebase-firestore-ktx:24.4.0'

Forma parte de Firebase. Es una base de datos en tiempo real para aplicaciones móviles y web que está optimizada para proyectos escritos en Kotlin (de ahí el sufijo "ktx").

Firebase UI

'com.firebaseui:firebase-ui-auth:8.0.2'

Relacionada con Firebase Authentication, una biblioteca de Firebase que simplifica la implementación de la autenticación de usuarios de manera más sencilla en aplicaciones móviles.

Camera

'androidx.camera:camera-mlkit-vision:1.2.0-alpha03'

Se enfoca con la integración de la cámara y el kit de visión de ML en aplicaciones Android. Permite que los dispositivos analicen y comprendan lo que ven en imágenes y vídeos.

Barcode Scanning

'com.google.mlkit:barcode-scanning:17.0.2'

Forma parte de ML Kit, una plataforma de Google para el desarrollo de aplicaciones que integran capacidades de aprendizaje automático (Machine Learning). Esta biblioteca específica se utiliza para escanear y procesar códigos de barras en imágenes o en tiempo real a través de la cámara de un dispositivo móvil. Proporciona la capacidad de detectar códigos de barras en imágenes y extraer información relevante de ellos, como números y otros datos codificados en los códigos de barras.

ZXING

'com.google.zxing:core:3.4.0'

Esta biblioteca forma parte de ZXing (Zebra Crossing), un proyecto de código abierto que se centra en la lectura y generación de códigos de barras y códigos QR en diferentes formatos. Además, permite crear representaciones gráficas de códigos de barras y códigos QR a partir de datos codificados, lo que puede ser útil en aplicaciones que necesitan generar estos códigos para su impresión o visualización en pantalla.

Material

'com.google.android.material:material:1.7.0'

Se utiliza para mejorar la apariencia visual de los elementos de la interfaz de usuario en aplicaciones Android, siguiendo las pautas de diseño de Material Design de Google. Por otro lado, proporciona una apariencia moderna y atractiva para botones, campos de texto y otros componentes.

Retrofit [30] [31]

'com.squareup.retrofit2:retrofit:2.9.0' 'com.squareup.retrofit2:converter-gson:2.9.0'

Estas dos dependencias se utilizan en combinación para realizar peticiones HTTP y transformarlas en formato JSON, al utilizar el convertidor Gson. Además, las respuestas en dicho formato se pueden convertir en objetos Kotlin, lo que resulta esencial en aplicaciones que interactúan con servicios web y API.

Picasso [32]

'com.squareup.picasso:picasso:2.8'

Es utilizada en aplicaciones Android para gestionar la carga, redimensionamiento y visualización de imágenes de manera eficiente y para garantizar una experiencia de usuario fluida. También simplifica la tarea de cargar imágenes desde recursos locales o fuentes en línea, además de ofrecer capacidades de almacenamiento en caché para mejorar el rendimiento y reducir el consumo de datos.

6.4.1. Detalles de implementación

En esta sección se abordarán las consideraciones iniciales y cuestiones clave que se han tenido en cuenta en el proceso de desarrollo de nuestra aplicación.

En el desarrollo de la aplicación móvil PocketMix, se han combinado varias herramientas y tecnologías para ofrecer a los usuarios una experiencia completa y funcional [2]. En primer lugar, se hace uso de Firebase Auth para permitir a los usuarios autenticarse en la aplicación, brindando opciones de inicio de sesión a través de cuentas de Google o correos electrónicos convencionales. Como Firebase Auth [21] no permite añadir campos como por ejemplo el género y la fecha de nacimiento, se ha usado como complemento el Cloud Firestore para poder tener campos adicionales en cada usuario, lo que enriquece la información disponible en la aplicación.

Para la gestión de tarjetas y tiendas, se ha implementado un backend utilizando Spring Boot y Spring Data [5], que aporta una sólida base para administrar estas funcionalidades de manera eficiente y escalable.

De forma independiente, se han creado algunos complementos web como un manual de ayuda [6], la página de descarga de la app o la capacidad de agregar puntos desde las tiendas.

Se ha diseñado cuidadosamente para garantizar un funcionamiento integral y preciso, permitiendo a los usuarios escanear tarjetas de fidelización y disfrutar de una experiencia completa en la gestión de sus puntos y recompensas.

6.5. Acciones seguidas en la implementación

Acción 1: Creación de un prototipo Android sin acceso a bases de datos en línea ni registro de usuarios

La etapa de prototipado en el desarrollo de PocketMix desempeña un papel esencial en la creación y validación de la interfaz de usuario y la funcionalidad de la aplicación antes de la implementación completa. Durante esta fase, se han seguido varias prácticas clave:

1. **Diseño de la Interfaz de Usuario (UI) [6]**: se ha diseñado la interfaz de usuario de PocketMix de manera detallada, teniendo en cuenta la usabilidad y la experiencia del usuario. Esto incluye la creación de diseños de pantalla, esquemas de color y elementos visuales que reflejan la identidad de la marca y aseguran una navegación intuitiva.

2. **Interacción de Usuario (UX)**: se ha trabajado en la experiencia del usuario para garantizar que las acciones y flujos de la aplicación sean lógicos y eficientes. Esto implica la disposición de botones, navegación entre pantallas y la forma en que los usuarios interactúan con las tarjetas de fidelización y los puntos.

3. **Validación de concepto**: se han utilizado prototipos interactivos para validar el concepto de la aplicación con usuarios de prueba. Esto ha permitido recopilar comentarios valiosos y realizar ajustes antes de avanzar en el desarrollo.

4. **Simulación de funcionalidad**: los prototipos también han permitido simular el comportamiento de la aplicación sin necesidad de conexión a bases de datos remotas ni autenticación de usuarios. Esto ha sido esencial para probar las características principales y la navegación antes de la implementación real.

5. **Retroalimentación y mejoras**: la retroalimentación recopilada durante el proceso de prototipado se ha utilizado para realizar mejoras en el diseño y la funcionalidad de la aplicación. Esto asegura que PocketMix cumpla con las expectativas de los usuarios.

Como conclusión, la fase de prototipado en PocketMix ha sido una etapa crucial para diseñar y validar la experiencia de interacción y las características clave de la aplicación antes de avanzar en el desarrollo completo. Esto garantiza que la aplicación cumpla con los estándares y satisfaga las necesidades del usuario.

Acción 2: Configuración inicial del proyecto en la plataforma Firebase

1. Para acceder a Firebase, simplemente hay que ingresar en `https://firebase.google.com/?hl=es` en tu navegador web.

2. Una vez en la página, hacer clic en *Ir a la consola* ubicado en la esquina superior derecha.

3. Accedemos a nuestra cuenta utilizando las credenciales de acceso.

4. Una vez dentro, veremos la siguiente pantalla donde podemos observar que ya disponemos de un proyecto denominado *proyectoTFG* asociado a nuestra cuenta.

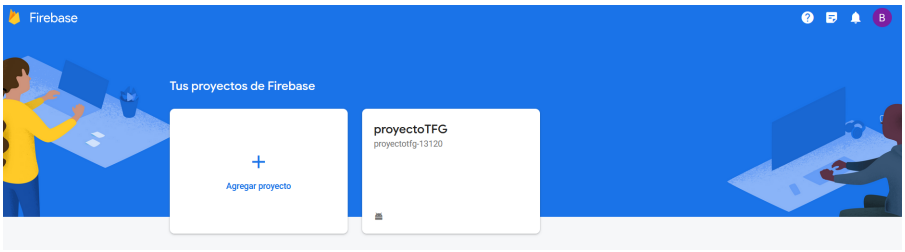

Figura 6.1: Portada Consola.

5. Para crear un nuevo proyecto hacemos clic en *Añadir proyecto*. A continuación, se mostrarán una serie de pantallas en las que podremos configurar aspectos generales, tales como el nombre y la integración de Google Analytics.

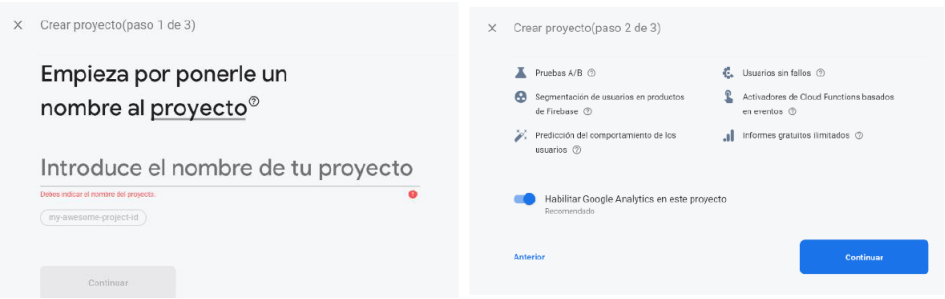

Figura 6.2: Creación proyecto Firebase.

6. Una vez que hayamos creado el proyecto y estemos en el entorno de Firebase, se divide en dos componentes distintos: un menú lateral que alberga todos los servicios ofrecidos por la plataforma y una pantalla principal que muestra información en función de nuestra selección en el menú.

Acción 3: Vinculación entre Firebase y la aplicación Android

1. Para continuar desde el último paso anterior, procederemos a incorporar Firebase en nuestra aplicación PocketMix [21]. Iniciaremos este proceso haciendo clic en el icono de Android, resaltado en la imagen siguiente, para iniciar la vinculación.

Figura 6.3: Vinculación con Firebase.

2. Se desplegará una pantalla en la que debemos completar una serie de pasos iniciales. El primer paso consiste en ingresar el nombre del paquete de nuestra aplicación para Android, proporcionarle un nombre descriptivo y, de manera opcional, agregar la firma digital.

Figura 6.4: Formulario FireAPP.

3. A continuación, procederemos con la descarga del archivo JSON que será necesario integrar en nuestro proyecto dentro de Android Studio.

Figura 6.5: Descarga del archivo json.

4. Para agregar el archivo *google-services.json*, debemos realizar esta incorporación en la raíz del proyecto de la aplicación dentro de Android Studio.

5. Para concluir el proceso de integración, debemos habilitar los servicios de Google (Google Services) y añadir las dependencias especificadas en los archivos *build.gradle*.


2. Luego, en el archivo `build.gradle` del **módulo (nivel de app)**, agrega los complementos `google-services` y cualquier SDK de Firebase que quieras usar en tu app:

Archivo de Gradle del módulo (nivel de app) (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.5.0')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Si usas la BoM de Firebase para Android, tu app siempre utilizará versiones compatibles de la biblioteca de Firebase.
Más información

3. Después de agregar el complemento y los SDK deseados, sincroniza tu proyecto de Android con archivos de Gradle.

Anterior Siguiente

Acción 4: Operaciones de inserción y extracción de datos [14] [15]

1. Una vez que hayamos completado la integración de Firebase en nuestra aplicación, podemos comenzar a desarrollar los métodos fundamentales que interactuarán con nuestra base de datos.

2. Añadir tarjetas de fidelización:

   - Mediante el escaneo:
     - Para escanear una nueva tarjeta , se ha implementado la clase llamada ***CameraScannerActivity*** que de forma reiterada procesa cada frame que recibe desde la cámara hasta encontrar un código de barras o QR válido, invocando al siguiente método ***processBarcode(barcodeList: List<Barcode>, scanner: BarcodeScanner)***.
     - Una vez que se verifica la validez de un código de barras o QR, la aplicación se comunica con el servidor para enviar los datos capturados. Luego, redirige al usuario a la pantalla principal para visualizar el listado de sus tarjetas.
     - Para comunicarnos con el servidor, se hace uso de la biblioteca ***Retrofit*** [30] [31], para gestionar las solicitudes de acuerdo al servicio seleccionado.
     - Con la llamada al método global ***RetrofitHelper.getRetrofit()*** obtenemos la instancia de *Retrofit* y mediante la función ***create(CardAPIService::class.java)*** instanciamos al servicio que deseamos usar.
     - Mediante el método ***addNewCard(cardRequest)*** enviamos los datos del objeto al servidor. En este caso, *Retrofit* se encarga de convertir el objeto Kotlin al Json de forma automática, y para la respuesta realiza el proceso inverso. Debido a que hacemos llamadas asíncronas, llamamos al método ***enqueue(object)*** que recibe por parámetro un objeto *Callback* con la respuesta del backend pudiendo llamar al

método ***onResponse()*** cuando la petición se ha realizado con éxito o al método ***onFailure()*** en caso de fallo.

- Desde el servidor, se utiliza la ruta *cards*, siguiendo el modelo de REST API. En este caso, se hace una solicitud *POST(/cards)* al servicio de tarjetas para que inserte la nueva tarjeta en la base de datos a través de un objeto repositorio.
- El método ***registerACard(cardRequest: CardRequest)*** del servicio de tarjetas se encarga de crear la entidad con todos sus campos. Si identifica que el número de tarjeta ya existe, activa una excepción que ocasiona que el servidor retorne el código de **error 409 (Conflicto)**. De esta forma el cliente puede proporcionar un mensaje más específico del error generado.

3. Consultar puntos de la tarjeta

- El proceso normal de gestión de puntos empieza en el momento en el que el usuario pasa por la tienda y la aplicación web, desarrollada con java script, hace una petición http a través de la función *fetch()* que devuelve una *promise* con el resultado de dicha petición.
- La primera parte de la cadena *promise* corresponde a la comprobación de la existencia de la tarjeta y luego continúa haciendo otra petición *fetch()* para la inserción de los puntos de la tarjeta.
- El backend recibirá la petición de puntos en el ***CardsController.kt*** a través del método ***addPoints()***.
- Una vez el usuario tiene sus puntos insertados, los podrá consultar en cualquier momento a través de la aplicación.
- Dentro del método ***onCreate()*** de la actividad ***PointsActivity***, se realizará una solicitud GET a través de *Retrofit*, utilizando la ruta */cards*. Esto nos proporcionará una lista de objetos de la clase ***PointsResponse*** como respuesta. Cabe destacar de esta clase que los puntos son siempre positivos debido a que utiliza el campo *received* para conocer si los puntos han sido ganados o descontados. Además, a través de este campo *received*, usado en la clase ***PuntosAdapterViewHolder***, que forma parte de ***PuntosAdapter***, e invocando al método ***updateWithPoint()***, se encarga de poner la visualización de los puntos de manera distinta según si el usuario ha ganado los puntos o los ha descontado.

Acción 5: Operaciones para modificar y eliminar información [16] [17]

1. Modificar tarjeta

- Cuando un usuario visualiza la información de una tarjeta, tiene disponible la opción de *Modificar tarjeta* para corregir su número de tarjeta o su tipo de código de barrar o código QR. El usuario puede actualizar esta información en el fragmento ***ModifyCardFragment*** de forma manual o escaneando nuevamente la tarjeta.
- En el caso de escanear nuevamente la tarjeta, llamamos a la clase ***CameraScannerActivityModify***, que es una clase heredada de ***CameraSacnnerActivity***. En dicha clase, el método ***processBarcode()*** está sobreescrito de forma que no va a crear una nueva tarjeta sino simplemente devolverla como una respuesta de la actividad.

- Una vez obtenidos los datos con una de las dos opciones, el usuario debe pulsar el botón para proceder con la actualización. El evento ***setOnClickListener*** valida los campos y llama al *Retrofit* para que haga una petición http usando el método ***PATCH(cards/id)***.
- Desde el lado del servidor backend recibe esta solicitud a través del punto de entrada ***patchCard()***, que llama al servicio de tarjetas mediante el método ***updateCard()*** para buscar la tarjeta correspondiente y actualizar su información.

2. Eliminar tarjeta

- Cuando el usuario visualiza la información de la tarjeta, selecciona la opción *Eliminar tarjeta*. Se muestra un cuadro de diálogo emergente de confirmación utilizando la clase ***MaterialAlertDialogBuilder*** de la librería de diseño de materiales de Android.
- Una vez que el usuario da su consentimiento, se procede a eliminar la tarjeta utilizando *Retrofit* al llamar al método DELETE en la ruta */cards/id* e indicando el identificador (id) de la tarjeta que se desea eliminar.
- En el lado del servidor backend se recibe la petición a través del punto de entrada ***DeleteCard()*** que llama al servicio de tarjeta *deleteCard()* para eliminar la tarjeta del repositorio.

6.5.1. Gestión de usuarios en Firebase

Un aspecto fundamental en nuestra aplicación se refiere al manejo y administración de los usuarios que hacen uso de la plataforma. En esta sección, exploraremos las tres funcionalidades clave en cuanto a la interacción de los usuarios: la incorporación de nuevos miembros, el proceso de inicio de sesión y la opción de cerrar sesión.

En un primer paso, se debe declarar una instancia de Firebase Auth para habilitar la funcionalidad de autenticación en la aplicación. Esta declaración se encuentra en las propiedades de las clases que la utilicen, ***private val firebaseAuth = FirebaseAuth.getInstance()***

Añadir nuevos usuarios

Para añadir nuevos usuarios a la aplicación, se ha utilizado Firebase UI por su funcionalidad completa tanto del login como del registro del usuario, que permite hacerlo a través de email y password o mediante las cuentas de Google. Esta función tiene la responsabilidad de crear un nuevo usuario en la plataforma mediante el uso de un par de credenciales que incluyen correo electrónico y contraseña. Esto permite agilizar el trabajo a la hora de implementar esta fase.

Además, mediante ***registerForActivityResult()***, observamos la respuesta de la actividad de Firebase UI de tal forma que podemos saber si el usuario se ha podido registrar registrar con éxito o no. En caso de que el usuario haya completado exitosamente esta acción, se procede a buscar sus tarjetas de fidelización registradas.

Por otro lado, con el siguiente método ***AddAuthStateListener()***, Firebase Auth se encarga de desencadenar un código en respuesta a cualquier cambio en la autenticación del usuario, ya

sea un inicio de sesión o un cierre de sesión. Esto permite actualizar la pantalla inicial, realizando comprobaciones para determinar si el usuario es nuevo o no. En consecuencia, se puede proceder a crear su registro en Firebase Firestore y mostrar la pantalla de *érminos y condiciones*.

Si la operación de inserción presenta un fallo, se evaluarán dos posibles escenarios: en el primero, podría existir un usuario registrado con el mismo correo electrónico en la aplicación; en el segundo, podría deberse a un fallo genérico causado por factores externos ajenos al proceso.

Autentificación de usuarios

Después de que el usuario se haya registrado en la aplicación, tendrá la opción de iniciar sesión utilizando las mismas credenciales de correo electrónico y contraseña que proporcionó durante el registro o mediante la cuenta de Google. Para llevar a cabo este proceso de autenticación, se recurre al proceso de Firebase UI de forma que el código está ya implementado y evita errores lógicos.

Cerrar sesión

Cuando un usuario autenticado desea finalizar su sesión en la aplicación, el método empleado es ***FirebaseAuth.getInstance().signOut()***, una vez que se procesa correctamente el *signOut*, el Firebase Auth llama al *Listener* que detecta la falta de usuario, y lanza en pantalla el Firebase UI para que el usuario haga un login.

6.6. Estructura interna de proyecto

En esta sección, analizamos la organización interna del proyecto, que se centra en cómo los archivos se estructuran en directorios y paquetes. Esta estructura está diseñada para optimizar el desarrollo de nuevas funcionalidades y la realización de tareas de mantenimiento en la aplicación, lo que resulta en una mayor eficiencia y facilidad para los desarrolladores.

Front-end

- */pocketMix/app/src*: es el directorio raíz de toda la base del proyecto Android, donde cuelgan subdirectorios para recursos, pruebas unitarias, código fuente.
- */pocketMix/app/src/main*: es donde se encuentran todos los archivos que incluyen el código fuente de la aplicación y los recursos necesarios para crear y personalizar la aplicación. Cabe destacar que se encuentra el archivo *AndroidManifest.xml* que es el que registra las actividades y los permisos. Asimismo, este directorio se subdivide en subdirectorios para organizar de manera específica diferentes componentes, manteniendo todo ordenado y estructurado.
- */pocketMix/app/src/main/java/com/example/miaplicacion*: los archivos con extensión *.java* que contienen la lógica fundamental de la aplicación se encargan de las operaciones y reglas de negocio de la misma. Estos archivos están agrupados en subdirectorios según la finalidad de su implementación, es decir, se organizan en carpetas separadas dependiendo de su función específica:

  - */api*: clases relacionadas con las peticiones http al backend.
  - */aplication*: clase que representa la aplicación.
  - */constants*: paquete para guardar las constantes globales.
  - */models*: clases que son modelos de datos.
  - */utils*: clases auxiliares que se emplean en determinadas partes de la aplicación.
  - */view/adapters*: clases adaptadoras de los *recycler views*.
  - */view/ui*: activities y los fragments.
  - */viewmodel*: modelos de las vistas.

- */pocketMix/app/src/main/res*: contiene todos los archivos que tienen la responsabilidad de definir cómo se ven las diferentes partes de la aplicación, que en su mayoría son archivos XML. También almacena los recursos utilizados, como textos, paleta de colores, temas e imágenes.

Back-end

- */pocketmix/src*: este directorio raíz es el núcleo principal de todo el proyecto Spring boot, y a partir de aquí se extienden subdirectorios para organizar pruebas unitarias y código fuente, estableciendo así una estructura organizativa fundamental.
- */pocketmix/src/main*: códigos fuente y el archivo de propiedades.
- */pocketmix/src/main /kotlin/com/proyectotfg/pocketmix*: paquete raíz del proyecto.
- */controllers*: clases controladores, es decir la capa de presentación.
- */entities*: las entidades conectadas a la base de datos.
- */exceptions*: excepciones propias, normalmente con un código de error http.
- */repositories*: clases que se encargan de manejar las bases de datos.
- */requests*: modelos de peticiones que acepta la aplicación.
- */responses*: modelos de respuesta que van a ser devueltos al usuario.
- */services*: clases encargadas de tener la lógica de negocio.
- */pocketmix/src/main/resources*: archivo *application.properties* que es el que contiene la configuración.

Parte web

- */TFG_ Parte_ Web*: diferentes archivos que fueron desarrollados en la parte web como por ejemplo; ayuda, descargas, manual y la tienda. Esta carpeta se divide en tres subcarpetas:
  - */img*: para las imágenes.
  - */css*: hojas de estilos, principalmente, son archivos de la librería bootstap.
  - */js*: archivos de java script, tanto de bootstrap como de la aplicación.

Capítulo 7

Pruebas

7.1. Pruebas de caja blanca

Las pruebas de caja blanca son una fase crítica que se ejecuta durante la etapa de implementación de un proyecto, ya que se centran en el análisis minucioso del código fuente con el propósito de evaluar y validar la funcionalidad interna de cada módulo que constituye la aplicación. Este proceso, fundamental para garantizar la integridad y la corrección del software, es llevado a cabo por los propios desarrolladores o por expertos capacitados en esta tarea. Durante estas pruebas, se exploran los posibles caminos lógicos del programa, permitiendo una inspección detallada de su comportamiento interno para detectar posibles fallos o deficiencias en el código. Este enfoque en la transparencia del código y en su funcionamiento intrínseco es esencial para lograr una aplicación robusta y confiable.

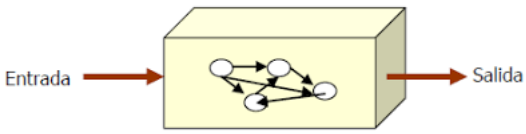

Figura 7.1: Pruebas de caja blanca.

Dado el considerable alcance de este proyecto, se ha llevado a cabo un número sustancial de pruebas. Por lo tanto, en este informe, se ofrece un resumen que destaca los aspectos más significativos sometidos a este riguroso proceso de evaluación:

- Evaluación de reconocimiento y distinción de roles, abordando la funcionalidad tanto para usuarios no registrados como para usuarios autenticados.
- Verificación de la adaptación de la interfaz a diferentes roles presentes en la plataforma, asegurando una experiencia de usuario coherente y efectiva.
- Comprobación exhaustiva de la funcionalidad asociada a cada uno de los roles, garantizando que todas las características operen según lo esperado.

- Control de la sesión para verificar que los usuarios puedan interactuar de manera segura y sin interrupciones.
- Evaluación del comportamiento del sistema en situaciones de falta de conexión, asegurando su capacidad para funcionar de manera adecuada en entornos offline.
- Inspección individual del funcionamiento y comunicación de cada servicio externo utilizado por la aplicación, con el objetivo de identificar posibles fallos o problemas de integración.
- Verificación de las peticiones a la base de datos en tiempo real, garantizando la precisión y la eficiencia de la información recuperada.
- Gestión de la asincronía inherente a las interacciones con la base de datos en todo tipo de peticiones, asegurando un flujo de datos coherente.
- Comprobación minuciosa de la correspondencia de datos entre la base de datos y la aplicación móvil, incluyendo la integridad de las clases y objetos Java.
- Evaluación de los formularios de entrada de datos, que implica la validación de campos, tareas de autocompletado y el filtrado de datos para garantizar la precisión y seguridad en la entrada de información.
- Verificación de la distinción entre los posibles sistemas de tarjetas, asegurando que las opciones de añadir tarjetas nuevas estén claramente diferenciadas y funcionen sin problemas.

Estos pasos representan un enfoque integral para garantizar la calidad y la eficacia de la aplicación en desarrollo, abordando aspectos críticos de funcionalidad, interacción y rendimiento.

7.2. Pruebas de caja negra

Este tipo de pruebas tienen como objetivo verificar si la aplicación cumple con todos los requisitos y funcionalidades establecidos para su funcionamiento. En este proceso, el evaluador no necesita poseer conocimientos previos sobre la estructura y los procesos internos del sistema. En su lugar, se enfoca en analizar y estudiar los resultados de las respuestas en función de las entradas recibidas.

Dado que este proyecto es de gran tamaño, se han llevado a cabo un elevado número de pruebas. Por tanto, en este documento se destacan únicamente algunas de las pruebas esenciales relacionadas con la funcionalidad básica que el sistema debe cumplir.

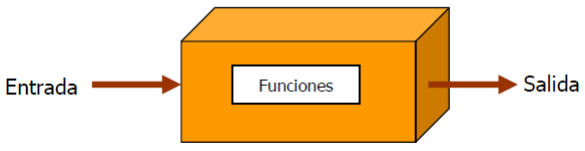

Figura 7.2: Pruebas de caja negra.

PCN-01: Registrar Usuario	
Objetivo de la prueba	Valorar la posibilidad de crear nuevos usuarios desde la aplicación móvil.
Precondiciones	La inexistencia de una cuenta de usuario en el sistema con las mismas credenciales.
Datos de entrada	▪ Email: bianca@outlook.com ▪ Nombre y apellidos de usuario: Bianca Illas ▪ Contraseña: admin123
Respuesta esperada	La creación de cuenta de usuario en el sistema se ha realizado con éxito.
Resultado final	VÁLIDO.

Tabla 7.1: Prueba de caja negra - Registrar usuario

PCN-02: Iniciar Sesión	
Objetivo de la prueba	Valorar el acceso a la cuenta a través de sus credenciales.
Precondiciones	El usuario debe estar previamente registrado en el sistema.
Datos de entrada	▪ Email: bianca@outlook.com ▪ Contraseña: admin123
Respuesta esperada	Al hacer clic en una de las dos opciones de *iniciar sesión*, el sistema redirecciona al usuario a la página principal y activa las funciones específicas del usuario autenticado.
Resultado final	VÁLIDO.

Tabla 7.2: Prueba de caja negra - Iniciar Sesión

PCN-03: Añadir tarjeta	
Objetivo de la prueba	Validar si el sistema añade una tarjeta de entre las tres opciones posibles: escaneo, manual o adjunto.
Precondiciones	El usuario debe tener una tarjeta a su disposición.
Datos de entrada	▪ Añadir nombre del comercio. ▪ Seleccionar una de las tres opciones posibles. ▪ En el caso de que sea manual se pide introducir el campo número de tarjeta y seleccionar el tipo de tarjeta.
Respuesta esperada	No se espera ningún mensaje de confirmación. Redirige directamente al usuario a la pantalla principal de la aplicación donde podrá visualizar sus tarjetas guardadas.
Resultado final	VÁLIDO.

Tabla 7.3: Prueba de caja negra - Añadir tarjeta

PCN-04: Consultar tarjeta	
Objetivo de la prueba	Validar que se muestra en pantalla la información y las diferentes opciones que se pueden hacer con la tarjeta.
Precondiciones	Debe existir una tarjeta registrada.
Datos de entrada	-
Respuesta esperada	Mostrar la información de la tarjeta por pantalla al usuario.
Resultado final	VÁLIDO.

Tabla 7.4: Prueba de caja negra - Consultar tarjeta

PCN-05: Consultar puntos	
Objetivo de la prueba	Comprobar que el sistema calcula y muestra el historial de puntos por pantalla de manera correcta.
Precondiciones	La tienda debe registrar los puntos previamente.
Datos de entrada	▪ Añadir nombre del comercio: ▪ Seleccionar una de las tres opciones posibles. ▪ En el caso de que sea manual se pide introducir el campo número de tarjeta y seleccionar el tipo de tarjeta.
Respuesta esperada	Mostrar el histórico de puntos calculado de forma correcta.
Resultado final	VÁLIDO.

Tabla 7.5: Prueba de caja negra - Consultar puntos

PCN-06: Modificar tarjeta	
Objetivo de la prueba	Validar que la información a modificar se haya actualizado de manera correcta en la base de datos.
Precondiciones	Debe existir una tarjeta registrada en la aplicación.
Datos de entrada	▪ Número de tarjeta. ▪ Tipo de código de la tarjeta (código de barras o código QR).
Respuesta esperada	Los datos deben actualizarse correctamente tanto en la aplicación como en la base de datos.
Resultado final	VÁLIDO.

Tabla 7.6: Prueba de caja negra - Modificar tarjeta

PCN-07: Atención al cliente	
Objetivo de la prueba	Validar que se muestra correctamente la información de contacto de cada comercio.
Precondiciones	Debe existir una tarjeta del comercio registrada en la aplicación.
Datos de entrada	-
Respuesta esperada	Mostrar la información del comercio: teléfono, email, sitio web.
Resultado final	VÁLIDO.

Tabla 7.7: Prueba de caja negra - Atención al cliente

PCN-08: Eliminar tarjeta	
Objetivo de la prueba	Validar si el sistema borra una tarjeta de forma permanente de la aplicación.
Precondiciones	El usuario debe haber registrado una tarjeta previamente en la aplicación.
Datos de entrada	El usuario confirma que desea eliminar una tarjeta del listado.
Respuesta esperada	Se elimina la tarjeta de forma correcta, actualizando el listado principal de tarjetas del usuario.
Resultado final	VÁLIDO.

Tabla 7.8: Prueba de caja negra - Eliminar tarjeta

PCN-09: Modificar perfil	
Objetivo de la prueba	Validar que el sistema actualiza la información relativa al usuario.
Precondiciones	El usuario debe tener una cuenta activa.
Datos de entrada	▪ Cambiar foto de perfil. ▪ Nombre: Bianca. ▪ Fecha de nacimiento. ▪ Teléfono. ▪ Género.
Respuesta esperada	La información del usuario debe quedar actualizada.
Resultado final	VÁLIDO.

Tabla 7.9: Prueba de caja negra - Modificar perfil

PCN-10: Modificar contraseña	
Objetivo de la prueba	Validar que el sistema actualiza la nueva contraseña propuesta por el usuario.
Precondiciones	El usuario debe estar dado de alta en la aplicación con sus correspondientes credenciales.
Datos de entrada	▪ Contraseña actual: pepito ▪ Nueva contraseña: pepito_2 ▪ Confirmar contraseña: pepito_2
Respuesta esperada	La contraseña del usuario debe quedar actualizada de manera correcta.
Resultado final	VÁLIDO.

Tabla 7.10: Prueba de caja negra - Modificar contraseña

PCN-11: Cerrar sesión	
Objetivo de la prueba	Validar que el sistema permite a cualquier usuario autenticado cerrar sesión en la aplicación.
Precondiciones	El usuario debe estar registrado en la aplicación.
Datos de entrada	-
Respuesta esperada	La aplicación muestra la pantalla principal del login/registro.
Resultado final	VÁLIDO.

Tabla 7.11: Prueba de caja negra - Cerrar sesión

PCN-12: Eliminar cuenta	
Objetivo de la prueba	Validar que el sistema permite borrar una cuenta de perfil de forma permanente de la aplicación.
Precondiciones	El usuario debe tener una cuenta registrada en nuestro sistema.
Datos de entrada	-
Respuesta esperada	Se elimina la cuenta correctamente, actualizando el listado de los usuarios.
Resultado final	VÁLIDO.

Tabla 7.12: Prueba de caja negra - Cerrar sesión

PCN-13: Preferencias	
Objetivo de la prueba	Validar que el sistema guarde la personalización de los parámetros preferidos por el usuario tales como: mostrar nombres de tarjeta, personalizar vista tarjeta o habilitar un bloqueo de pantalla.
Precondiciones	▪ El usuario debe estar autenticado en la aplicación. ▪ El usuario debe tener tarjetas registradas.
Datos de entrada	▪ Habilitar nombre tarjetas. ▪ Personalizar vista tarjetas: una o dos columnas. ▪ Habilitar pin bloqueo. ▪ Introducir pin bloqueo: 1234
Respuesta esperada	El sistema muestra, en el listado, los nombres de cada tarjeta. en una columna o en dos, en función de lo elegido por el usuario. Por último, se aplicará el pin de bloqueo de la aplicación con el código escogido.
Resultado final	VÁLIDO.

Tabla 7.13: Prueba de caja negra - Preferencias

PCN-14: Recomendar a un amigo	
Objetivo de la prueba	Validar si el sistema comparte de forma correcta el enlace de la aplicación mediante diferentes vías.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	Enlace de descarga de la aplicación: vía sms.
Respuesta esperada	-
Resultado final	VÁLIDO.

Tabla 7.14: Prueba de caja negra - Recomendar a un amigo

PCN-15: Permisos	
Objetivo de la prueba	Comprobar que el sistema haya registrado los permisos aceptados por el usuario.
Precondiciones	▪ El usuario debe estar autenticado en la aplicación. ▪ El usuario debe dar su consentimiento a los permisos.
Datos de entrada	▪ Cámara: Aceptar ▪ Galería de fotos: Aceptar
Respuesta esperada	El sistema muestra los permisos aceptados correctamente por pantalla.
Resultado final	VÁLIDO.

Tabla 7.15: Prueba de caja negra - Permisos

PCN-16: Ayuda	
Objetivo de la prueba	Comprobar que el sistema redirige al manual de ayuda en la página web.
Precondiciones	El usuario debe estar dado de alta en la aplicación.
Datos de entrada	-
Respuesta esperada	Visualizar correctamente el manual de ayuda desde la página web.
Resultado final	VÁLIDO.

Tabla 7.16: Prueba de caja negra - Ayuda

PCN-17: Contactar con soporte	
Objetivo de la prueba	Validar que el sistema envía correctamente el formulario con la incidencia redactada por el usuario.
Precondiciones	▪ El usuario debe estar autenticado en la aplicación. ▪ Al usuario le debe haber surgido algún inconveniente o duda con nuestra aplicación.
Datos de entrada	Redacción de la incidencia vía correo electrónico: *No se visualiza de manera correcta el listado de las tarjetas.*
Respuesta esperada	Ninguna. El soporte técnico funciona como una simulación.
Resultado final	VÁLIDO.

Tabla 7.17: Prueba de caja negra - Contactar con soporte

PCN-18: Consultar acerca de PocketMix	
Objetivo de la prueba	Validar si el sistema muestra correctamente la información relacionada con la aplicación como por ejemplo la versión.
Precondiciones	El usuario debe estar dado de alta en la aplicación.
Datos de entrada	-
Respuesta esperada	El sistema muestra correctamente la información relevante de la aplicación.
Resultado final	VÁLIDO.

Tabla 7.18: Prueba de caja negra - Consultar acerca de PocketMix

Capítulo 8

Manuales

8.1. Guía de instalación y puesta en marcha.

La aplicación desarrollada solo requiere la instalación en un dispositivo para comenzar a utilizarla, sin necesidad de pasos adicionales. Para llevar a cabo la instalación, siga estos pasos:

1. Asegurarse de habilitar la opción en el dispositivo móvil que permite la instalación de aplicaciones de fuentes desconocidas (aquellas que no provienen de tiendas de aplicaciones oficiales, como Google Play).
2. Descargar el archivo .apk de la aplicación directamente en su dispositivo o transfiera el archivo a través de un cable USB o una tarjeta SD.
3. Una vez tenga el archivo de instalación, simplemente hay que ejecutarlo para iniciar el proceso.
4. En caso de que la aplicación haga uso de servicios de Google y estos estén desactualizados por alguna razón, el sistema pedirá al usuario que los actualice.
5. Después de completar la instalación, la aplicación requerirá una serie de permisos para llevar a cabo ciertas funciones, como el acceso a la galería de imágenes y a la cámara de nuestro dispositivo móvil.

Con estos pasos, la aplicación queda instalada al 100 % en el dispositivo, con todas sus funciones y permisos activados y disponibles en cualquier momento.

8.2. Guía de usuario

En esta sección, después de la evaluación de todos los aspectos técnicos relacionados con la funcionalidad, requisitos y diseño, ahora es el momento de presentar y explicar el funcionamiento fundamental de la aplicación de una manera más accesible y fácil de entender. Este enfoque proporciona a los usuarios el punto de partida para aprender a utilizar la herramienta de manera efectiva y también les brinda un recurso de soporte para consultar cualquier pregunta que puedan tener sobre su uso en cualquier momento.

8.2.1. Cuenta

Registro de usuario

Con una cuenta de PocketMix podrá restaurar las tarjetas en caso de necesitar reinstalar la aplicación, cambiar de dispositivo o incluso cambiar el sistema operativo. Además, la sincronización de las tarjetas en múltiples dispositivos es posible. Por lo tanto, la recomendación es registrarse con una cuenta de usuario para garantizar la seguridad de los datos y facilitar el acceso desde todos los dispositivos móviles en cualquier momento.

Cómo darnos de alta

1. Abrir la aplicación de PocketMix y seleccionar *Iniciar sesión*. Este botón también es válido.
2. Seleccionar uno de los dos métodos disponibles: correo normal o con los servicios de Google.
3. Si elegimos el método registrarse con correo normal, se habilitará el formulario para que el usuario escriba una contraseña.
4. A continuación, ya estaría registrado y aparecería su sesión iniciada en la configuración de su cuenta de PocketMix. Por tanto, ya podría acceder desde todos sus dispositivos simplemente al iniciar sesión de nuevo.

Inicio de sesión

Si desea acceder a su cuenta existente para restaurar sus tarjetas guardadas, siga estos pasos:

1. Abrir la aplicación de PocketMix.
2. Seleccionar *Iniciar sesión*.
3. Seleccionar uno de los dos métodos disponibles.

Cerrar sesión

Para cerrar sesión desde su cuenta existente:

1. Abrir la aplicación de PocketMix.
2. Existen dos opciones para cerrar la sesión:
   - Desde el botón *Ajustes* y dentro de este apartado, dándole a *Cerrar sesión*.
   - Haciendo clic en *Ajustes*, seleccionando la opción *Cuenta* donde aparece la opción de *Cerrar sesión*.
3. Seleccionar uno de los dos métodos disponibles.

Eliminar cuenta

1. Abrir *Ajustes*.
2. Hacemos clic en la opción *Cuenta* y dentro de este apartado, seleccionamos el botón *Eliminar cuenta*.
3. En la pantalla siguiente, confirmar si realmente deseamos eliminar la cuenta.
4. Presionar el botón *Eliminar cuenta* para confirmar o *Mantener cuenta* si no queremos eliminarla.

Cambiar contraseña

1. Si te has registrado en PocketMix con Google y necesitas cambiar la contraseña, tendrás que hacerlo directamente en tu cuenta de Google ya que a través de nuestra aplicación no está permitido.
2. Si el registro ha sido mediante el correo normal, tienes que ir a *Ajustes*, *Cuenta* e ir a la opción *Modificar contraseña*. Esto redirige a otra pantalla para completar los campos *Contraseña antigua* y *Contraseña nueva*.

Modificar perfil

1. Seleccionamos *Ajustes* y elegimos la opción *Cuenta*.
2. Actualizamos la información de los campos a modificar.

8.2.2. Uso de PocketMix

Agregar tarjetas

Desde la ventana principal, seleccionar el botón *Añadir tarjeta* que se encuentra en la parte inferior derecha.

Escanear

1. Añadir nombre del comercio y, de manera opcional, alguna nota relevante.
2. Escanear el código de barras utilizando la cámara de su dispositivo móvil inteligente. Para acceder a la cámara, deberá aceptar los permisos. Además, podrá habilitar de luminosidad para precisar un mejor escaneo.

Manualmente

1. Añadir nombre del comercio y, de manera opcional, alguna nota relevante.
2. Seleccionar el tipo de número que vamos a introducir: QR o código de barras.
3. Introducir manualmente el número de la tarjeta.
4. A partir de aquí, se genera automáticamente el código.

Adjuntar desde un archivo de la galería

1. Añadir nombre del comercio y, de manera opcional, alguna nota relevante.
2. Aceptar los permisos para que la aplicación pueda acceder a la galería de fotos y pueda seleccionar el archivo deseado.

Marcar tarjetas favoritas

Desde la ventana principal, se muestra un listado de tarjetas. Seleccionando el icono ♡ en la parte superior derecha de cada tarjeta, marcamos dicha tarjeta como favorita y pasa a formar parte del listado de *tarjetas favoritas*.

Consultar tarjetas

1. Desde la ventana principal, seleccionamos la tarjeta a consultar.
2. En la nueva ventana, se muestra el número de la tarjeta junto con su código de barras y el nombre del comercio.

Consultar puntos

1. Desde la ventana principal, seleccionamos la tarjeta a consultar.
2. En la nueva ventana, seleccionamos la opción *Puntos*.
3. Nos aparece una nueva ventana, donde se muestran los puntos obtenidos o descontados del comercio.

Modificar tarjeta

1. Desde la ventana principal, seleccionamos la tarjeta a modificar.
2. En la nueva ventana, seleccionamos el botón *Modificar* que nos abre una pestaña nueva.
3. Podemos modificar el campo de forma manual o escaneando la tarjeta mediante el icono de cámara 📷 y seleccionando el tipo de código: QR o código de barras.
4. Seleccionamos *Actualizar tarjeta* o *Cancelar*.

Atención al cliente

item Desde la ventana principal seleccionamos el botón *Atención al cliente* que nos abre una pestaña nueva donde vienen reflejados las diferentes vías de contacto de cada comercio: email, teléfono, etc.

Eliminar tarjeta

1. Desde la ventana principal, seleccionamos la tarjeta a eliminar.
2. En la nueva ventana, seleccionamos el botón *Eliminar tarjeta*.
3. La aplicación nos muestra un pop-up para confirmar si realmente deseamos o no eliminar la tarjeta, seleccionamos *Sí* o *No* respectivamente.

Buscar tarjeta

Desde la ventana principal, haciendo clic en la barra de búsqueda, introducimos el nombre del comercio que queremos buscar.

Ordenar tarjetas

Desde la ventana principal, haciendo clic en el icono de ordenación, podemos ordenar nuestras tarjetas por: *más usadas, usadas recientemente, añadidas recientemente y alfabéticamente*.

Previsualización de tarjetas

1. Dentro de *Ajustes*, seleccionamos la opción *Preferencias*.
2. Nos redirige a una nueva ventana donde podemos habilitar los siguientes parámetros:
   - Mostrar nombre tarjetas.
   - Personalizar vista tarjetas (en una o dos columnas).
   - Pin de bloqueo de la aplicación. Se habilita la opción de introducir pin con un número de 4 cifras.

Permisos

1. Dentro de *Ajustes*, seleccionamos la opción *Permisos*.
2. Nos redirige a una nueva ventana donde podemos configurar los permisos de la aplicación y ver si están habilitados los de la cámara y la galería de fotos.

Soporte

1. Dentro de *Ajustes*, seleccionamos la opción *Soporte*.
2. La aplicación nos muestra un pop-up para seleccionar los *Recomendados*: gmail, correo normal o sms y redactar la incidencia a soporte técnico.

Recomendar a un amigo

1. Dentro de *Ajustes*, seleccionamos la opción *Recomendar a un amigo*.
2. La aplicación nos muestra un pop-up para seleccionar los *Recomendados* la manera en la que queremos compartir el enlace de descarga de la aplicación: gmail, correo normal o sms.

Ayuda

1. Dentro de *Ajustes*, seleccionamos la opción *Ayuda*.
2. La aplicación nos redirige al navegador web donde nos muestra las diferentes opciones de ayuda.

Acerca de PocketMix

1. Dentro de *Ajustes*, seleccionamos la opción *Acerca de PocketMix*.
2. La aplicación nos abre una nueva ventana con la información de la versión de la aplicación y los derechos de autor.

Capítulo 9

Conclusiones

En conclusión, la creación y desarrollo de la aplicación PocketMix ha sido un proyecto integral y exitoso donde se ha aprovechado de manera efectiva y nos ha obligado a adquirir más conocimiento y formación en las herramientas y tecnologías más avanzadas en el ámbito de desarrollo de software. La elección de Android Studio y el lenguaje Kotlin como plataforma de desarrollo, permite una experiencia de usuario fluida y amigable, mientras que la integración de una API REST externa basada en Spring Data, Spring Boot y MySQL brinda una gestión eficiente de datos y una sólida conectividad. Esta API se ha diseñado con el propósito de gestionar de manera óptima las tarjetas de fidelización en la aplicación PocketMix. Mediante endpoints cuidadosamente diseñados, la API permite la creación, lectura, actualización y eliminación de tarjetas, garantizando así un control completo para los usuarios. Además, la utilización de Spring Data ha simplificado la interacción con la base de datos MySQL, facilitando operaciones como la búsqueda rápida y la actualización de puntos. La implementación de Spring Boot, por su parte, ha asegurado una arquitectura modular y eficiente, facilitando el despliegue y la escalabilidad de la API.

La implementación de tres páginas web independientes ha sido clave para ampliar significativamente la funcionalidad y la accesibilidad de PocketMix. Estas páginas web, desarrolladas con Bootstrap, CSS, HTML y JavaScript, han sido diseñadas con propósitos específicos. La primera página web proporciona a las tiendas una interfaz intuitiva para registrar los puntos de las tarjetas de fidelización. La segunda página web permite que los usuarios descarguen la aplicación, garantizando un acceso fácil y directo a la herramienta desde cualquier dispositivo de forma ficticia. Finalmente, la tercera página web ofrece un manual de ayuda detallado, proporcionando a los usuarios recursos claros y concisos para sacar el máximo provecho de todas las características de PocketMix.

En el núcleo de la aplicación se encuentra la funcionalidad principal de escaneo de tarjetas de fidelización, ofreciendo una solución conveniente y eficaz para usuarios y empresas por igual. La sinergia entre la aplicación móvil y las plataformas web auxiliares crea un ecosistema completo y coherente que mejora la experiencia del usuario.

En resumen, PocketMix cumple con su propósito principal de escanear tarjetas de fidelización

y destaca por su enfoque integral, su diseño cuidadoso y su integración de tecnologías.

9.1. Posibles mejoras

Considerando la estructura existente de PocketMix, hay varias áreas en las que podríamos explorar mejoras para potenciar aún más la experiencia del usuario y la eficiencia operativa:

1. **Integración de tecnologías emergentes**: explorar la integración de tecnologías emergentes como la realidad aumentada para el proceso de escaneo de tarjetas, ofreciendo una experiencia más interactiva y moderna para los usuarios.

2. **Implementación de un sistema de recompensas personalizado**: desarrollar un sistema de recompensas más personalizado y adaptable, utilizando algoritmos de aprendizaje automático para analizar los patrones de comportamiento de los usuarios y sugerir recompensas más relevantes y atractivas.

3. **Mejora de la Interfaz de Usuario (UI)**: refinar la interfaz de usuario de la aplicación y las páginas web para garantizar una experiencia más intuitiva y atractiva. La implementación de un diseño centrado en el usuario puede facilitar la navegación y aumentar la participación.

4. **Ampliación de funcionalidades en la API**: expandir las capacidades de la API REST para incluir funciones avanzadas, como la integración con sistemas de pago móvil y la posibilidad de gestionar ofertas exclusivas directamente desde la aplicación.

5. **Incorporación de elementos gamificados**: introducir elementos gamificados en la aplicación para hacer que la participación en los programas de lealtad sea más divertida y atractiva. Pueden incluir desafíos, logros y tablas de clasificación.

6. **Desarrollo de una plataforma analítica**: implementar una plataforma analítica que permita a los comercios obtener información detallada sobre el rendimiento de sus programas de lealtad, facilitando la toma de decisiones informadas y estratégicas.

7. **Expansión de la plataforma a múltiples dispositivos**: considerar la expansión de PocketMix a plataformas adicionales, como iOS, para llegar a un público más amplio y diversificado.

Estas posibles mejoras, buscan no solo fortalecer la funcionalidad existente sino también anticiparse a las tendencias del mercado y las expectativas cambiantes de los usuarios.

Bibliografía

[1] Bill Phillips, Chris Stewart and Kristin Marsicano. *Android Programming: The Big Nerd Ranch Guide.* Big Nerd Ranch Guides. Enero 2022.

[2] Namrata Bandekar, Darryl Bayliss, Tom Blankenship, Fuad Kamal. *Android Apprentice (Second Edition) Beginning Android Development with Kotlin.* Raywenderlich Tutorial Team. Mayo 2019.

[3] Irina Galata, Joe Howard, Ellen Shapiro. *Kotlin Apprentice (Second Edition) Beginning Programming with Kotlin.* Raywenderlich Tutorial Team. Octubre 2019.

[4] Neil Smyth. *Firebase Essentials - Android Edition.* Createspace Independent Pub. 2017.

[5] Felipe Gutiérrez. *Pro Spring Boot 2.* Apress. Diciembre 2018.

[6] Jennifer Robbins. *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.* O'Reilly. Mayo 2018.

Webgrafía

[7] Nielsen:
https://www.reasonwhy.es/actualidad/estudio-nielsen-fidelidad-marca-consumidor-espana
Último acceso: Octubre 2022

[8] Documentación oficial Spring: https://spring.io/guides
Último acceso: Octubre 2022

[9] Baeldung: https://www.baeldung.com/
Último acceso: Octubre 2022

[10] freeCodeCamp: https://www.freecodecamp.org/
Último acceso: Junio 2023

[11] W3Schools: https://www.w3schools.com/
Último acceso: Mayo 2023

[12] Mozilla Developer Network (MDN): https://developer.mozilla.org/
Último acceso: Mayo 2023

[13] Bootstrap: https://getbootstrap.com
Último acceso: Mayo 2023

[14] Material Design Android: https://material.io/develop/android
Último acceso: Junio 2023

[15] Android Developers: https://developer.android.com/
Último acceso: Julio 2023

[16] Desarrollador Android: https://desarrollador-android.com/
Último acceso: Junio 2023

[17] Android Tutorials:
https://devofandroid.blogspot.com/2018/11/send-email-using-intent-android-studio.html
Último acceso: Junio 2023

[18] Stack Overflow: https://stackoverflow.com/
Último acceso: Septiembre 2023

[19] Github: https://github.com
Último acceso: Junio 2023

[20] Píldoras informáticas:
https://www.pildorasinformaticas.es/politica-de-privacidad/
Último acceso: Enero 2023

[21] Firebase Authentication | Firebase Google: https://firebase.google.com/docs/auth
Último acceso: Mayo 2023

[22] Stocard: https://stocardapp.com/en/de
Último acceso: Septiembre 2022

[23] FidMe: https://www.fidme.com/
Último acceso: Septiembre 2022

[24] Mobile-pocket: https://hub.mobile-pocket.com/en/
Último acceso: Septiembre 2022

[25] Metodología incremental:
https://4tesosite.wordpress.com/modelo-iterativo-incremental/
Último acceso: Septiembre 2022

[26] Modelo iterativo incremental:
https://danelly1236.files.wordpress.com/2013/12/modelo-iterativo-incremental-presentacion
pdf
Último acceso: Octubre 2022

[27] Desarrollo iterativo incremental:
https://es.slideshare.net/noriver/desarrollo-iterativo-e-incremental
Último acceso: Octubre 2022

[28] Material Design: https://www.geeksforgeeks.org/material-design-date-picker-in-android/
Último acceso: Junio 2023

[29] Digital Ocean:
https://www.digitalocean.com/community/tutorials/java-simpledateformat-java-date-format
Último acceso: Diciembre 2022

[30] Retrofit: https://square.github.io/retrofit/
Último acceso: Abril 2023

[31] Retrofit2: https://howtodoinjava.com/retrofit2/query-path-parameters/
Último acceso: Abril 2023

[32] Tutorial AndroidHive Picasso:
https://www.androidhive.info/2015/05/android-working-with-picasso-image-loader-library/
Último acceso: Abril 2023

[33] JavaTpoint: https://www.javatpoint.com/android-popup-menu-example
Último acceso: Mayo 2023

[34] Adding Settings to an App:
https://google-developer-training.github.io/android-developer-fundamentals-course-practic

en/Unit%204/92_p_adding_settings_to_an_app.html
Último acceso: Junio 2023

[35] Free Logo Maker: https://looka.com/explore
Último acceso: Marzo 2023

[36] Codepath: https://guides.codepath.com/android/settings-with-preferencefragment
Último acceso: Julio 2023

[37] Smartsheet: https://es.smartsheet.com/
Último acceso: Mayo 2023

[38] Overleaf: https://es.overleaf.com/
Último acceso: Noviembre 2023