Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención ingeniería del Software

NAVA APP : APLICACIÓN DE PEDIDOS PARA FOMENTAR EL COMERCIO LOCAL

Alumno
Daniel Melendre Vian

Tutor
Joaquin Adiego

Índice

Índice de tablas

Índice de figuras

1. Introducción

En los últimos años muchas actividades cotidianas se han digitalizado, es decir, han pasado a hacerse desde dispositivos electrónicos. Comprar es una de estas actividades que se ha digitalizado, ya que han surgido muchas opciones para ello (se hablará después en el apartado de Estado del arte).

Aunque esto ha sido un gran avance visto desde el lado del cliente, también ha provocado que muchos pequeños comercios hayan quedado obsoletos frente a estas nuevas opciones. Un ejemplo de esto ha ocurrido durante estos 2 últimos años, con la pandemia del Covid. Durante meses muchos negocios se tuvieron que adaptar a trabajar de forma remota con los clientes, en muchos casos recibiendo los pedidos de forma desorganizada, ya que no había alternativas para gestionar estos pedidos.

Por esto en este proyecto se propone el crear una aplicación gratuita que permita a cualquier empresario dar de alta su tienda, desde la que gestionar pedidos realizados por cualquier usuario, permitiendo un seguimiento cercano del desarrollo del pedido a ambas partes. Esta permitiría tanto realizar pedidos para recoger en la tienda como para llevar a casa del cliente.

1.1. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación para dispositivos móviles de Android capaz de realizar y gestionar pedidos por parte de usuarios a cualquier tienda registrada.

El otro gran objetivo, como ya se ha introducido, es fomentar el comercio local intentando hacer comprar a los usuarios en tiendas locales antes que otras remotas, y acercando a los pequeños comercios locales a las nuevas tecnologías intentando mantenerlos competitivos. Por ello un objetivo derivado de esto es que pueda ser usada en cualquier local, independientemente de la localidad, a diferencia de muchas otras aplicaciones de este estilo.

1.2. Alcance

La aplicación va destinada a todo tipo de usuario, ya que cualquier persona puede realizar un pedido por la aplicación, pero especialmente está dirigida a pequeños empresarios para intentar ayudarles a adaptar sus tiendas a las nuevas tecnologías.

En otro nivel, esta aplicación está pensado para usuarios que dispongan de un dispositivo electrónico con sistema operativo Android y que cuenten con una conexión a Internet estable para poder consultar la aplicación y acceder a todas sus funcionalidades en cualquier momento que lo deseen.

1.3. Funcionalidades de la aplicación

Para poder acceder la las principales funcionalidades de la aplicación, primero un usuario debe iniciar sesión usando su cuenta (correo y contraseña del usuario), o registrarse en el sistema si no dispone de una.

Una vez un usuario ha iniciado sesión, la aplicación se puede dividir en 2 partes, una sección para todos los clientes y una sección especializada en gestionar un negocio, diseñada para los empresarios.

1.3.1. Modo de cliente

Aquí se encuentran las funcionalidades disponibles para todos los usuarios registrados:

- Gestión del perfil: incluye las características y funcionalidades correspondientes a la gestión del perfil del usuario. Esto incluye la modificación de datos del usuario, tales como su nombre, apellido, nombre de usuario público, teléfono, dirección, correo, contraseña o una imagen de usuario.
- Gestión de ajustes de la aplicación: incluye las características y funcionalidades correspondientes a la gestión de los ajustes de la aplicación. Esto incluye activar el modo de negocio, ajustar la ubicación del usuario, activar notificaciones de la aplicación y permitir o denegar a tiendas ver información personal del usuario.
- Realizar un pedido: incluye las características y funcionalidades correspondientes a la selección de una tienda en la que pedir y la posterior creación del pedido, seleccionando los productos que se quieren comprar, junto con la cantidad de cada uno.También si fuera necesario añadiría la dirección a la que entregar el pedido y el "ticket" de promoción que se quiere aplicar.
- Seguir pedidos: incluye las características y funcionalidades correspondientes al seguimiento de sus pedidos, pudiendo filtrar los ya completados o cancelados. En cualquier momento el cliente puede ver el estado de un pedido, los productos pedidos y el precio del mismo; y, si lo quisiera, chatear con la tienda sobre este.
- Comprar promociones: incluye las características y funcionalidades correspondientes a la compra de "tickets" de descuento por medio de puntos internos de la app, los cuales servirán para recibir descuentos en futuros pedidos.

1.3.2. Modo de negocio

Aquí se encuentran las funcionalidades exclusivas para los empresarios:

- Creación de una tienda: incluye las características y funcionalidades correspondientes a dar de alta una tienda en el sistema. Aquí se introduce la información de la tienda, tal como su nombre, tipo(restaurante, panadería, etc.), teléfono, dirección, horarios, imágenes y si la tienda permite llevar los pedidos a casa del cliente.
- Gestión de una tienda: incluye las características y funcionalidades correspondientes a la modificación de la información de la tienda.
- Gestión del catálogo: incluye las características y funcionalidades correspondientes a la gestión del catálogo de cada tienda; añadir, modificar o quitar productos del catálogo, cada uno con su nombre, precio y una pequeña descripción.
- Gestión de los pedidos: incluye las características y funcionalidades correspondientes a la gestión de los pedidos realizados por clientes. Esto incluye la visualización de la información de cada pedido: precio total, productos, cantidad de cada uno, ultima actualización y dirección de entrega (en caso necesario). También se incluye aquí el avanzar el estado del pedido por ejemplo de *Haciendo* a *Listo para recoger*.

2. Estado del arte

Para poder comenzar el proyecto planteado lo primero que debemos hacer es analizar las aplicaciones similares presentes ya en el mercado actual. En el mercado hay muchas apps que cumplen una función similar a la que queremos cumplir, por lo que se van a exponer las más populares para compararlas.

Primero se van a mostrar apps de reparto de pedidos "genéricos", como es el caso de Glovo ya que los objetivos que se pretenden conseguir con nuestra aplicación son similares a los que ofrece este servicio.

También se va a hablar de aplicaciones de reparto de comida ya que en algunas de estas apps especializadas encontramos funciones interesantes de las que coger inspiración.En concreto se va a hablar de UberEats ya que es una de las más completas en este aspecto, aunque existen muchas como son JustEat, Deliveroo, etc. Dentro de este apartado también se va a hablar de aplicaciones de algunas de las cadenas de comida rápida más populares.

2.1. Glovo

Glovo es una aplicación de reparto de todo tipo de pedidos, la cual opera en países de Europa (España, Italia, Eslovenia, Croacia, etc.) y algunos países africanos. Hay otras aplicaciones similares como PedidosYa, la cual funciona en latinoamérica.

Glovo está basada en la contratación de repartidores llamados "glovers", los cuales funcionan como trabajadores por cuenta propia, encargados de recoger el pedido en la tienda y llevarlo hasta el cliente. Esto es una ventaja en grandes ciudades donde hay de estos repartidores, ya que permiten a las tiendas llevar sus pedidos a casa del cliente de forma rápida y sencilla, pero también es un inconveniente ya que en poblaciones más pequeñas donde no hay repartidores Glovo no puede funcionar.

2.2. Uber Eats

Uber Eats es una aplicación de entrega de comida. Esta funciona por todo el mundo, y su funcionamiento es similar al de glovo, disponen de repartidores encargados de recoger los pedidos en las tiendas y llevárselos a los clientes.

Una de las grandes ventajas de esta app respecto a otras es que permite realizar pedidos tanto para recoger en la tienda como para llevar a casa, lo cual es una función muy interesante para nuestros objetivos, aunque teniendo el problema de no estar disponible en localidades pequeñas (donde no disponen de repartidores).

2.3. Cadenas de comida rápida

Muchas grandes cadenas de comida rápida cuentan con su propia app para hacer pedidos a domicilio. McDonalds, Telepizza, Burger King, Domino's o KFC son ejemplos de ellas.

Estas apps están menos pulidas y tienen menos opciones comparadas con UberEats, Glovo y similares, aunque aquí podrás encontrar ofertas especiales y descuentos exclusivos relacionados con sus productos. Algunas de ellas también disponen de la opción de recoger el pedido en la tienda, con su propio servicio o usando alguno de los ya mencionados (McDonalds usa Glovo por ejemplo).

2.4. Comparación de propuestas de mercado

Tras analizar todas estas opciones se puede observar que hay una gran variedad de aplicaciones de pedidos, pero la mayoría de estas tienen un enfoque y objetivos distintos a los que se quiere lograr en este proyecto.

A pesar de que con un enfoque distinto, hay muchas cosas que se han tomado como ejemplo en nuestro proyecto, como puede el sistema de poder comprar de todo (de Glovo), el poder escoger recoger pedidos en la tienda si no hay repartidores (de UberEats) o el sistema de ofertas (de cadenas de comida rápida) del cual se hablará en detalle más adelante.

En la tabla Comparación con otras propuestas se puede ver una comparativa entre todas las aplicaciones mostradas:

	Glovo	UberEats	Cadenas	Nava
Todo tipo de pedidos (no solo comida)	Si	No	No	Si
Pedidos para llevar	Si	Si	Si	Si
Pedidos para recoger en tienda	No	Si	Algunas	Si
Pagos dentro de la app	Si	Si	Si	No*
Ofertas especiales	Si(código)	Si(código)	Si(promos)	Si(promos)

Tabla 1: Comparación con otras propuestas

** - Se podría implementar en futuras versiones*

Cabe destacar que actualmente a pesar de que nuestra aplicación permite crear pedidos para llevar, a diferencia de otras no tiene un sistema de repartidores encargados de llevar estos pedidos, si no que la responsabilidad de esto recae sobre la tienda, la cual es la que escogerá si los clientes pueden pedir para llevar en su tienda. Es decir, la aplicación se plantea tal que su funcionamiento principal sea el recoger los pedidos en tienda, aunque si desde esta se puede permitir enviar los pedidos, como una funcionalidad extra.

3. Planificación del proyecto

En esta sección se van a describir la metodología de trabajo empleada para el desarrollo del proyecto, la organización y la distribución de las fases de trabajo, junto con los costes y estimaciones iniciales del presupuesto requerido para la finalización de este.

3.1. Metodología de trabajo

Para este proyecto se ha decidido seguir el modelo de desarrollo incremental, en este un proyecto es descompuesto en una serie de pequeños proyectos o incrementos, cada uno de los cuales suministra una parte de la funcionalidad respecto de la totalidad de los requisitos del proyecto.

Estos incrementos constan con sus actividades de análisis, diseño, implementación y pruebas. El resultado de cada uno de estas iteraciones es una versión del proyecto con una nueva funcionalidad añadida y también una evaluación de la misma.

Se ha decidido escoger esta metodología sobre otras ya que ofrece las siguientes ventajas:

- Progreso visible desde las primeras etapas
- Permite una mitigación temprana de posibles riesgos altos que afecten al desarrollo.
- Es un modelo flexible, por lo que se reduce el coste en un cambio de alcance o de requisitos.
- Permite probar y depurar más fácilmente las funcionalidades en iteraciones más pequeñas.

Aunque tenga estas ventajas también hay inconvenientes, para esta metodología podemos destacar que cada fase de una iteración es rígida, lo cual significa que para terminar cada una de los incrementos es necesario haber completado todas las fases para garantizar el correcto funcionamiento del sistema antes de añadir nuevas funcionalidades. También pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido al inicio del desarrollo.

3.1.1. Incrementos

Para este proyecto, la carga de trabajo se ha dividido en los siguientes incrementos:

- Iteración inicial para el análisis exhaustivo de requisitos.
- Diseño de la interfaz general de la aplicación.
- Desarrollo del inicio de sesión y registro del usuario.
- Conexion de la aplicación con la base de datos.
- Desarrollo de la creación y gestión de tiendas.
- Desarrollo de la creación de un pedido por parte del cliente.
- Desarrollo de la consulta y seguimiento de pedidos.
- Sistema de notificaciones en el seguimiento del pedido.
- Desarrollo del chat asociado a un pedido.
- Desarrollo del perfil de usuario.
- Desarrollo de los ajustes de sistema.

- Desarrollo del sistema de promociones.
- Iteración final para revisión final y documentación

Cabe destacar que dentro de cada iteración se ha realizado tanto trabajo de desarrollo de interfaces específicas de la funcionalidad como de obtención y carga de datos de la base de datos dentro del diseño e implementación de la misma.

3.2. Plan de trabajo

Un plan de trabajo se utiliza para la planificación y organización de un proyecto, de forma que se pueda tener una visión del trabajo a realizar bien definida.

Antes de hablar de la planificación del proyecto, es necesario puntualizar que se trabajará en el proyecto de lunes a viernes, por lo que al contar los días no se sumaran sábados ni domingos. El tiempo de trabajo será de entre 20 y 25 horas semanales, correspondientes a la franja horaria de las tardes.

A continuación se va a detallar la planificación inicial de las tareas que se van a realizar a lo largo del proyecto en cada una de las iteraciones del mismo, junto con la duración estimada para ellas:

Descripción	Fecha de Inicio	Fecha de Fin	Días
Análisis de requisitos	14/2/2022	4/3/2022	15
Diseño de interfaz general	7/3/2022	10/3/2022	4
Inicio de sesión y registro	11/3/2022	15/3/2022	3
Conexión con B.D.	16/3/2022	18/3/2022	3
Creación de tiendas	21/3/2022	30/3/2022	13
Creación del pedido	31/3/2022	8/4/2022	7
Seguimiento del pedido	11/4/2022	15/4/2022	5
Sistema de notificaciones	18/4/2022	20/4/2022	3
Chat	21/4/2022	22/4/2022	2
Perfil de usuario	25/4/2022	27/4/2022	3
Ajustes de sistema	28/4/2022	29/4/2022	2
Sistema de promociones	2/5/2022	6/5/2022	5
Revisión final y documentación	9/5/2022	27/5/2022	15

Tabla 2: Etapas de la planificación

Estas etapas son estimaciones y pueden tener modificaciones en los días de desarrollo dependiendo de la velocidad de desarrollo y posibles problemas que pudieran surgir.

3.3. Gestión de riesgos

La finalidad de la gestión de riesgos es identificar, analizar y tratar los riesgos que puedan surgir durante el desarrollo de un proyecto. Las fases de la gestión de riesgos son:

- Identificar: Localizar los posibles riesgos antes de que se presenten.
- Analizar: Evaluar el impacto y la probabilidad.
- Planificar: Elaborar un plan de actuación para en caso de que ocurra un riesgo saber como actuar para así disminuir el efecto del riesgo.
- Monitorizar: Seguir los indicadores de riesgo durante el proyecto.

Podemos clasificar los riesgos según su probabilidad de que ocurran y las consecuencias que pueden tener para el desarrollo para valorar el impacto que pueden tener sobre el proyecto (como se muestra en [**1**]).

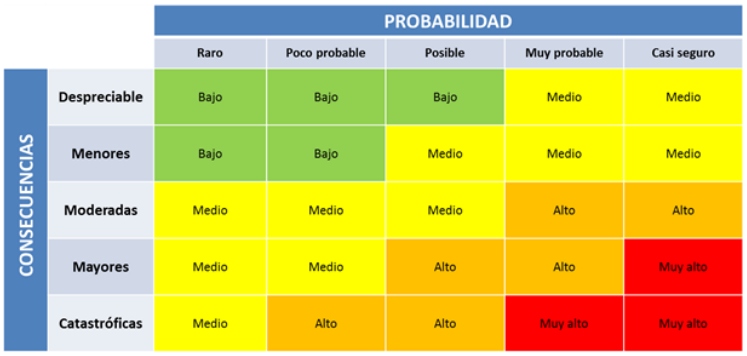


Imagen 1: Tabla de riesgos.

Basándonos en esta clasificación se pueden analizar filtrar los riesgos con un impacto alto o superior, y para cada uno de estos crear un plan de contingencia:

Descripción	Retrasos en la planificación original
Consecuencias	Mayores
Probabilidad	Muy probable
Impacto	Alto
Plan de contingencia	Reorganizar la planificación en base a la nueva disponibilidad. Usar días no laborables para recuperar los posibles días perdidos.

Tabla 3: Riesgo R01. Retrasos en la planificación

Descripción	Pérdida de datos del proyecto
Consecuencias	Catastróficas
Probabilidad	Posible
Impacto	Alto
Plan de contingencia	Realizar copias de seguridad en varios sitios (local, github, etc.) usualmente.

Tabla 4: Riesgo R02. Pérdida de datos

Descripción	Rotura de la máquina usada
Consecuencias	Catastróficas
Probabilidad	Poco probable
Impacto	Alto
Plan de contingencia	Disponer de una segunda máquina en la que trabajar durante la reparación o sustitución. También aplicar lo visto en Riesgo R02. Pérdida de datos

Tabla 5: Riesgo R03. Rotura de la máquina usada

Descripción	Los requisitos necesitan ser modificados o surge alguno no tenido en cuenta en el inicio.
Consecuencias	Catastróficas
Probabilidad	Poco probable
Impacto	Alto
Plan de contingencia	Modificar el orden de las tareas para poder añadir los nuevos requisitos al desarrollo del proyecto. Aplicar lo visto en Riesgo R01. Retrasos en la planificación para reestructurar el tiempo

Tabla 6: Riesgo R04. Requisitos incorrectos

Descripción	Fallo en la creación del diseño, este no se ajusta a los requisitos pedidos.
Consecuencias	Mayores
Probabilidad	Posible
Impacto	Alto
Plan de contingencia	Corregir los errores revisando los requisitos de nuevo. Aplicar lo visto en Riesgo R01. Retrasos en la planificación para recuperar el tiempo

Tabla 7: Riesgo R05. Diseño incorrecto

3.4. Costes del proyecto

3.4.1. Costes de personal

Estos costes hacen referencia a los recursos humanos empleados durante el proceso de desarrollo del proyecto.

En este caso no hay costes de personal ya que todos los roles (jefe de proyecto, analista, diseñador de interfaces, desarollador, etc.) han sido cubiertos por el autor.

3.4.2. Costes hardware

Estos costes hacen referencia a los dispositivos empleados en la realización del proyecto.

Descripción	Precio (€)
Ordenador portátil	850
Monitor	225
Teléfono móvil	200
Ratón inalámbrico	40

Tabla 8: Costes hardware

3.4.3. Costes software

Estos costes hacen referencia a los programas utilizados en la realización del proyecto.

Descripción	Precio (€)
Windows 10	Gratuito
Word (Office 365)	Gratuito
Android Studio	Gratuito
GitHub	Gratuito
Firebase	Gratuito
Google Maps	Gratuito
Overleaf (v. gratis)	Gratuito

Tabla 9: Costes software

3.4.4. Costes adicionales

Aparte de los costes ya hablado podemos añadir dos costes más.

- El primero es la conexión a internet y la tarifa de soporte del teléfono, de 80€ mensuales, aunque esta incluye más productos como una segunda linea móvil o una línea de teléfono fijo.
- El segundo es el gasto eléctrico consumido durante el desarrollo, aproximadamente el valor de este gasto ronda los 20€.

3.4.5. Posibles costes futuros

En esta sección se va a hablar de costes que se han valorado durante el desarrollo de la aplicación, y podrían añadirse en un futuro a los ya mencionados en los apartados anteriores.

- Cuenta de desarrollador de Google Play - Necesaria para publicar aplicaciones en Google Play, poder publicar la aplicación en Google Play facilitaría todo el proceso de descarga para los usuarios, ya que esta es la forma “oficial” de descargar apps en android (ver Instalación de la aplicación).

  Tiene una cuota de registro de 25€ que deberíamos añadir a los costes software.

- Places API - Esta API brinda acceso a la base de datos de Google de información local de lugares y negocios. En el caso de este proyecto se plantea usarla para obtener datos de los negocios tales como reseñas de usuarios, horarios o imágenes.

  Esta tiene un coste mensual de 17$ (16,37€) que habría que añadir a los costes de software.

- Repartidores - Como ya se ha mencionado en apartados anteriores (ver Estado del arte), una opción de mejorar la aplicación es contar con repartidores encargados de llevar los pedidos a casa de los clientes, como hacen otras propuestas.

  Obviamente el coste saldría de pagar a estos por su servicio, el cual habría que añadir a los costes de personal.

3.5. Rentabilidad del proyecto

Aunque en la realidad no se va a comercializar la aplicación, en este apartado se van a plantear sistemas que podrían funcionan para obtener un ingreso de la misma, así como cómo podríamos saber su rentabilidad.

- Comisiones en cada pedido - Un sistema utilizado por bastantes aplicaciones del sector, como es el ejemplo de Glovo o UberEats, es el cobrar un porcentaje extra por cada pedido realizado.

  Por poner un ejemplo, con un 1 % si un pedido costase 10€ al cliente, el beneficio sería de 10 céntimos, y para la tienda de 9.90€. Para ello primero habría que implementar un sistema de pagos desde la aplicación y habría que ajustar los precios para que este porcentaje estuviera integrado.

  Este sistema sería el preferido para aplicar, ya que depende directamente del uso que se le dé a la aplicación (pedidos realizados).

- Uso de pago - Este sistema consistiría en que para usar el modo de empresario dentro de la aplicación habría que pagar una tasa, la cual podría ser un único pago o varios pagos mensuales o anuales.

  Este sistema sería menos dependiente de los clientes y los pedidos realizados, pero más de que haya tiendas registradas en el sistema. Podría funcionar si se quiere beneficio a corto plazo, pero en general parece una opción peor que la anterior.

- Anuncios - Algo muy popular entre aplicaciones móviles (especialmente juegos) es el incluir anuncios dentro de la app, ya sea en una parte de la pantalla o apareciendo de vez en cuando.

  Aunque no sea lo más popular entre aplicaciones de este tipo este sistema no dependería de los usuarios de la aplicación, ya que el ingreso vendría de un tercero, como podría ser Google pero haría que la experiencia general fuera bastante peor para el usuario.

En cualquiera de los casos, habría que calcular el ROI (Retorno de Inversión) del proyecto. Este es una forma de analizar un resultado económico que nos diría si un proyecto ha generado pérdidas o si ha resultado rentable, y nos mostrará el rendimiento de la inversión que hubieramos realizado.

Para calcular el ROI habría que aplicar: *ROI = (Ingresos - Inversión) / Inversión*

4. Análisis del sistema

En este apartado se van a mostrar los distintos aspectos que se han considerado previamente a la realización del diseño e implementación del sistema. Esto incluye actores, requisitos, hasta los requisitos que debe satisfacer el sistema, junto con las especificaciones y diagramas para su representación.

4.1. Actores del sistema

Los actores representan a cualquier agente interno o externo que participan en el sistema. En este proyecto se han localizado los siguientes:

- Usuario : cualquier usuario registrado que accede a la aplicación. Puede acceder a todas las características del Modo de cliente
- Empresario : usuario registrado con acceso a las características del Modo de negocio
- Firebase : servicio en la nube donde se almacena la información mostrada al usuario en la aplicación. Se comunica con la aplicación a través de peticiones realizadas por el usuario. (Actor Externo)
- Maps : es un servidor de mapas externo. La aplicación se comunica con este cuando un usuario solicita consultar direcciones. (Actor Externo)

4.2. Requisitos

La especificación de requisitos es una descripción completa del sistema que se va a desarrollar.

4.2.1. Requisitos funcionales

Los requisitos funcionales representan las funcionalidades que un sistema debe satisfacer. En este proyecto se va a organizar estos requisitos según el actor que va a ser capaz de desempeñarlos.

Los dos actores que van a realizar la mayoría de estas funcionalidades, el actor “usuario” y el actor “empresario” ya que los otros actores son externos al sistema. Cabe destacar que un empresario es un usuario con permisos extra, por lo que este va a poder realizar todas las funcionalidades asociadas a usuario.

Los requisitos identificados para el actor usuario son los siguientes:

ID	Descripción
RF-01	El sistema permitirá iniciar sesión para acceder a la pantalla principal
RF-02	El sistema permitirá registrarse para acceder a la pantalla principal
RF-03	El sistema mostrará la lista de tiendas registradas en la BD
RF-04	El sistema permitirá filtrar la lista de tiendas visualizadas
RF-05	El sistema mostrará la localización de una tienda
RF-06	El sistema permitirá seleccionar una tienda para realizar un pedido en ella
RF-07	El sistema permitirá preparar un pedido
RF-08	El sistema mostrará la lista de pedidos realizados previamente por un usuario
RF-09	El sistema permitirá filtrar la lista de pedidos realizados previamente por un usuario
RF-10	El sistema permitirá al usuario cancelar un pedido realizado
RF-11	El sistema permitirá intercambiar mensajes con la tienda de un pedido realizado
RF-12	El sistema mostrará las promociones disponibles
RF-13	El sistema permitirá adquirir un ticket de una promoción seleccionada a un usuario
RF-14	El sistema mostrará la lista de tickets de un usuario adquiridos previamente
RF-15	El sistema permitirá ver al usuario su perfil de usuario
RF-16	El sistema permitirá al usuario modificar datos de su perfil
RF-17	El sistema permitirá actualizar la ubicación al usuario
RF-18	El sistema permitirá configurar si recibir notificaciones

Tabla 10: Requisitos del actor usuario

Los requisitos identificados para el actor empresario son los siguientes:

ID	Descripción
RF-19	El sistema mostrará sus tiendas al empresario
RF-20	El sistema permitirá crear una nueva tienda
RF-21	El sistema permitirá modificar datos de una tienda
RF-22	El sistema permitirá modificar el catalogo de una tienda
RF-23	El sistema permitirá eliminar una tienda
RF-24	El sistema mostrará al empresario la lista de pedidos recibidos en sus tiendas
RF-25	El sistema permitirá filtrar la lista de pedidos recibidos
RF-26	El sistema permitirá cancelar un pedido recibido
RF-27	El sistema permitirá intercambiar mensajes con el cliente de un pedido recibido
RF-28	El sistema permitirá avanzar el estado de un pedido recibido

Tabla 11: Requisitos del actor empresario

4.2.2. Requisitos no funcionales

Describen limitaciones que afectan al sistema.Para desarrollar estos requisitos se ha utilizado el sistema [F]URPS:

- F-Funcionalidad: Representan los requisitos funcionales del sistema (ya definidos).
- U-Usabilidad: Requisitos asociados a la interfaz de usuario.
- R-Fiabilidad: Frecuencia y gravedad de los fallos, además de la capacidad para recuperarse de ellos.
- P-Rendimiento: Tiempo de respuesta del sistema
- S-Soporte: Facilidad de mantenimiento y configuración del sistema.

ID	Descripción
RNF-01	El sistema será ejecutado en cualquier dispositivo con sistema operativo Android con versión 9 o superior.
RNF-02	El sistema funcionará en posición vertical.
RNF-03	La aplicación será adaptable a los diferentes tamaños de dispositivos presentes en el mercado para su correcta visualización.
RNF-04	El sistema mostrará al usuario una alerta en caso de no cargar alguna información o por necesidad de alguna confirmación
RNF-05	El sistema deberá estar disponible 24 horas al día durante todo el año.
RNF-06	El sistema deberá soportar grandes cantidades de tráfico de usuarios.
RNF-06	Los datos de los usuarios introducidos en el sistema estarán almacenados de forma segura.
RNF-07	El sistema se comunicará con el servicio “Cloud Firestore” (de Firebase) cada vez que el usuario solicite una información, para recuperarla y posteriormente mostrársela al usuario.
RNF-08	El sistema deberá tener un tiempo de respuesta menor de 5 segundos.

Tabla 12: Requisitos no funcionales

4.3. Casos de uso

En esta sección se va a mostrar el modelo de casos de uso realizado en base a los actores y requisitos mencionados en los apartados anteriores así como la especificación de cada uno.

4.3.1. Diagrama de casos de uso

El diagrama de casos de uso se vería de la siguiente manera:

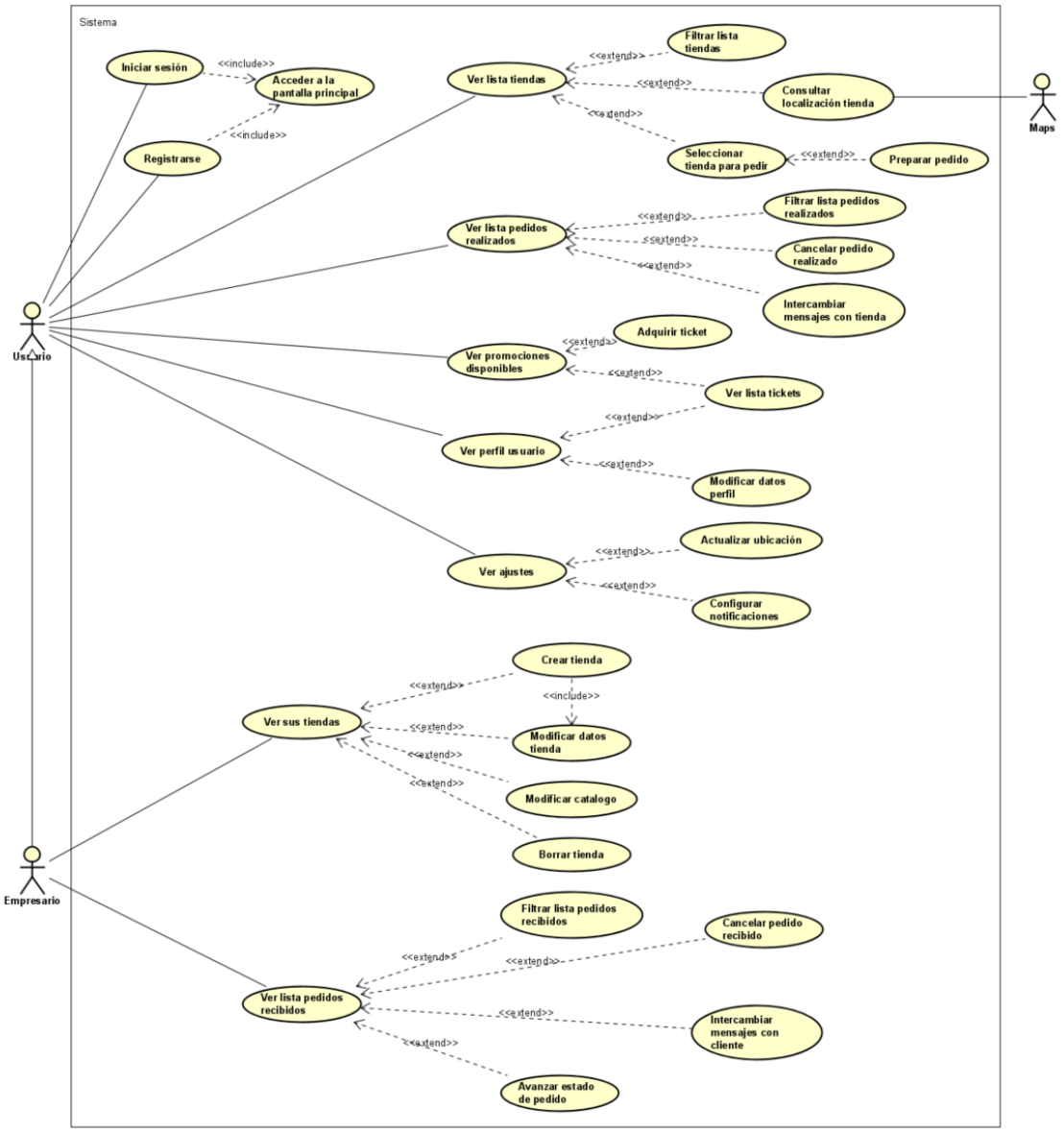


Imagen 2: Diagrama de casos de uso.

Para mejorar la comprensión del diagrama se han eliminado alguna líneas pero todos los casos de uso de "Ver algo" extienden del caso de uso de *Acceder a la pantalla principal*, es decir, necesitas ser un usuario registrado para acceder a estas opciones.

4.3.2. Especificaciones de casos de uso

Identificador	CU-01 Iniciar sesión
Descripción	Caso de uso correspondiente al RF-01
Actores	Usuario
Precondición	
Flujo principal	1. El usuario introduce correo y contraseña 2. El sistema lee los datos 3. El usuario pulsa el botón de acceder 4. El sistema comprueba que el usuario existe CU-03 Acceder a la pantalla principal
Flujo alternativo	4a. Si el usuario no existe, notifica al usuario.El caso de uso vuelve al paso 1
Postcondición	El usuario tiene una sesión activa

Tabla 13: CU-01 Iniciar sesión

Identificador	CU-02 Registrarse
Descripción	Caso de uso correspondiente al RF-02
Actores	Usuario
Precondición	
Flujo principal	1. El usuario pulsa el botón registrarse 2. El sistema abre el menú de registro 3. El usuario introduce sus datos 4. El sistema lee los datos 5. El usuario pulsa el botón registrarse 6. El sistema crea al usuario CU-03 Acceder a la pantalla principal
Flujo alternativo	6a. Si el nombre de usuario o correo ya existe, notifica al usuario. El caso de uso vuelve al paso 3 *. Si el usuario pulsa el botón de retroceder entre el paso 3 y 6 vuelve al menú inicial
Postcondición	El usuario tiene una sesión activa

Tabla 14: CU-02 Registrarse

Identificador	CU-03 Acceder a la pantalla principal
Descripción	Acciones del sistema cuando un usuario ha iniciado sesión o se ha registrado
Actores	Usuario
Precondición	El usuario ha iniciado sesión o se ha registrado en el sistema
Flujo principal	1. El sistema carga toda la información relativa al usuario 2. El sistema abre el menú principal
Flujo alternativo	
Postcondición	El usuario está en el menú principal con sus datos cargados

Tabla 15: CU-03 Acceder a la pantalla principal

Identificador	CU-04 Ver lista de tiendas
Descripción	Caso de uso correspondiente al RF-03
Actores	Usuario
Precondición	El usuario tiene una sesión activa y está en el menú principal
Flujo principal	1. El usuario selecciona en el menú el apartado “Comprar” 2. El sistema muestra una lista de tiendas
Flujo alternativo	
Postcondición	El usuario visualiza una lista de tiendas

Tabla 16: CU-04 Ver lista de tiendas

Identificador	CU-05 Filtrar lista de tiendas
Descripción	Caso de uso correspondiente al RF-04
Actores	Usuario
Precondición	El usuario visualiza una lista de tiendas. CU-04 Ver lista de tiendas
Flujo principal	1. El usuario pulsa el botón de Filtros 2. El sistema despliega un menú con los filtros disponible 3. El usuario modifica alguno de los filtros 4. El sistema muestra una lista modificada de tiendas en base a los filtros
Flujo alternativo	*.Si el usuario pulsa el botón de Filtros de nuevo entre el paso 2 y 4 el sistema oculta los filtros (vuelve al paso 1)
Postcondición	El usuario visualiza una lista de tiendas

Tabla 17: CU-05 Filtrar lista de tiendas

Identificador	CU-06 Consultar localización de una tienda
Descripción	Caso de uso correspondiente al RF-05
Actores	Usuario, Maps
Precondición	El usuario visualiza una lista de tiendas. CU-04 Ver lista de tiendas
Flujo principal	1. El usuario pulsa el botón con la dirección de la tienda que quiera consultar 2. El sistema abre la aplicación de Maps con la información de la tienda
Flujo alternativo	
Postcondición	El usuario está en Maps con la localización de la tienda

Tabla 18: CU-06 Consultar localización de una tienda

Identificador	CU-07 Seleccionar una tienda para pedir
Descripción	Caso de uso correspondiente al RF-06
Actores	Usuario
Precondición	El usuario visualiza una lista de tiendas. CU-04 Ver lista de tiendas
Flujo principal	1. El usuario pulsa el botón realizar pedido de la tienda que quiera 2. El sistema abre el menú de compra con el catálogo de la tienda seleccionada
Flujo alternativo	
Postcondición	El usuario está en el menú de compra

Tabla 19: CU-07 Seleccionar una tienda para pedir

Identificador	CU-08 Preparar pedido
Descripción	Caso de uso correspondiente al RF-07
Actores	Usuario
Precondición	El usuario está en el menú de la tienda seleccionada en CU-07 Seleccionar una tienda para pedir
Flujo principal	1. El usuario añade los productos que quiera del catálogo al pedido 2. El sistema actualiza el pedido en pantalla 3. El usuario pulsa el botón de Completar pedido 4. El sistema guarda el pedido en la base de datos 5. El sistema avisa a la tienda de que tiene un pedido 6. El sistema vuelve al menú principal
Flujo alternativo	4a. Si el pedido está vacío avisa al usuario. El caso de uso vuelve al paso 1. *. Si el usuario pulsa el botón de retroceder, se abrirá una alerta que preguntará si quiere volver, si la acepta vuelve al menú principal sin guardar nada.
Postcondición	El usuario está en el menú principal y el pedido ha sido realizado

Tabla 20: CU-08 Preparar pedido

Identificador	CU-09 Ver lista de pedidos realizados
Descripción	Caso de uso correspondiente al RF-08
Actores	Usuario
Precondición	El usuario tiene una sesión activa y está en el menú principal
Flujo principal	1. El usuario selecciona en el menú el apartado “Mis pedidos” 2. El sistema consulta y muestra la lista de pedidos realizados por el usuario
Flujo alternativo	
Postcondición	El usuario visualiza su lista de pedidos realizados

Tabla 21: CU-09 Ver lista de pedidos realizados

Identificador	CU-10 Filtrar lista de pedidos realizados
Descripción	Caso de uso correspondiente al RF-09
Actores	Usuario
Precondición	El usuario visualiza su lista de pedidos realizados. CU-08 Preparar pedido
Flujo principal	1. El usuario pulsa el botón de Mostrar pedidos completados/cancelados 2. El sistema muestra una lista modificada de pedidos en base a los filtros
Flujo alternativo	
Postcondición	El usuario visualiza su lista de pedidos realizados

Tabla 22: CU-10 Filtrar lista de pedidos realizados

Identificador	CU-11 Cancelar pedido realizado
Descripción	Caso de uso correspondiente al RF-10
Actores	Usuario
Precondición	El usuario visualiza su lista de pedidos realizados. CU-08 Preparar pedido
Flujo principal	1. El usuario selecciona un pedido de la lista 2. El sistema muestra información del pedido seleccionado 3. El usuario pulsa el botón de cancelar pedido 4. El sistema cancela el pedido 5. El sistema notifica a la tienda de la cancelación
Flujo alternativo	2a. El pedido no es cancelable (por ejemplo está completado) y el caso de uso termina.
Postcondición	El pedido seleccionado es cancelado

Tabla 23: CU-11 Cancelar pedido realizado

Identificador	CU-12 Intercambiar mensajes con tienda
Descripción	Caso de uso correspondiente al RF-11
Actores	Usuario
Precondición	El usuario visualiza su lista de pedidos realizados. CU-08 Preparar pedido
Flujo principal	1. El usuario selecciona un pedido de la lista 2. El sistema muestra información del pedido seleccionado 3. El usuario pulsa el botón de mostrar el chat 4. El sistema abre el menú del chat 5. El usuario escribe y envia un mensaje 6. El sistema guarda el mensaje
Flujo alternativo	*. Si el usuario pulsa el botón de retroceder entre los pasos 4 y 6 el sistema regresa al menú. El caso de uso vuelve al punto 1.
Postcondición	El usuario ha enviado un mensaje a la tienda

Tabla 24: CU-12 Intercambiar mensajes con tienda

Identificador	CU-13 Ver lista de promociones
Descripción	Caso de uso correspondiente al RF-12
Actores	Usuario
Precondición	El usuario tiene una sesión activa y está en el menú principal
Flujo principal	1. El usuario selecciona en el menú el apartado “Promociones” 2. El sistema muestra una lista de promociones
Flujo alternativo	
Postcondición	El usuario visualiza una lista de promociones

Tabla 25: CU-13 Ver lista de promociones

Identificador	CU-14 Adquirir ticket
Descripción	Caso de uso correspondiente al RF-13
Actores	Usuario
Precondición	El usuario visualiza una lista de promociones. CU-13 Ver lista de promociones
Flujo principal	1. El usuario selecciona el botón comprar promoción 2. El sistema genera un ticket con la promoción 3. El sistema guarda el ticket en la base de datos del usuario
Flujo alternativo	
Postcondición	El usuario tiene un ticket de la promoción

Tabla 26: CU-14 Adquirir ticket

Identificador	CU-15 Ver lista tickets
Descripción	Caso de uso correspondiente al RF-14
Actores	Usuario
Precondición	El usuario visualiza una lista de promociones (CU-13 Ver lista de promociones) o su perfil (CU-16 Ver perfil usuario)
Flujo principal	1. El usuario selecciona el botón ver mis promociones 2. El sistema muestra una lista con los tickets del usuario
Flujo alternativo	
Postcondición	El usuario cisualiza una lista con sus tickets

Tabla 27: CU-15 Ver lista tickets

Identificador	CU-16 Ver perfil usuario
Descripción	Caso de uso correspondiente al RF-15
Actores	Usuario
Precondición	El usuario tiene una sesión activa y está en el menú principal
Flujo principal	1. El usuario selecciona en el menú el apartado “Mi perfil” 2. El sistema muestra el perfil del usuario
Flujo alternativo	
Postcondición	El usuario visualiza su perfil

Tabla 28: CU-16 Ver perfil usuario

Identificador	CU-17 Modificar datos del perfil de usuario
Descripción	Caso de uso correspondiente al RF-16
Actores	Usuario
Precondición	El usuario está en el menú de su perfil. CU-16 Ver perfil usuario
Flujo principal	1. El usuario selecciona el campo a modificar 2. El sistema muestra un cuadro donde introducir el dato 3. El usuario introduce el dato que quiera modificar 4. El sistema comprueba que el dato sea valido 5. El sistema actualiza la información del usuario
Flujo alternativo	4a. Si el dato introducido no es correcto el sistema notifica al usuario. El caso de uso vuelve al paso 1
Postcondición	El usuario visualiza su perfil actualizado

Tabla 29: CU-17 Modificar datos del perfil de usuario

Identificador	CU-18 Ver ajustes
Descripción	El usuario accede al menú donde configurar sus ajustes.
Actores	Usuario
Precondición	El usuario tiene una sesión activa y está en el menú principal
Flujo principal	1. El usuario selecciona en el menú el apartado “Ajustes” 2. El sistema consulta y muestra los ajustes del usuario
Flujo alternativo	
Postcondición	El usuario visualiza sus ajustes

Tabla 30: CU-18 Ver ajustes

Identificador	CU-19 Actualizar ubicación
Descripción	Caso de uso correspondiente al RF-17
Actores	Usuario
Precondición	El usuario está en el menú de ajustes. CU-18 Ver ajustes
Flujo principal	1. El usuario pulsa el botón de actualizar la ubicación 2. El sistema localiza al usuario 3. El sistema actualiza la ubicación del usuario
Flujo alternativo	2a. Si no se consigue localizar, el sistema notifica al usuario
Postcondición	El usuario tiene la ubicación actualizada

Tabla 31: CU-19 Actualizar ubicación

Identificador	CU-20 Configurar notificaciones
Descripción	Caso de uso correspondiente al RF-18
Actores	Usuario
Precondición	El usuario está en el menú de ajustes. CU-18 Ver ajustes
Flujo principal	1. El usuario activa/desactiva las notificaciones 2. El sistema actualiza la configuración del usuario
Flujo alternativo	
Postcondición	El usuario tiene la configuración de notificaciones actualizada

Tabla 32: CU-20 Configurar notificaciones

Identificador	CU-21 Ver sus tiendas
Descripción	Caso de uso correspondiente al RF-19
Actores	Empresario
Precondición	El empresario tiene una sesión activa y está en el menú principal
Flujo principal	1. El empresario selecciona en el menú el apartado “Mis tiendas” 2. El sistema muestra la lista de tiendas asociadas al empresario
Flujo alternativo	
Postcondición	El usuario visualiza sus tiendas registradas

Tabla 33: CU-21 Ver sus tiendas

Identificador	CU-22 Crear una tienda
Descripción	Caso de uso correspondiente al RF-20
Actores	Empresario
Precondición	El empresario visualiza sus tiendas registradas
Flujo principal	1. El empresario pulsa el botón crear tienda 2. El sistema crea una tienda CU-23 Modificar datos de una tienda
Flujo alternativo	
Postcondición	El usuario ha registrado una nueva tienda

Tabla 34: CU-22 Crear una tienda

Identificador	CU-23 Modificar datos de una tienda
Descripción	Caso de uso correspondiente al RF-21
Actores	Empresario
Precondición	El empresario visualiza sus tiendas registradas
Flujo principal	1. El empresario seleciona modificar una tienda. 2. El sistema muestra una pantalla con información detallada de la tienda 3. El empresario selecciona el dato a modificar 4. El sistema muestra una interfaz para modificar 5. El empresario introduce el nuevo valor 6. El sistema verifica y guarda el cambio 7. El empresario pulsa el botón de guardar 8. El sistema actualiza la BD con los nuevos datos
Flujo alternativo	6a. Si el dato introducido no cumple alguna restricción, el usuario es notificado del error *. Si el empresario pulsa el botón de volver, entre los pasos 3 y 6 el caso de uso termina.
Postcondición	El usuario ha modificado datos de una tienda

Tabla 35: CU-23 Modificar datos de una tienda

Identificador	CU-24 Modificar catálogo de una tienda
Descripción	Caso de uso correspondiente al RF-22
Actores	Empresario
Precondición	El empresario visualiza sus tiendas registradas
Flujo principal	1. El empresario seleciona modificar el catálogo de una tienda. 2. El sistema muestra una pantalla con el catálogo actual de la tienda 3. El empresario modifica el catálogo 4. El sistema verifica las modificaciones 5. El empresario pulsa el botón de guardar 6. El sistema actualiza la BD con el nuevo catálogo
Flujo alternativo	4a. Si las modificaciones no cumplen alguna restricción, el usuario es notificado del error *. Si el empresario pulsa el botón de volver, entre los pasos 1 y 4 el caso de uso termina.
Postcondición	El usuario ha modificado el catálogo de una tienda

Tabla 36: CU-24 Modificar catálogo de una tienda

Identificador	CU-25 Eliminar una tienda
Descripción	Caso de uso correspondiente al RF-23
Actores	Empresario
Precondición	El empresario visualiza sus tiendas registradas
Flujo principal	1. El empresario seleciona borrar una tienda. 2. El sistema muestra una pantalla de confirmación 3. El empresario confirma 4. El sistema elimina la tienda de la BD 5. El sistema cancela todos los pedidos asociados a esa tienda
Flujo alternativo	3a. Si el empresario no confirma, el caso de uso termina
Postcondición	El usuario ha eliminado una tienda

Tabla 37: CU-25 Eliminar una tienda

Identificador	CU-26 Ver lista de pedidos recibidos
Descripción	Caso de uso correspondiente al RF-24
Actores	Empresario
Precondición	El empresario tiene una sesión activa y está en el menú principal
Flujo principal	1. El empresario selecciona en el menú el apartado “Pedidos recibidos” 2. El sistema consulta y muestra la lista de pedidos recibidos por las tiendas del empresario
Flujo alternativo	
Postcondición	El empresario visualiza su lista de pedidos recibidos

Tabla 38: CU-26 Ver lista de pedidos recibidos

Identificador	CU-27 Filtrar lista de pedidos recibidos
Descripción	Caso de uso correspondiente al RF-25
Actores	Empresario
Precondición	El empresario visualiza su lista de pedidos recibidos. CU-26 Ver lista de pedidos recibidos
Flujo principal	1. El empresario pulsa el botón de Mostrar pedidos completados/cancelados 2. El sistema muestra una lista modificada de pedidos en base a los filtros
Flujo alternativo	
Postcondición	El empresario visualiza su lista de pedidos recibidos

Tabla 39: CU-27 Filtrar lista de pedidos recibidos

Identificador	CU-28 Cancelar pedido recibido
Descripción	Caso de uso correspondiente al RF-26
Actores	Empresario
Precondición	El empresario visualiza su lista de pedidos recibidos. CU-26 Ver lista de pedidos recibidos
Flujo principal	1. El empresario selecciona un pedido de la lista 2. El sistema muestra información del pedido seleccionado 3. El empresario pulsa el botón de cancelar pedido 4. El sistema cancela el pedido 5. El sistema notifica al cliente de la cancelación
Flujo alternativo	2a. El pedido no es cancelable (por ejemplo está completado) y el caso de uso termina.
Postcondición	El pedido seleccionado es cancelado

Tabla 40: CU-28 Cancelar pedido recibido

Identificador	CU-29 Intercambiar mensajes con cliente
Descripción	Caso de uso correspondiente al RF-27
Actores	Empresario
Precondición	El empresario visualiza su lista de pedidos recibidos. CU-26 Ver lista de pedidos recibidos
Flujo principal	1. El empresario selecciona un pedido de la lista 2. El sistema muestra información del pedido seleccionado 3. El empresario pulsa el botón de mostrar el chat 4. El sistema abre el menú del chat 5. El empresario escribe y envia un mensaje 6. El sistema guarda el mensaje
Flujo alternativo	*. Si el empresario pulsa el botón de retroceder entre los pasos 4 y 6 el sistema regresa al menú. El caso de uso vuelve al punto 1.
Postcondición	El empresario ha enviado un mensaje al cliente

Tabla 41: CU-29 Intercambiar mensajes con cliente

Identificador	CU-30 Avanzar estado de pedido recibido
Descripción	Caso de uso correspondiente al RF-28
Actores	Empresario
Precondición	El empresario visualiza su lista de pedidos recibidos. CU-26 Ver lista de pedidos recibidos
Flujo principal	1. El empresario selecciona un pedido de la lista 2. El sistema muestra información del pedido seleccionado 3. El empresario pulsa el botón de avanzar pedido 4. El sistema cambia el estado del pedido 5. El sistema notifica al cliente del cambio de estado
Flujo alternativo	2a. El pedido está completado o cancelado (no puede avanzar) y el caso de uso termina.
Postcondición	El pedido seleccionado avanza de estado

Tabla 42: CU-30 Avanzar estado de pedido recibido

5. Diseño

5.1. Arquitectura

5.1.1. Arquitectura Lógica

La arquitectura lógica consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el código fuente del software. Una arquitectura de software se selecciona y diseña con base en objetivos (requisitos) y restricciones.

Para elaborar el diseño de la arquitectura lógica de la aplicación se va a definir el número de capas de las que va a constar la aplicación. En nuestro caso hemos elegido una arquitectura de 3 capas (presentación, lógica y persistencia):

- Capa de presentación: la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura las acciones del usuario. Esta capa se comunica únicamente con la capa de negocio.
- Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.
- Capa de persistencia (datos): es donde residen los datos y es la encargada de acceder a los mismos. Está formada por los de bases de datos que realizan el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. En este caso se realiza a través del acceso a una base de datos situada en la nube (Firebase). Esta capa además incluye los servicios que interactúan con el sistema, en nuestro caso los servicios usados de Google Maps.

Aparte de la división por capas ya mostrada se van a aplicar patrones de software. A continuación se van a enumerar estos, con una explicación de cada uno y donde son usados:

- Patrón MVP - El patrón Modelo Vista Presentador se basa en separar la vista de la lógica de negocio (el modelo), para ello incluye un nuevo componente intermedio para realizar la comunicación entre ellas, los presentadores.
  - Modelo: Representa la capa de datos y la lógica de negocio. Contiene la información, pero no realiza acciones sobre ella.
  - Vista: Se encarga de mostrar la información al usuario gráficamente. Define la estructura que saldrá por pantalla e invoca los métodos de los presentadores. En un entorno de Android son conocidas como “Activity”.
  - Presentador: Realiza la comunicación entre la vista y el modelo. Controla la lógica de la vista y maneja las invocaciones al modelo para obtener los datos.

  En este patrón la Vista realiza la petición, luego el Presentador solicita información a la capa Modelo, cuando la información es devuelta al Presentador, este la entrega a la Vista.

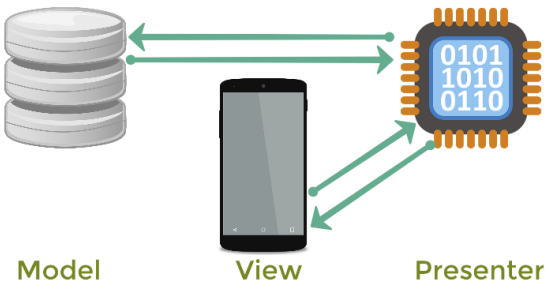


Imagen 3: Patrón MVP

- Patrón DAO - Este pretende independizar la aplicación de la forma de acceder a la base de datos, o cualquier otro tipo de repositorio de datos (en nuestro caso Firebase). Para ello se centraliza el código relativo al acceso al repositorio de datos en clases especificas para ello.

5.1.2. Arquitectura Física

Se encarga de representar la forma en que se distribuye nuestra aplicación a los usuarios finales, en la que se denotan los actores y el medio a través del cual se hace llegar la aplicación al computador y/o dispositivo del usuario.

En este proyecto se ha decidido usar un modelo cliente servidor. El lado del cliente será el dispositivo usado por el usuario para acceder a la aplicación y en el lado del servidor se encontrarán el componente de Firebase para acceder a la base de datos, junto a los servicios de Google Maps.

Este tipo de arquitectura cuenta con las siguientes ventajas:

- Seguridad
- Centralización de los datos
- Tecnologías más maduras y robustas

5.2. Modelos de Diseño

En este apartado se adjuntarán diferentes diagramas y representaciones que ayudarán a profundizar en el diseño empleado en nuestra herramienta.

5.2.1. Diagrama de Clases

Los diagramas de clases muestran la estructura de un sistema concreto al modelar sus clases, atributos, operaciones y relaciones entre objetos.

Para una mejor visualización y comprensión del diagrama se ha decidido dividir este en varios, según el patrón MVP:

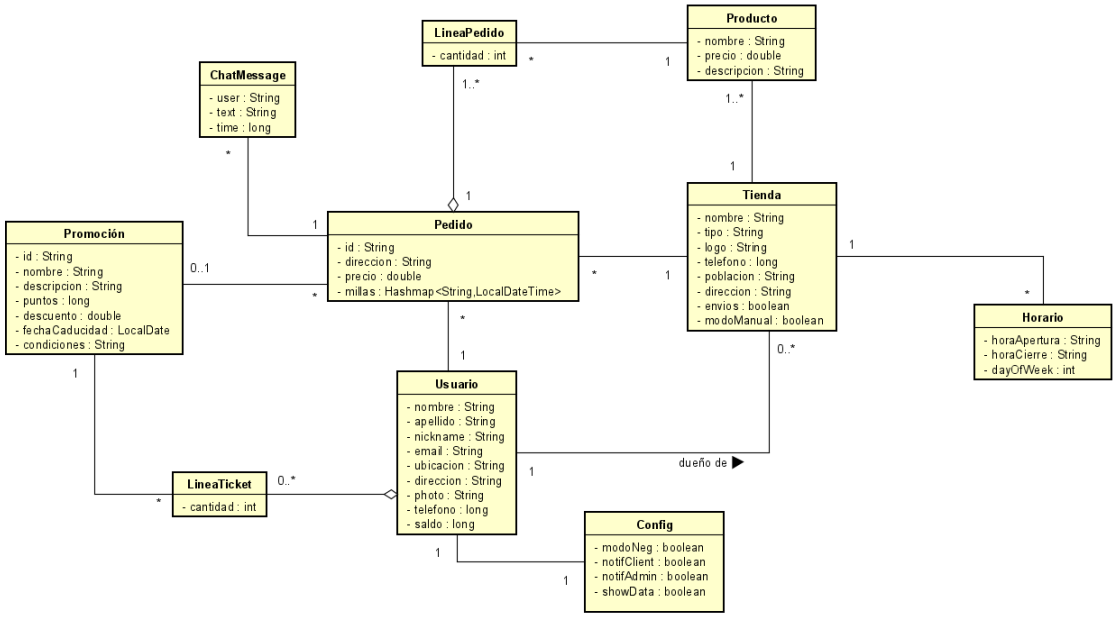


Imagen 4: Diagrama de clases del modelo

A continuación se va a explicar las distintas clases del diagrama:

- Usuario - Información de cada usuario registrado en el sistema.
- Config - Ajustes de cada usuario dentro de la aplicación.
- Tienda - Información de las tiendas guardadas en el sistema.
- Horario - Representa un periodo de tiempo donde la tienda está abierta.
- Pedido - Información de los pedidos realizados dentro de la aplicación.
- ChatMessage - Representa cada mensaje del chat asociado a un pedido.
- Producto - Información de cada producto registrado en el sistema.
- LineaPedido - Representa cada línea de un pedido, es decir cada par cantidad-producto de este.
- Promocion - Información de cada promoción que existe en el sistema.
- LineaTicket - Representa cada par cantidad-promoción que tiene adquirido un usuario.

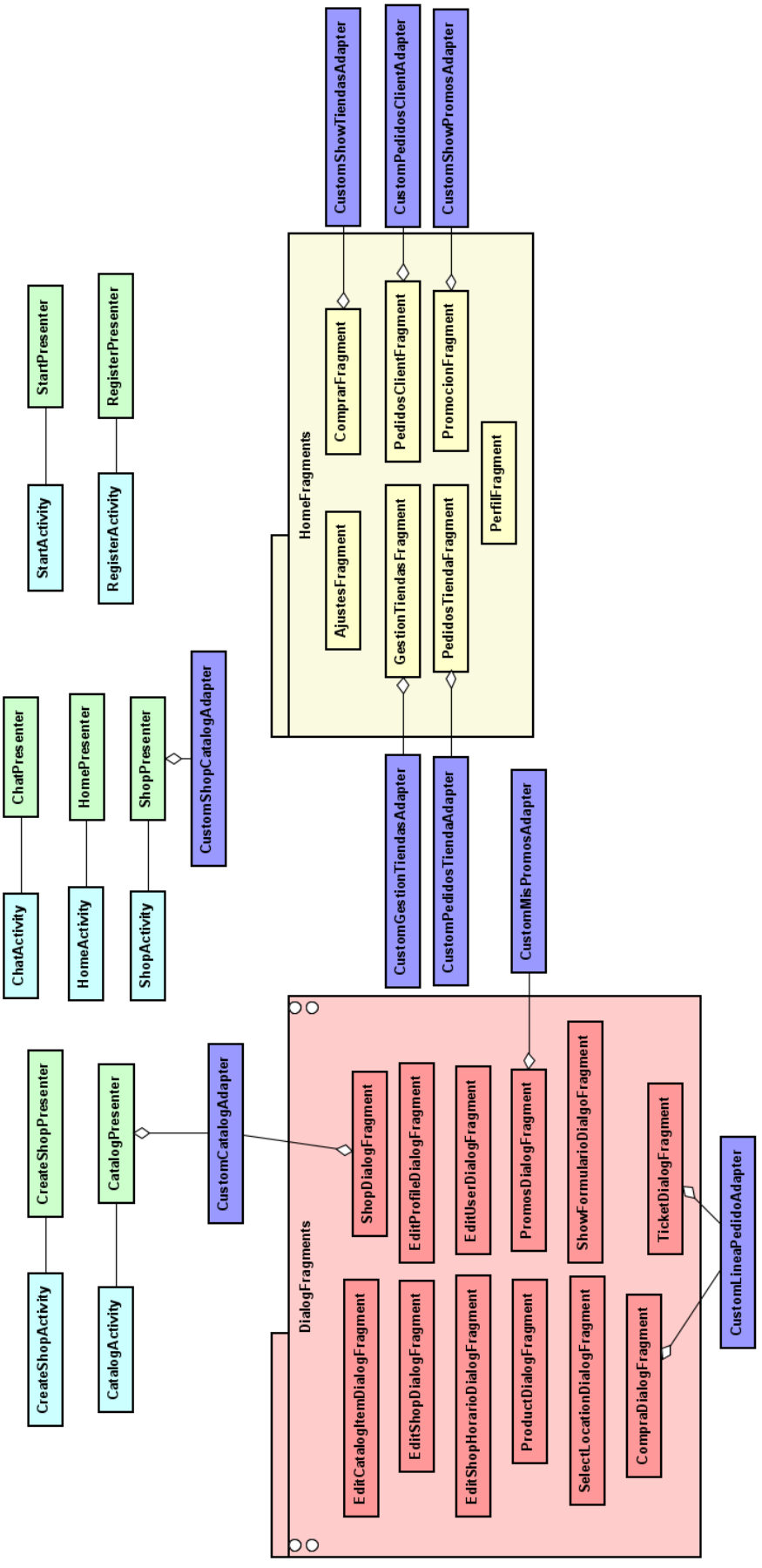


Imagen 5: Diagrama de clases de la vista y presentadores

En las clases de las vistas y presentadores podemos hacer las siguientes diferencias:

- Activity - Clase que representa las actividades, encargadas de comunicar la vista. Representadas en el diagrama por el color celeste. Todas ellas extienden de *AppCompatActivity*
- Presenter - Clase que representa los comunicadores entre las vistas y el modelo. Controla la lógica de la vista y maneja las invocaciones al modelo para obtener los datos. Representados en el diagrama por el color verde.
- HomeFragments - Fragmento que representa cada uno de los "submenús" de la actividad *HomeActivity*. Se explicará más a fondo el funcionamiento de estos al hablar de la interfaz. Representados en el diagrama por el color amarillo.
- DialogFragments - Fragmento que muestra una ventana de diálogo, "flotando" sobre la ventana de la actividad desde la que son invocados. Se explicará más a fondo el funcionamiento de estos al hablar de la interfaz. Representados en el diagrama por el color rojo. Todos ellos extienden de *AppCompatDialogFragment*
- Adapters - Clase que, como su nombre indica, adapta una lista (ArrayList, HashMap, etc.) a la interfaz de la pantalla. Se explicará más a fondo el funcionamiento de estos al hablar de la interfaz. Representados en el diagrama por el color morado.

Aparte de estas clases, hay 2 clases más, las cuales se van a mencionar aquí:

- La primera es la clase referente al patrón DAO, es decir, la clase encargada de realizar las conexiones con la base de datos. Esta es la clase *DBAccess* y contiene toda la lógica de lectura y escritura de toda la información guardada en la base de datos.
- La segunda es una clase auxiliar, llamada *Utils*, la cual contiene métodos y funciones utilizados en muchas partes del proyecto, las cuales se encuentran en esta para evitar duplicaciones del código.

  Un ejemplo de esto es la conversión del dinero de valor numérico a cadena de texto con el formato correcto (con 2 decimales y el símbolo del €), la cual necesitamos cada vez que queremos mostrar el dinero por pantalla, lo cual se hace en muchas vistas distintas.

5.2.2. Diagramas de Secuencia

Los diagramas de secuencia son un tipo de diagrama de interacción, ya que describen cómo, y en qué orden un grupo de objetos funcionan en conjunto. Se centran específicamente en líneas de vida o en los procesos y objetos que coexisten simultáneamente, y los mensajes intercambiados entre ellos para ejecutar una función antes de que la línea de vida termine.

En este documento no se van a mostrar los diagramas de secuencia de todos los casos de uso, si no que se van a escoger los considerados más importantes dentro del proyecto.

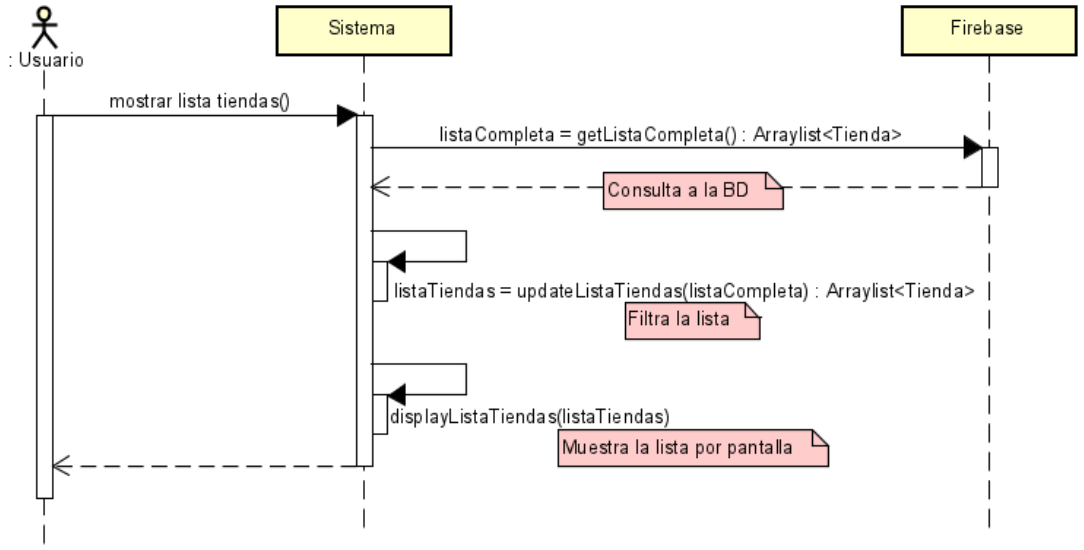


Imagen 6: Diagrama de secuencia CU-04 Ver lista de tiendas

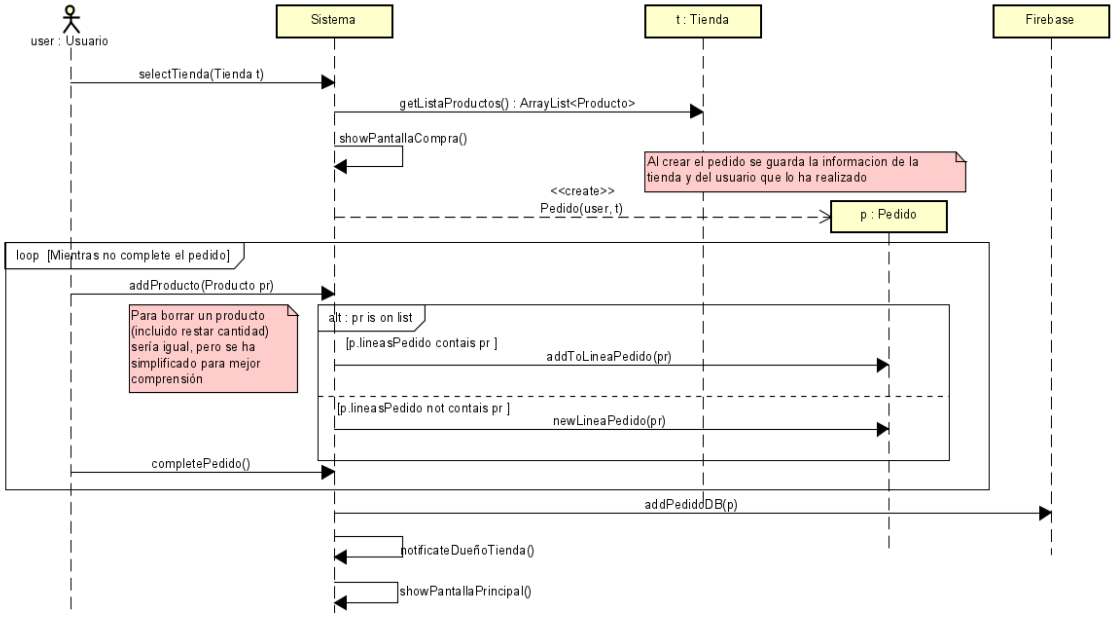


Imagen 7: Diagrama de secuencia CU-07 Seleccionar una tienda para pedir y CU-08 Preparar pedido

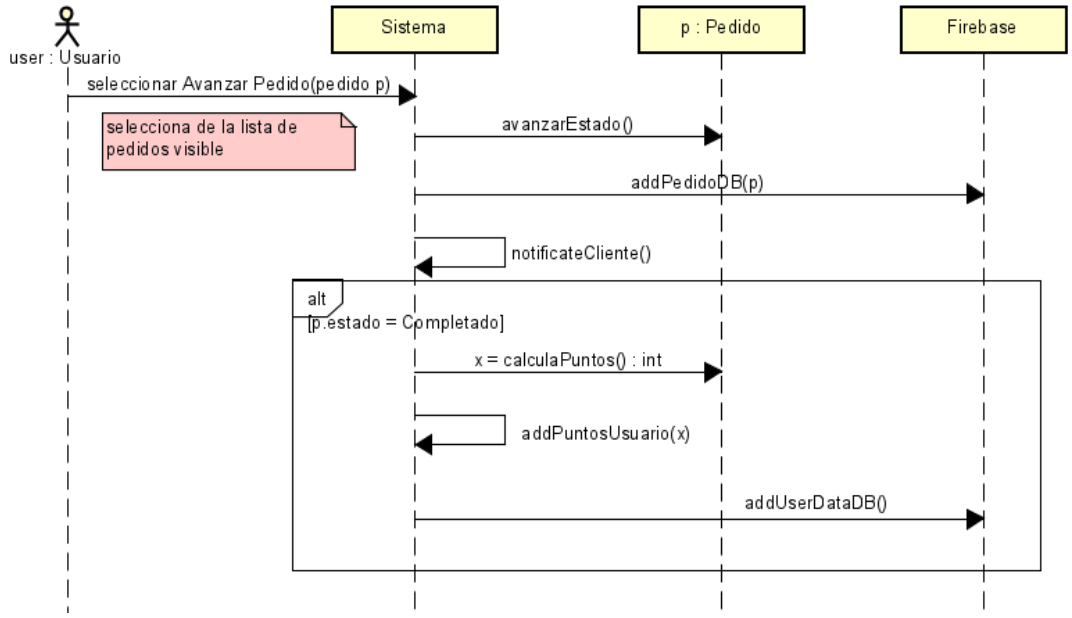


Imagen 8: Diagrama de secuencia CU-30 Avanzar estado de pedido recibido

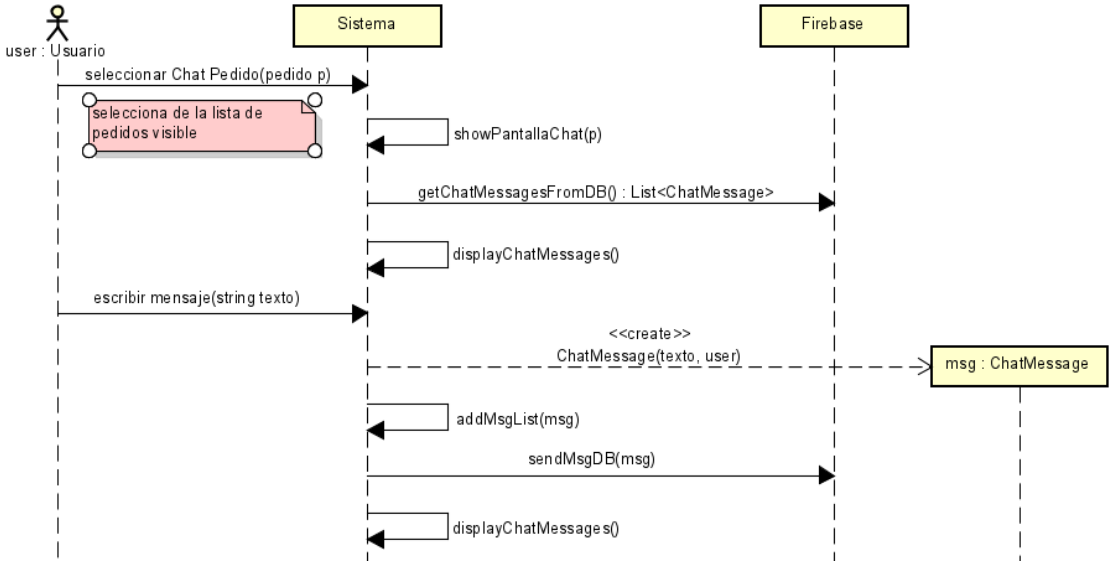


Imagen 9: Diagrama de secuencia CU-12 Intercambiar mensajes con tienda o CU-29 Intercambiar mensajes con cliente

5.3. Diseño de la Base de Datos

Aunque dentro del servicio de Firebase usamos varias herramientas, las cuales explicaremos en detalle más adelante, en este apartado nos vamos a centrar en la información guardada en Firebase Cloud Firestore, ya que es la base de datos principal donde guardamos la mayor parte de datos del programa.

En este servicio guardamos en colecciones toda la información de usuarios, pedidos tiendas y promociones según las tablas vistas a continuación, para cada una de estas se verá el nombre del campo, una descripción, el tipo de dato, si estos pueden ser nulos, si son únicos dentro del sistema junto con un ejemplo de cada uno. También se va a mostrar las claves primarias (PK) y foraneas (FK) que haya en cada tabla.

Usuario

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
nombre	Nombre real del usuario	string	No	No	"Daniel"
apellido	Apellido/s del usuario	string	No	No	"Melendre"
email *PK*	Correo electrónico del usuario	string	Si	No	"daniel@gmail.com"
nickname	Nombre de usuario usado dentro de la aplicación	string	Si	No	"danimele10"
ubicacion	Localidad donde se encuentra el usuario	string	No	No	"Madrid"
direccion	Direccion real del usuario	string	No	Si	"Calle Mayor,4,1ºB"
telefono	Teléfono del usuario	number	No	Si	987654321
photo	Enlace de Firebase Storage donde se encuentra guardada la foto de perfil	string	No	Si	"urlFirebase/images/users/daniel@gmail.com.png"
saldo	Puntos de la aplicación acumulados por el usuario	number	No	No	1250
config	Configuraciones del usuario dentro de la aplicación, cada una tiene un nombre y un valor (true o false)	Map (string, boolean)	No	No	modoNeg:true notifAdmin:true notifClient:true showData:true
misPromos	Tickets de promociones guardados en la cuenta del usuario,se guarda el id de la promo y la cantidad, cada id es *FK* de la promo a la que referencia	Map (string, number)	No	Si	usmfj28knAOSDMK:1 SEMC2314dadwdqF:2

Tabla 43: Organización de la tabla de usuarios de la base de datos

Tienda

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
nombre *PK*	Nombre de la tienda	string	Si	Si	"Bar Manolo"
poblacion	Localidad donde se encuentra la tienda	string	No	No	"Madrid"
direccion	Direccion real de la tienda	string	No	No	"Calle Mayor,4,1ºB"
dueño *FK*	Nombre de usuario dueño de la tienda	string	No	No	"danimele10"
telefono	Teléfono de la tienda	number	No	No	987654321
tipo	Tipo de la tienda, tiene que ser uno de los establecidos por la aplicación	string	No	No	"Bar-Restaurante"
envios	La tienda permite llevar pedidos a casa de los usuarios	boolean	No	No	true
modoMan	La tienda es localizada automáticamente o manualmente	boolean	No	No	false
horario	Franjas horarias en las que la tienda está abierta, cada cadena separada por "," representa una franja	string	No	No	13:00;16:00;6, 13:00;16:00;7
photo	Enlace de Firebase Storage donde se encuentra guardada la foto de de la tienda	string	No	Si	"urlFirebase/images/ tiendas/bar-manolo.png"
catalogo	Lista de productos de la tienda	List (Producto)	No	No	(se ve más abajo)

A continuación se va a ver como se guarda cada producto (dentro de la tabla de tiendas):

Producto

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
nombre	Nombre del producto	string	Si*	No	"Sepia"
descripción	Breve descripción del producto	string	No	No	"Sepia rebozada"
precio	Precio del producto	number	No	No	2.50

** unico dentro de cada tienda*

Tabla 44: Organización de la tabla de tiendas de la base de datos

Pedido

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
id *PK*	Cadena de 15 caracteres que identifica cada pedido	string	Si	No	“09J6mBsOpQS0QOe”
nickUser *FK*	Usuario que realiza el pedido	string	No	No	“danimele10”
tienda *FK*	Tienda donde se realiza el pedido	string	No	No	“Bar Manolo”
precio	Precio total del pedido	number	No	No	16.30
promocion	Promocion asignada al pedido	string	No	Si	“HgecNIvYn2JoELp”
direccion	Direccion de entrega del pedido	string	No	Si	“Calle Mayor,4,1ºB”
lineasPedido	Lista de productos pedidos	Map (string, number)	No	No	Sepia:2 Patatas:1
millas	Lista de estados por lo que pasa el pedido	Map (string, string)	No	No	Cancelado:null Completado: “13-05-2022 11:32:58” En cola: “13-05-2022 11:32:41” Haciendo: “13-05-2022 11:32:58” Listo: “13-05-2022 11:32:58”

Tabla 45: Organización de la tabla de pedidos de la base de datos

Promocion

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
id *PK*	Cadena de 15 caracteres que identifica cada promocion	string	Si	No	“jH7ipXSTm1aOX57”
nombre	Nombre de cada promocion	string	Si	No	‘Descuento”
descripción	Descripción vista por el cliente de la promoción	string	No	No	“Texto largo explicando la promoción”
condiciones	Condiciones comprobadas por el sistema de una promocion	string	No	No	“categoria:Otros”
descuento	Descuento que se aplica al usar la promoción	number	No	No	15.50
puntos	Cantidad de puntos que cuesta adquirir la promoción	number	No	No	200
fechaCadudidad	Fecha en la que la promoción deja de estar disponible para adquirir	string	No	No	“28-06-2022”

Tabla 46: Organización de la tabla de promociones de la base de datos

5.4. Diseño de Interfaces

En este apartado se van a enumerar todas las interfaces de la aplicación, mostrando la información que muestra cada una así como el interactuar con ellas.

Antes de eso, se va a hablar ligeramente sobre algunos de los componentes más interesantes utilizados en el desarrollo de las interfaces:

- ListView - Permite mostrar listas de layouts por pantalla. Para configurar estos layouts se usan “Adapters” personalizados para mostrar una interfaz concreta sobre cada elemento de una lista que tengamos (tiendas, pedidos, promociones, productos, etc.). Estos adapters son creados y asignados a cada listview y en su creación se les asigna los valores de la lista a la interfaz de cada elemento de la interfaz.
- Spinner - Menú desplegable, permite seleccionar una opción entre varias. Se usa para seleccionar tipos de tiendas y ordenar el catálogo.

Para todas las pantallas de las que se va a hablar a continuación existen 2 versiones, la aplicación puede detectar si el dispositivo tiene el modo oscuro del sistema activado y cambiar su interfaz en base a ello, como se muestra en [**10**]

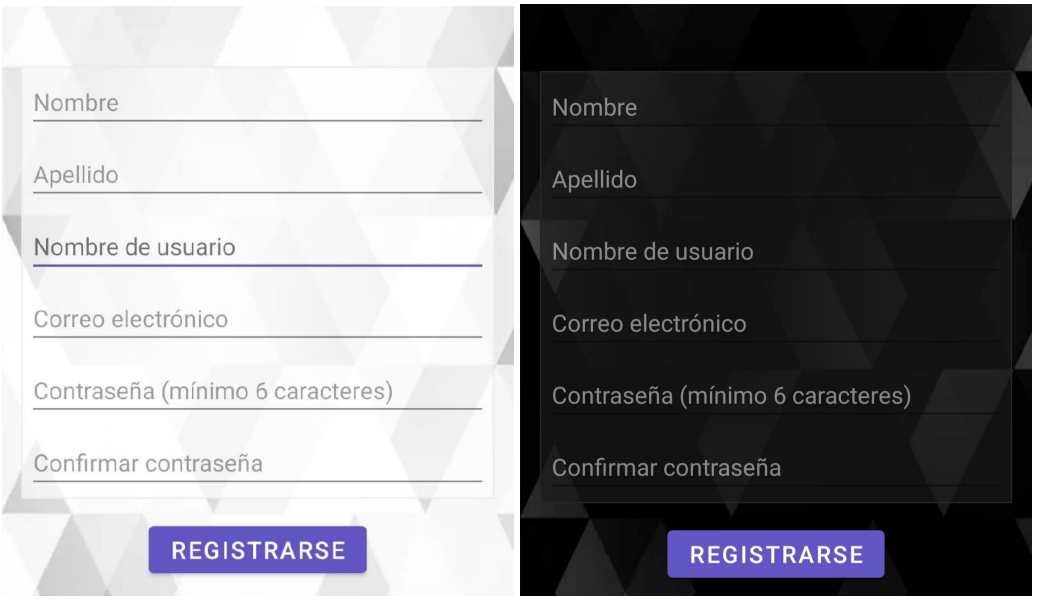


Imagen 10: Comparacíon de modo claro y modo oscuro

5.4.1. Inicio de sesión/Registro

Interfaz de inicio de sesión

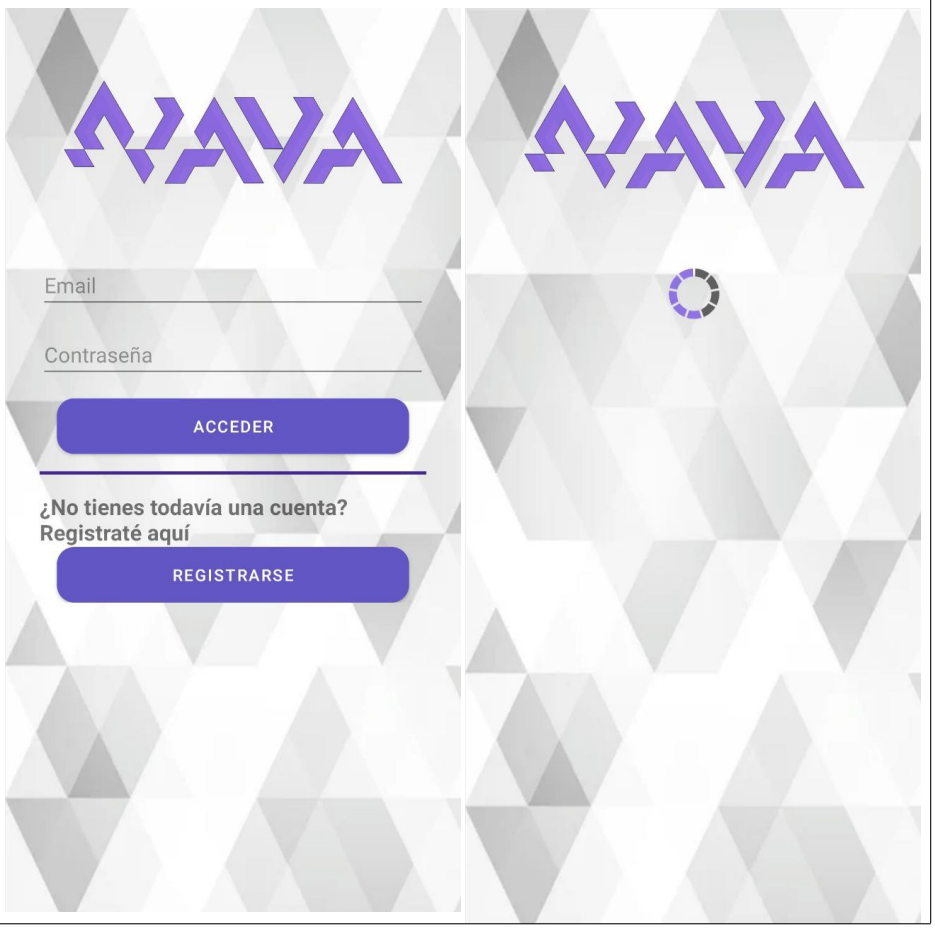
Descripción	Pantalla mostrada al iniciar la aplicación y en la que el usuario puede iniciar sesión. Muestra el logo de la aplicación, y dependiendo de si se tiene una sesión guardada muestra una interfaz u otra. Si se inicia teniendo una sesión guardada mostrará un gif de carga en lo que verifica la sesión, si no mostrará los campos de texto (EditText) y un botón para iniciar sesión manualmente, junto con un botón para poder registrarse.
Eventos	▪ Si se inicia sesión, ya sea automáticamente o introduciendo los datos manualmente y pulsando el botón se mostrará la pantalla principal de la aplicación. ▪ Si al introducir los datos no se consigue iniciar sesión, se muestra un cuadro de alerta informando al usuario. ▪ Si el usuario pulsa el botón de registrarse, se mostrará la Interfaz de registro de usuario
Captura/s	

Tabla 47: Interfaz de inicio de sesión

Interfaz de registro de usuario

Descripción	Pantalla a la que través de la cual el usuario puede registrarse en el sistema. Contiene unos campos de texto (EditText) donde introducir los datos y un botón de “registrarse”
Eventos	▪ Si el usuario pulsa el botón de registrarse se comprueban que los datos introducidos (en los EditText) sean válidos. ▪ Si lo son, se registra al usuario en el sistema y se le lleva a la pantalla principal de la aplicación. ▪ Si al introducir los datos no son válidos, se muestra un cuadro de alerta informando al usuario. ▪ Si pulsa el botón del teléfono de volver atrás, vuelve a Interfaz de inicio de sesión
Captura/s	

Tabla 48: Interfaz de registro de usuario

5.4.2. Pantalla principal

Una vez el usuario está iniciado sesión se mostrará la pantalla principal de la aplicación, esta se divide en varias pantallas pero todas comparten una “toolbar” en la parte superior , donde se muestra el nombre del fragmento mostrado y el icono que abre el menú donde seleccionar estas sub-pantallas.

El menú es un “Navigation Drawer”, el típico menú lateral deslizante desde la izquierda y utilizado como uno de los principales elementos de navegación de las aplicaciones móviles.

Interfaz del menú de la pantalla principal

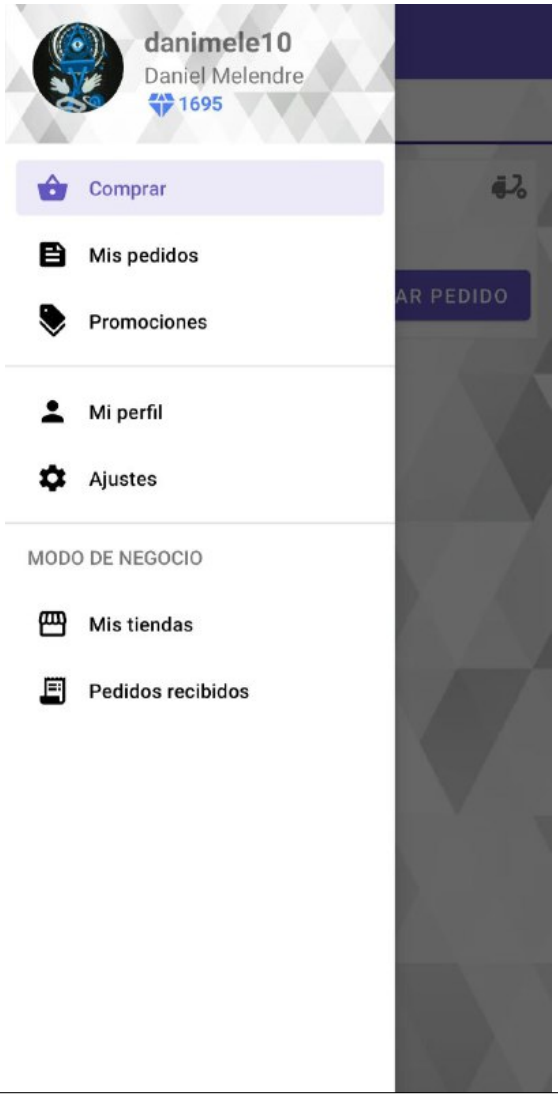
Descripción	Se usa en la pantalla principal de la aplicación y aparte del botón del menú para desplegarlo, puede ser abierto deslizando el contenido de la pantalla desde el extremo izquierdo. En él se encuentra una cabecera donde se ve información sobre el usuario, como su nombre real, nombre de usuario, icono y el saldo actual de puntos. Debajo de esta se encuentra una lista con las diferentes pantallas seleccionables. Si el usuario no tiene activado el modo de negocio no se mostrará la última sección.
Eventos	▪ Si el usuario selecciona un elemento, se cargará la pantalla asignada a ese elemento. ▪ Si pulsa el botón del teléfono de volver atrás, se cierra el menú desplegado.
Captura/s	

Tabla 49: Interfaz del menú de la pantalla principal

A continuación se van a mostrar todas las pantallas contenidas dentro de la pantalla principal.

Interfaz de lista de tiendas

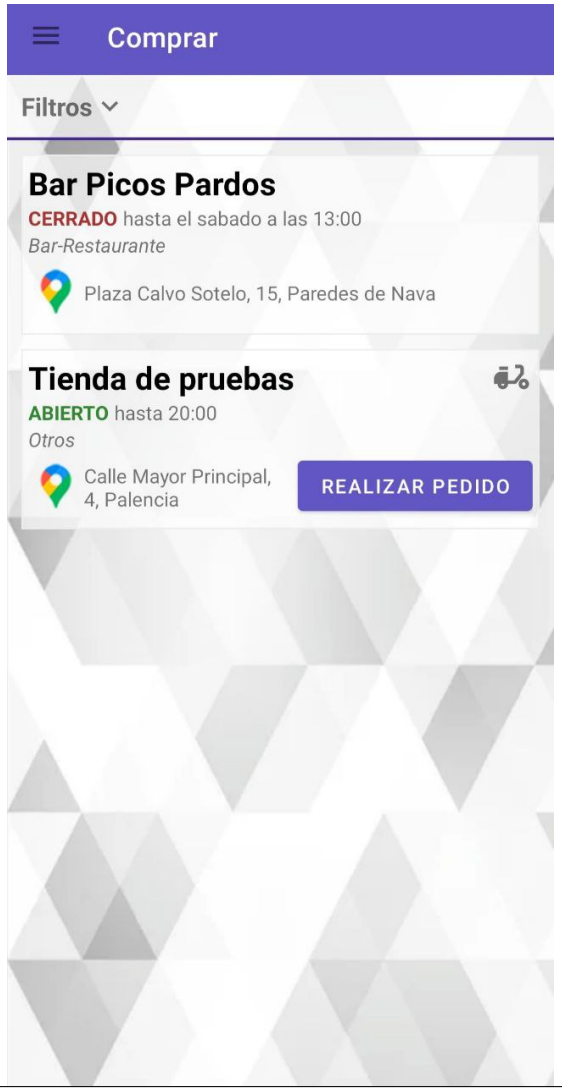
Descripción	Muestra la lista de tiendas disponibles (usando un listview). Para cada tienda muestra información como nombre, horario, tipo de tienda y localización, junto con un botón de realizar pedido (si la tienda está abierta).
Eventos	▪ Si el usuario pulsa sobre una tienda, se le mostrará Interfaz del dialog de tienda. ▪ Si pulsa el botón de realizar pedido, se le mostrará Interfaz para realizar pedido. ▪ Si pulsa sobre la dirección, se le abrirá Google Maps con la dirección del local. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 50: Interfaz de lista de tiendas

Interfaz de lista de pedidos realizados

Descripción	Muestra la lista de pedidos (usando un listview) realizados previamente por el usuario. Para cada pedido muestra información como nombre de la tienda, última actualización y precio.
Eventos	▪ Si el usuario pulsa "Mostrar pedidos completados/cancelados" se cambiará la lista mostrada, en base a lo seleccionado. ▪ Si el usuario pulsa sobre un pedido, se le mostrará Interfaz del dialog de pedido. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 51: Interfaz de lista de pedidos realizados

Interfaz de promociones

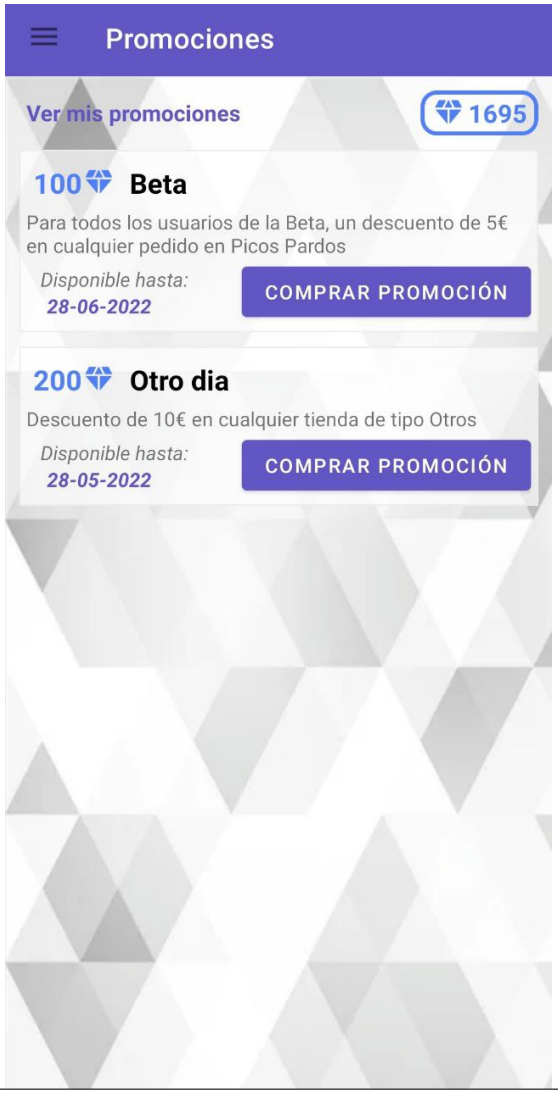
Descripción	Muestra la lista de promociones (usando un listview) disponibles, junto con los puntos del usuario. Para cada promoción muestra información como precio (en puntos), nombre, descripción, fecha de caducidad y un botón de Comprar.
Eventos	▪ Si el usuario pulsa "Ver mis promociones" se mostrará Interfaz del dialog de mis promociones. ▪ Si el usuario pulsa el botón de Comprar, comprará una promoción y se le restarán los puntos. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 52: Interfaz de promociones

Interfaz del perfil

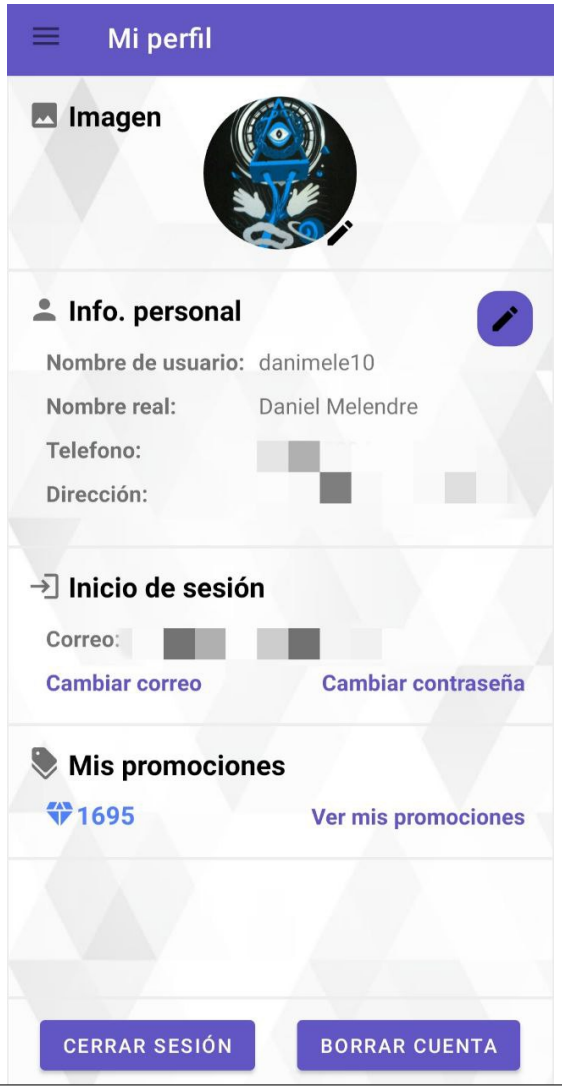
Descripción	Muestra la información del perfil de usuario, aquí se ve la imagen, información personal (ocultada en la captura) y datos de la cuenta.
Eventos	▪ Si el usuario pulsa editar imagen, se le mostrará la galería para seleccionar una nueva foto. ▪ Si el usuario pulsa editar información personal, se mostrará Interfaz del dialog editar información personal. ▪ Si el usuario pulsa cambiar correo o contraseña, se mostrará Interfaz del dialog editar campo con el campo. ▪ Si el usuario pulsa "Ver mis promociones" se mostrará Interfaz del dialog de mis promociones. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 53: Interfaz del perfil

Interfaz de ajustes

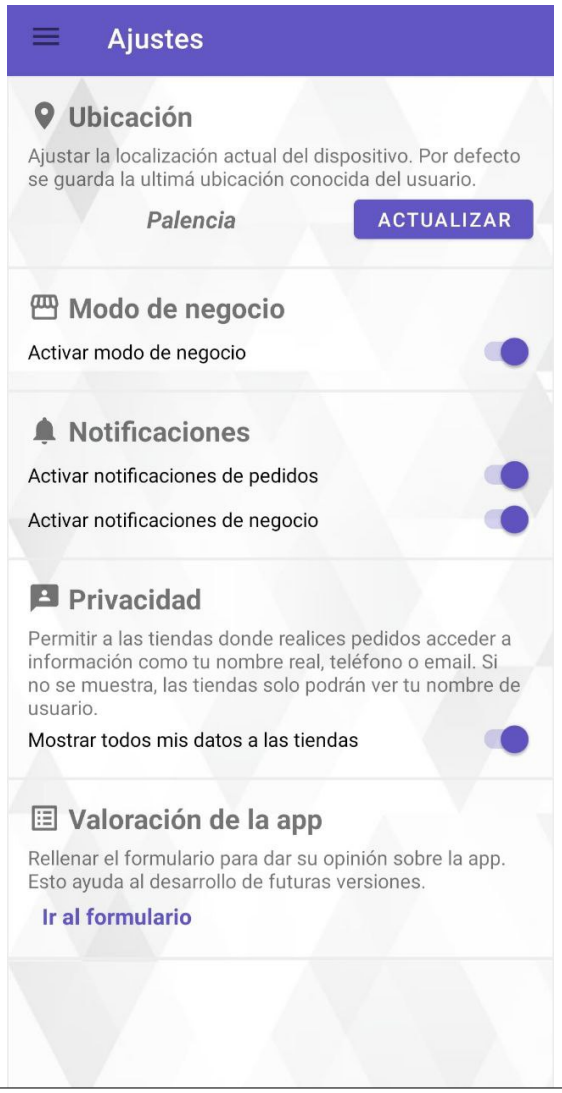
Descripción	Muestra los ajustes del usuario en la aplicación.
Eventos	▪ Si el usuario pulsa “Actualizar”, se actualizará su ubicación guardada. ▪ Si el usuario pulsa uno de los checkbox, se cambiará ese ajuste en su cuenta. ▪ Si el usuario pulsa “Ir al formulario” se abrirá un formulario externo. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 54: Interfaz de ajustes

Interfaz de lista de tiendas creadas

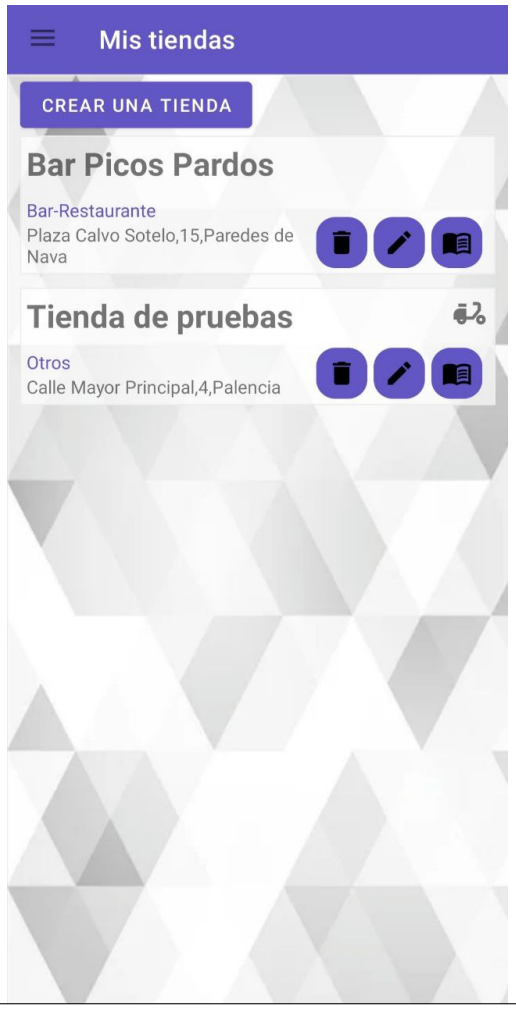
Descripción	Muestra la lista de tiendas creadas por el usuario (empresario). Para cada tiendas se muestra el nombre, tipo y ubicación, junto con los botones de borrar, editar y catálogo.
Eventos	▪ Si el usuario pulsa sobre una tienda, se le mostrará Interfaz del dialog de tienda. ▪ Si el usuario pulsa borrar, se le mostrará un cuadro para confirmar (si acepta, se borrará la tienda). ▪ Si el usuario pulsa editar, se le mostrará Interfaz de edición de tiendas para esa tienda. ▪ Si el usuario pulsa catálogo, se le mostrará Interfaz de edición de catálogo de una tienda para esa tienda. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 55: Interfaz de lista de tiendas creadas

Interfaz de lista de pedidos recibidos

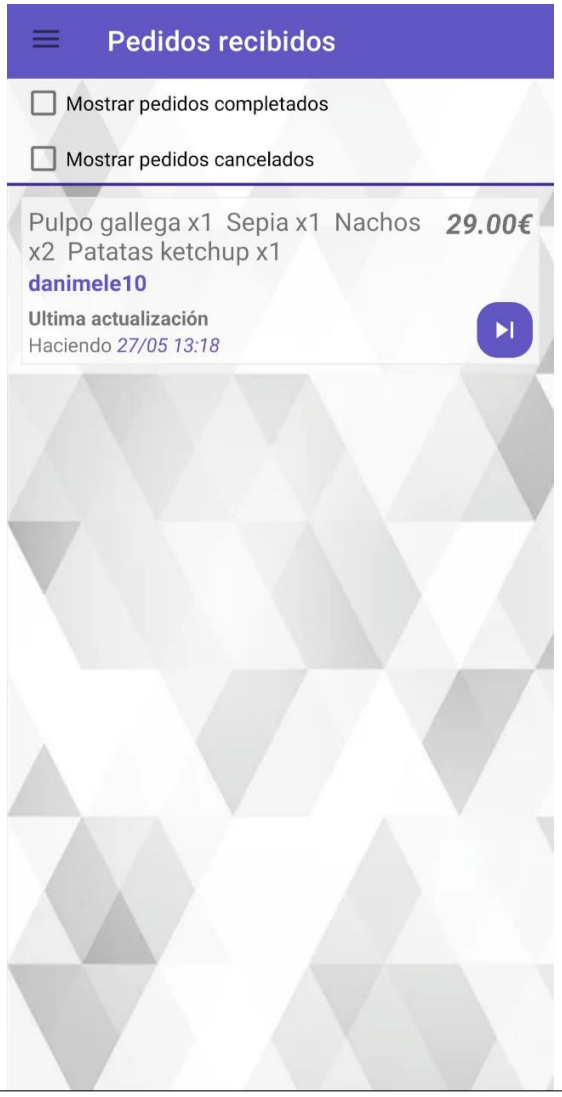
Descripción	Muestra la lista de pedidos en tiendas creadas por el usuario (empresario). Para cada pedido muestra información como los productos pedidos, nombre del cliente, última actualización y precio. También muestra un botón para avanzar de estado de pedido.
Eventos	▪ Si el usuario pulsa "Mostrar pedidos completados/cancelados" se cambiará la lista mostrada, en base a lo seleccionado. ▪ Si el usuario pulsa sobre un pedido, se le mostrará Interfaz del dialog de pedido. ▪ Si el usuario pulsa el botón de avanzar, el sistema cambiará el estado del pedido. ▪ Si pulsa el botón del teléfono de volver atrás, se abre el menú.
Captura/s	

Tabla 56: Interfaz de lista de pedidos recibidos

5.4.3. Otras pantallas

Interfaz para realizar pedido

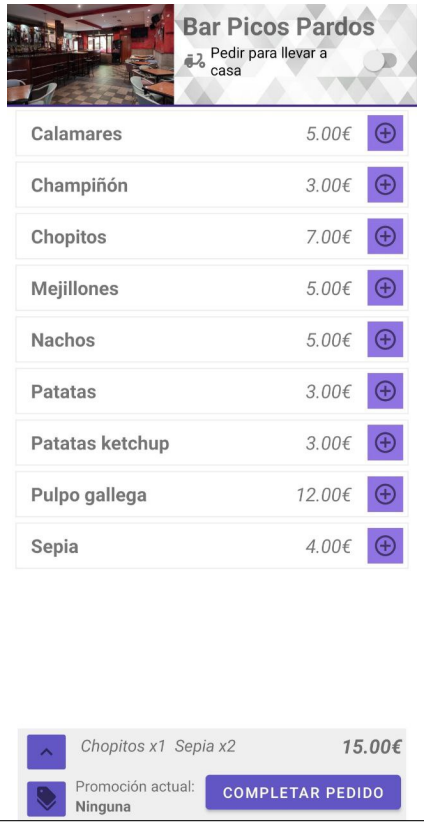
Descripción	Muestra la interfaz para realizar un pedido.En la parte superior se ve información de la tienda y el selector de pedir para llevar a casa, en la parte central se ve la lista de productos de esa tienda y en la inferior información sobre el pedido.
Eventos	▪ Si el usuario pulsa sobre un producto, se le mostrará Interfaz del dialog editar producto ▪ Si el usuario pulsa añadir, se le añadirá un producto al pedido. ▪ Si el usuario pulsa ver detalles, se le mostrará Interfaz del dialog de modificar pedido. ▪ Si el usuario pulsa promoción, se le mostrará Interfaz del dialog de mis promociones. ▪ Si el usuario pulsa completar, se guardará el pedido en el sistema y volverá al menú principal. ▪ Si pulsa el botón del teléfono de volver atrás, se le mostrará un cuadro para confirmar (si acepta, se volverá a la pantalla anterior).
Captura/s	

Tabla 57: Interfaz para realizar pedido

Interfaz del chat asociado a un pedido

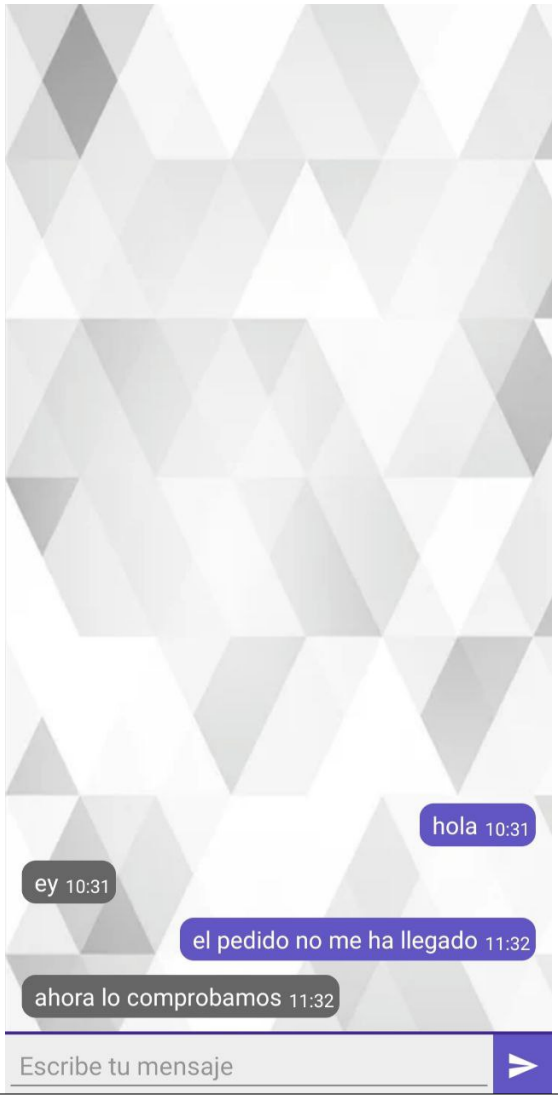
Descripción	Muestra un chat asociado a un pedido con los mensajes entre el cliente y la tienda, muestra los mensajes “propios” a la derecha en morado y los del “otro” a la izquierda en gris. En la parte inferior hay un cuadro de texto y un botón para enviar mensajes.
Eventos	▪ Si el usuario envía un mensaje, se guardará en el sistema y se actualizará. ▪ Si pulsa el botón del teléfono de volver atrás, se regresa a la pantalla anterior.
Captura/s	

Tabla 58: Interfaz del chat asociado a un pedido

Interfaz de edición de tiendas

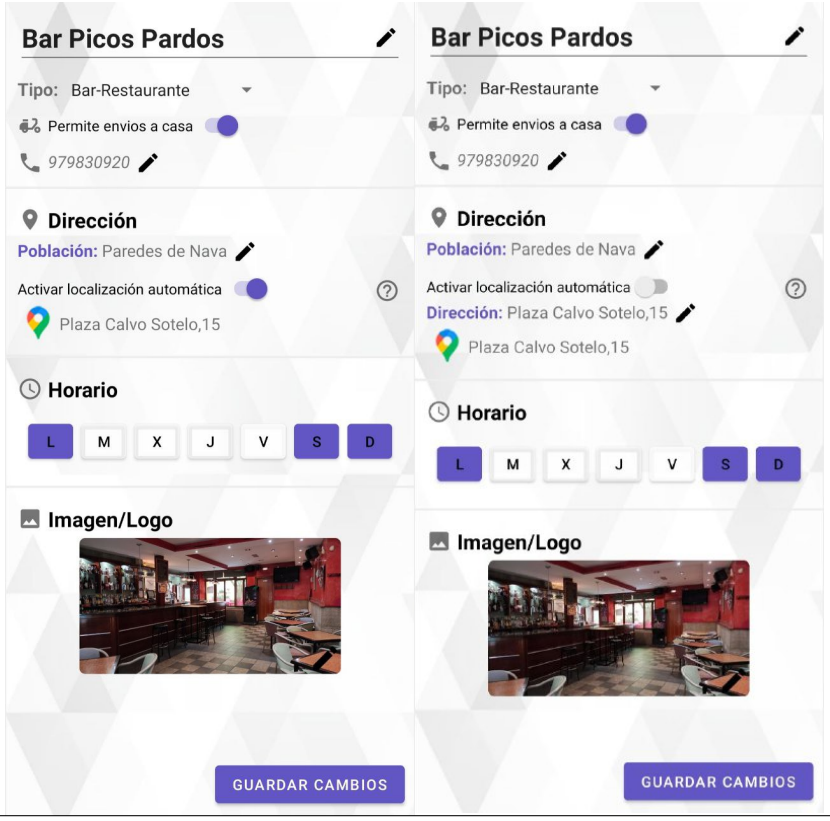
Descripción	Muestra la información de la tienda para ser modificada.
Eventos	▪ Si el usuario seleccina cambiar algún campo de texto (telefono, población, dirección manual), se le mostrará Interfaz del dialog editar campo ▪ Si el usuario selecciona tipo (usando un Spinner), se le mostrará los tipos disponibles, la opción seleccionada será el tipo de la tienda. ▪ Si el usuario cambia la localización se mostrará (o no) la dirección manual para introducir. ▪ Si se pulsa sobre la dirección, se le abrirá Google Maps con la dirección cargada. ▪ Si el usuario pulsa alguno de los botones de días, se mostrará Interfaz del dialog editar horario para ese día. ▪ Si el usuario pulsa editar imagen, se le mostrará la galería para seleccionar una nueva foto. ▪ Si pulsa el botón del teléfono de volver atrás, se le mostrará un cuadro para confirmar (si acepta, se volverá a la pantalla anterior).
Captura/s	

Tabla 59: Interfaz de edición de tiendas

Interfaz de edición de catálogo de una tienda

Descripción	Muestra la información del catálogo de una tienda,es decir una lista de productos con su precio.
Eventos	▪ Si el usuario pulsa sobre un producto, se le mostrará Interfaz del dialog editar producto ▪ Si el usuario pulsa añadir, se le mostrará Interfaz del dialog de producto. ▪ Si el usuario pulsa guardar, se guardarán los cambios realizados. ▪ Si el usuario selecciona ordenar (usando un Spinner), se le mostrará las formas de ordenar (nombre y precio ascendente y descendente); la seleccionada será la forma de ver los productos. ▪ Si el usuario pulsa editar un producto, se le mostrará Interfaz del dialog de producto. ▪ Si el usuario pulsa borrar un producto, este se borrará. ▪ Si pulsa el botón del teléfono de volver atrás, se le mostrará un cuadro para confirmar (si acepta, se volverá a la pantalla anterior).
Captura/s	

Tabla 60: Interfaz de edición de catálogo de una tienda

5.4.4. Cuadros de dialogo

Un DialogFragment es un fragmento que muestra una ventana de diálogo, flotando sobre la ventana de la actividad dejando esta en "segundo plano".

Estos son usados en la aplicación para mostrar información detallada sobre tiendas, pedidos, etc o para abrir cuadros de edición.

Interfaz del dialog de valoración

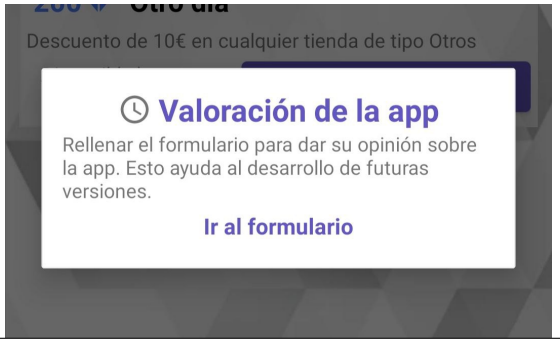
Descripción	Muestra un cuadro preguntando por rellenar el formulario sobre la aplicación. Se muestra aleatoriamente al cambiar de pestaña.
Eventos	▪ Si el usuario pulsa "Ir al formulario" se abrirá un formulario externo. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 61: Interfaz del dialog de valoración

Interfaz del dialog de tienda

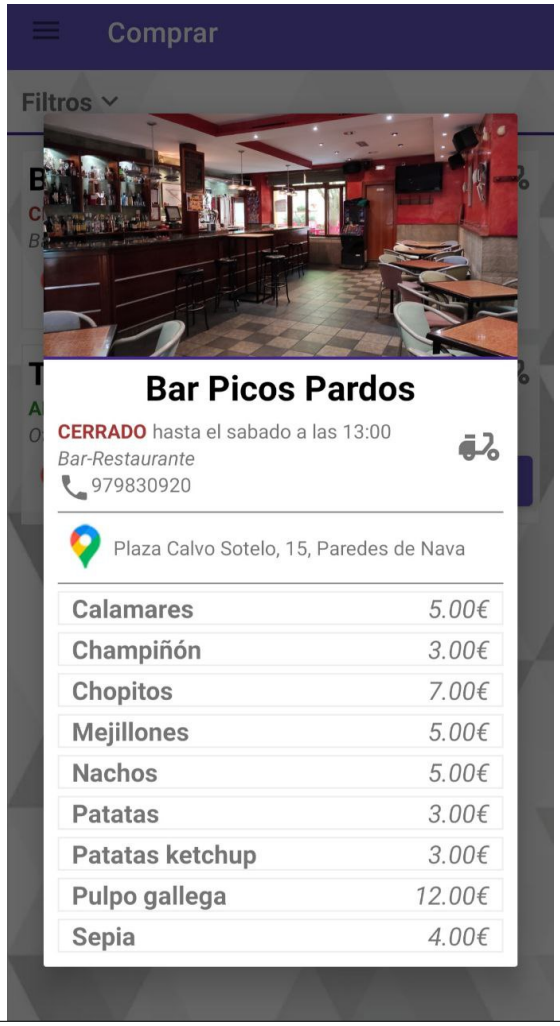
Descripción	Muestra información detallada sobre una tienda. Para la tienda seleccionada muestra información como imagen, nombre, horario, teléfono, tipo de tienda y localización, junto con un botón de realizar pedido (si la tienda está abierta).
Eventos	▪ Si pulsa el botón de realizar pedido, se le mostrará Interfaz para realizar pedido. ▪ Si el usuario pulsa sobre un producto, se le mostrará Interfaz del dialog editar producto ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 62: Interfaz del dialog de tienda

Interfaz del dialog de pedido

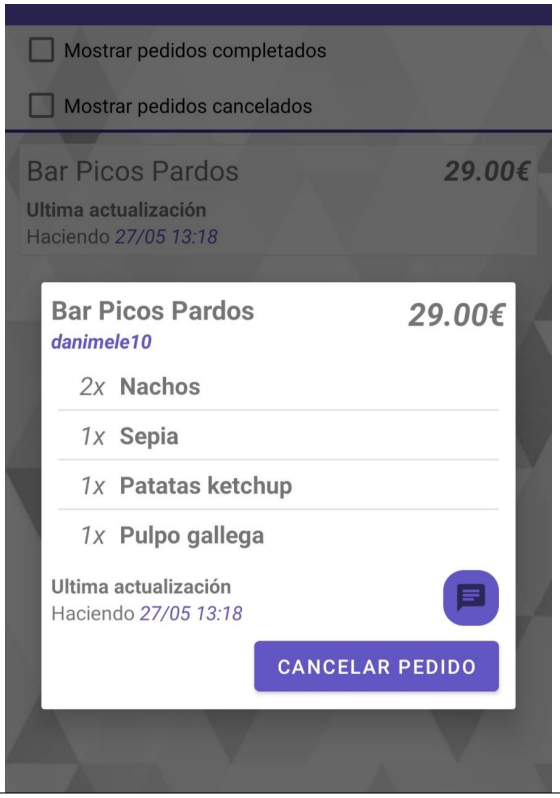
Descripción	Muestra información detallada sobre un pedido realizado o recibido. Para el pedido muestra información como los productos, cantidad de cada uno, la última actualización, precio; junto con un botón de cancelar pedido (si se puede) y uno para el chat.
Eventos	▪ Si pulsa el botón de cancelar, se cancelará el pedido. ▪ Si pulsa el botón del chat, se le mostrará Interfaz del chat asociado a un pedido. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 63: Interfaz del dialog de pedido

Interfaz del dialog de mis promociones

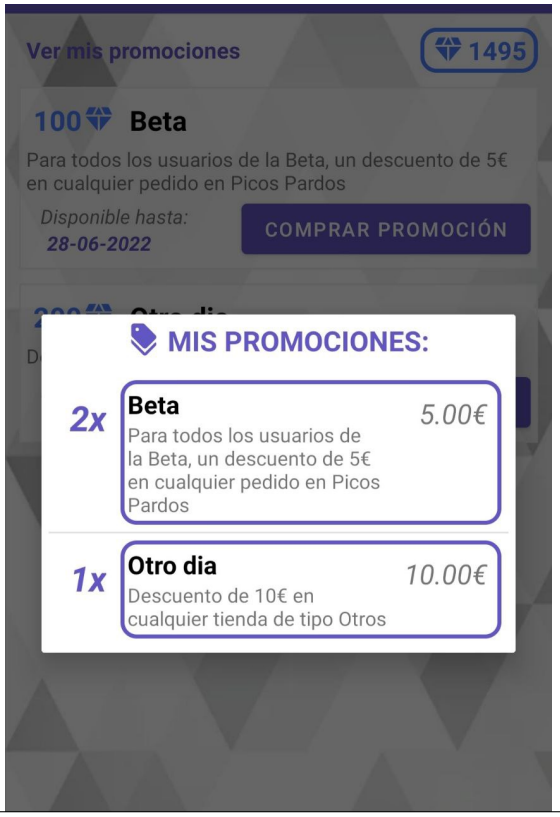
Descripción	Muestra la lista de promociones adquiridas por el usuario. Para cada una muestra cantidad, nombre, descuento y descripción.
Eventos	▪ Si se está seleccionando promoción y se pulsa sobre una, se asignará al pedido. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 64: Interfaz del dialog de mis promociones

Interfaz del dialog editar información personal

Descripción	Muestra un cuadro con campos de texto donde modificar la información personal (nombre, apellido, nombre de usuario, telefono y dirección)
Eventos	▪ Si pulsa guardar, se verifica que los campos sean válidos y se actualizan si lo son. ▪ Si no lo son, aparece una alerta al usuario. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 65: Interfaz del dialog editar información personal

Interfaz del dialog editar campo

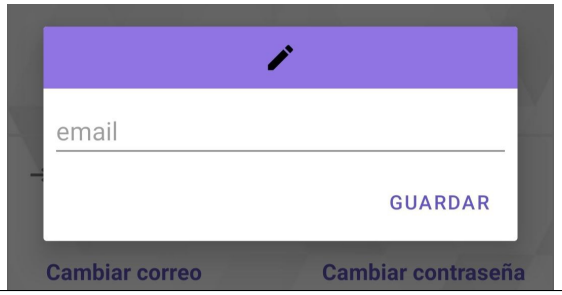
Descripción	Muestra un cuadro con un campo de texto donde modificar el dato que se quiera (se usa para muchos distintos).
Eventos	▪ Si pulsa guardar, se verifica que lo introducido sea válido y se actualiza si lo es. ▪ Si no lo es, aparece una alerta al usuario. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 66: Interfaz del dialog editar campo

Interfaz del dialog editar horario

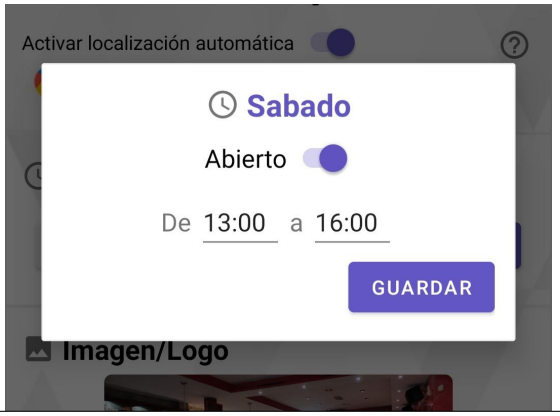
Descripción	Muestra un cuadro para modificar el horario de un día de la semana.
Eventos	▪ Si se cambia el selector de abierto, se muestra (o no) los campos de texto de las horas. ▪ Si pulsa guardar, se verifica que lo introducido sea válido y se actualiza si lo es. ▪ Si no lo es, aparece una alerta al usuario. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 67: Interfaz del dialog editar horario

Interfaz del dialog de modificar pedido

Descripción	Muestra información sobre el pedido que se está realizando, peromitiend editarlo.
Eventos	▪ Si se pulsa una flecha, cambia la cantidad de un producto del pedido. ▪ Si se pulsa la X se elimina el producto del pedido. ▪ Si se pulsa borrar ,se elimina la promoción asociada al pedido. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 68: Interfaz del dialog de modificar pedido

Interfaz del dialog de producto

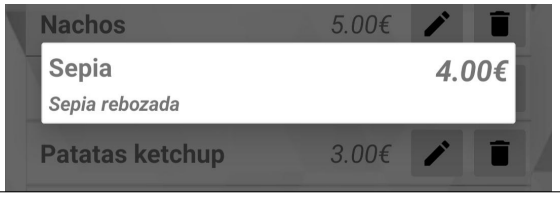
Descripción	Muestra información sobre el producto.
Eventos	▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 69: Interfaz del dialog de producto

Interfaz del dialog editar producto

Descripción	Muestra un cuadro con campos de texto para editar la información sobre el producto.
Eventos	▪ Si pulsa guardar, se verifica que lo introducido sea válido y se actualiza si lo es. ▪ Si no lo es, aparece una alerta al usuario. ▪ Si pulsa el botón del teléfono de volver atrás, se cerrará el dialog.
Captura/s	

Tabla 70: Interfaz del dialog editar producto

6. Implementación

6.1. Requerimientos Hardware y Software

En este apartado se van a exponer los requisitos necesarios para poder usar la aplicación de manera eficiente.

6.1.1. Software

- Sistema Operativo Android versión 5 o superior, aunque la versión recomendada es la 9 o superior.
- Otros : Google Maps (para localizar tiendas).

6.1.2. Hardware

Tras varias pruebas y análisis usando la herramienta interna de Android Studio para medir el consumo de la aplicación, se han aproximado los requisitos mínimos de software:

- RAM: 300 MB
- Velocidad CPU: 1.2 GHz.
- Compatibilidad CPU: No hay incompatibilidades de CPU.
- Espacio en dispositivo: 40 MB.
- Conexión a internet estable, ya sea por Wi-Fi o servicio de datos.
- Acceso a recursos de ubicación del teléfono.

6.2. Herramientas empleadas

6.2.1. Herramientas para el desarrollo de la aplicación

El desarrollo de la aplicación se ha realizado utilizando Android Studio.

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Se basa en la tecnología IntelliJ IDEA y consta de diferentes herramientas de desarrollo como:

- Editor libre de código. Android Studio soporta lenguajes de programación como Kotlin, C++ y Java (usado en este proyecto).
- Soporte para construcción basada en Gradle.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Soporte integrado para Google Cloud Platform, que permite la integración con Firebase.
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones.

6.2.2. Herramientas de soporte

Aparte de las herramientas ya mencionadas, se han usado las siguientes como soporte al proyecto:

- Overleaf - Es un editor laTeX colaborativo basado en la nube que se utiliza para escribir, editar y publicar documentos. Brinda la conveniencia de un editor LaTeX fácil de usar con colaboración en tiempo real y la salida totalmente compilada producida automáticamente en segundo plano a medida que escribe. En este proyecto se ha usado para escribir toda la memoria.
- Astah - Es una herramienta de modelado UML. En este proyecto se ha usado para modelar los diagramas usados en el proyecto, vistos durante esta memoria.
- Adobe Photoshop - Este programa es un editor de fotografías. Como su nombre lo indica, Photoshop "taller de fotos" es usado principalmente para el retoque de fotografías y gráficos. En este proyecto se ha usado como soporte al crear y modificar fondos e imágenes para la interfaz de la aplicación, así como el logotipo de la misma.
- GitHub - Es una plataforma la cual ofrece un servicio de almacenamiento de repositorios en la nube. En este proyecto se ha empleado para guardar las distintas versiones desarrolladas de nuestro sistema.

6.3. Tecnologías empleadas

- Java - Es un lenguaje de programación que se utiliza para desarrollar sistemas de software capaces de ejecutarse en distintas plataformas. En nuestro caso ha servido para implementar toda la lógica de negocio de la aplicación.
- JSON - JavaScript Object Notation (JSON) es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos. En nuestro caso se ha usado para pasar información necesaria de los objetos al cambiar de actividad, por medio de la serialización de estos objetos en formato JSON.
- XML - Es un es un lenguaje de marcado, el cual contiene una serie de reglas para codificar documentos que contienen información estructurada, y que así pueda ser compatible con los dispositivos electrónicos. En nuestro proyecto estos archivos representan las vistas que se van a mostrar al usuario.

6.4. Servidores de bases de datos utilizados

Como ya se ha mencionado en el apartado de Arquitectura Física, en este proyecto se ha usado una arquitectura cliente servidor, en la que se ha usado Firebase para gestionar este último.

Firebase es una plataforma de Google para el desarrollo de aplicaciones web y aplicaciones móviles lanzada en 2011. Esta ubicada en la nube, y usa un conjunto de herramientas para la creación y sincronización de proyectos.

Las principales ventajas de usar esta plataforma son:

- Sincronizar fácilmente los datos de sus proyectos sin tener que administrar conexiones o escribir lógica de sincronización compleja.
- Usa un conjunto de herramientas multiplataforma: se integra fácilmente para plataformas web como en aplicaciones móviles. Es compatible con grandes plataformas, como IOS, Android, aplicaciones web, Unity y C++.
- Usa la infraestructura de Google y escala automáticamente para cualquier tipo de aplicación, desde las más pequeñas hasta las más potentes.
- Crea proyectos sin necesidad de un servidor: Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.

En los siguientes apartados se van a explicar las diferentes herramientas utilizadas, así como la integración de Firebase en el proyecto.

6.4.1. Creación y configuración de Firebase

Antes de poder agregar Firebase al proyecto, se debes crear un proyecto de Firebase. A continuación se van a mostrar los pasos para crear un proyecto de Firebase:

1. En Firebase console, haz clic en Agregar proyecto.

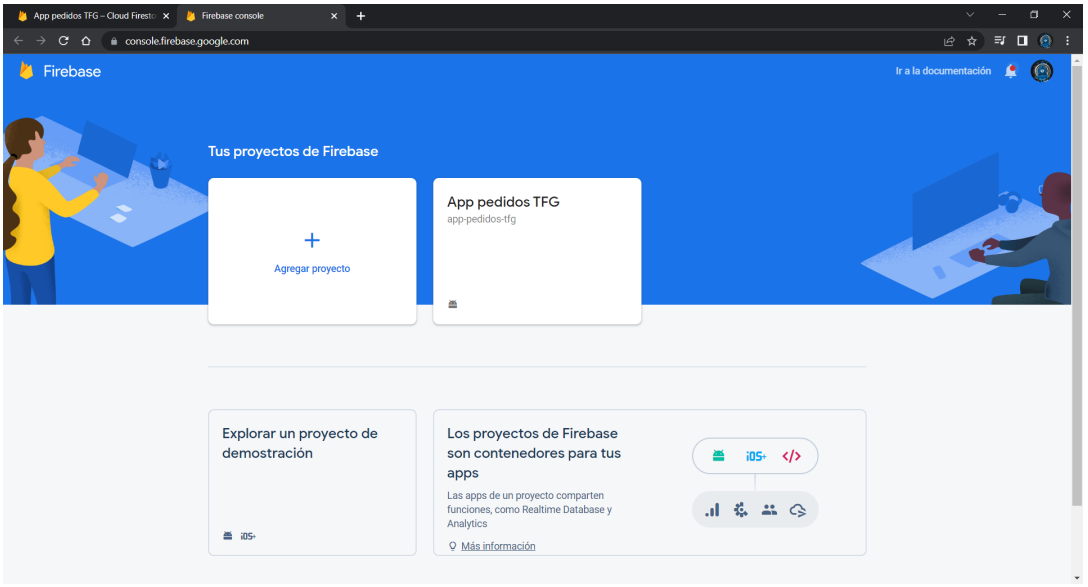


Imagen 11: Paso 1 de creación de proyecto de Firebase

2. Ingresa el nombre del proyecto, si se te solicita, revisa y acepta las Condiciones de Firebase y haz clic en Continuar.

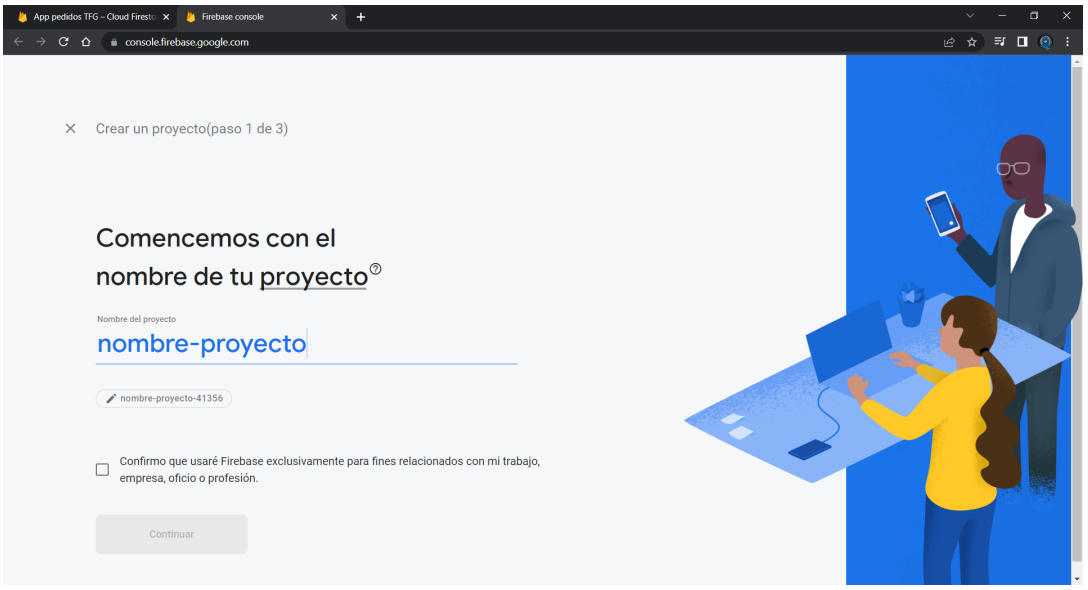


Imagen 12: Paso 2 de creación de proyecto de Firebase

3. Opcional: Configura Google Analytics para tu proyecto a fin de tener una experiencia óptima con sus productos.

   Luego, acepta la configuración de uso compartido de datos y las condiciones de Google Analytics para el proyecto.

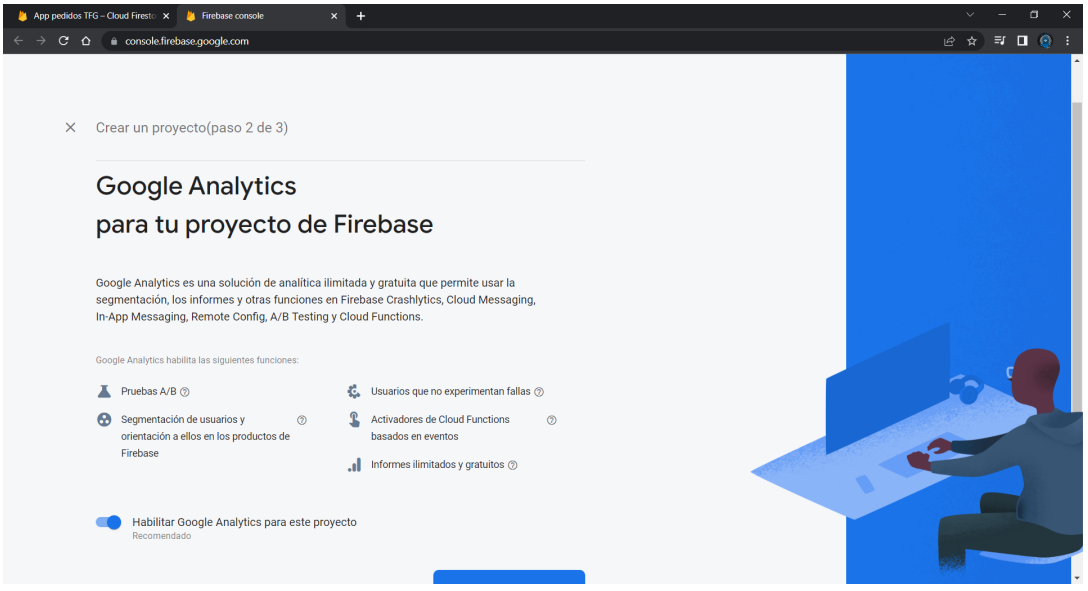


Imagen 13: Paso 3 de creación de proyecto de Firebase

4. Haz clic en Crear proyecto. Firebase aprovisiona los recursos para tu proyecto de forma automática. Cuando finalice, verás la página de descripción general del proyecto en Firebase console.

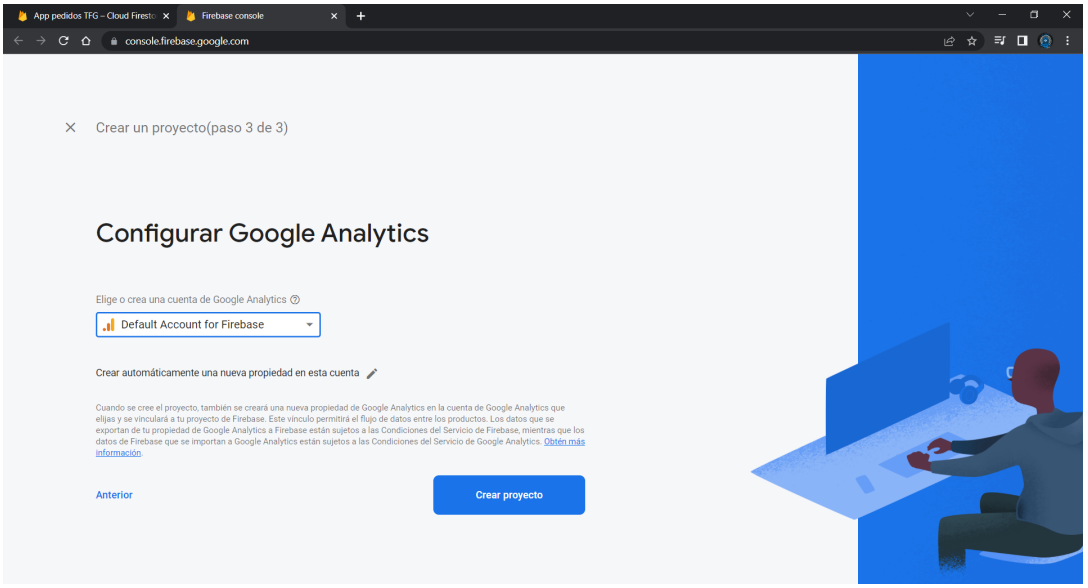


Imagen 14: Paso 4 de creación de proyecto de Firebase

Una vez creado el proyecto, el siguiente paso es registrar la app en el proyecto de Firebase. El registro de tu app a menudo se conoce como “agregar” la app a tu proyecto.

1. Dirígete a Firebase console.
2. En el centro de la página de descripción general del proyecto, haz clic en el ícono de Android o en Agregar app para iniciar el flujo de trabajo de configuración.

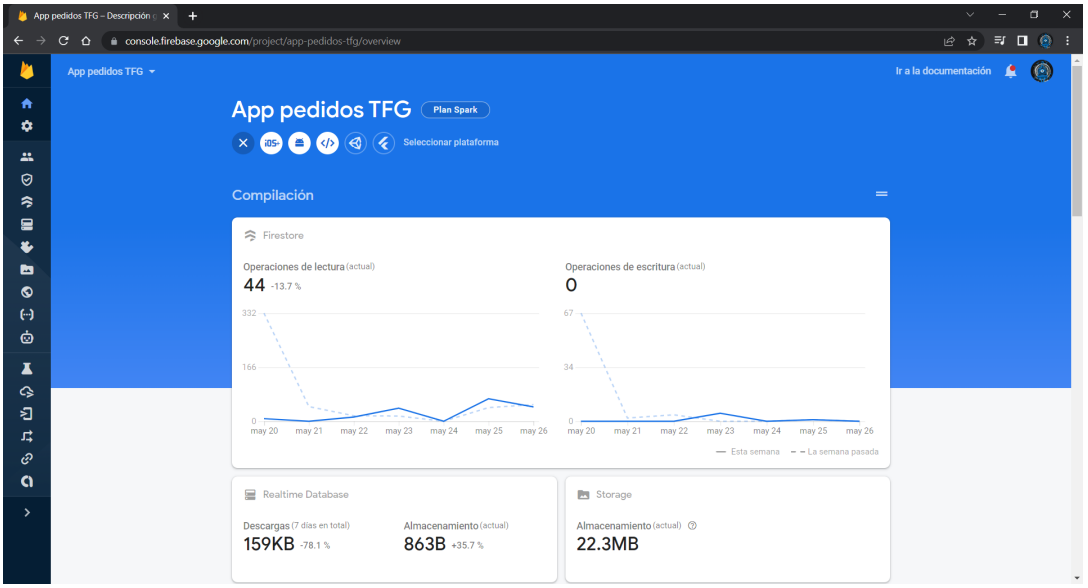


Imagen 15: Paso 2 de registro de app en Firebase

3. Ingresa el nombre del paquete de tu app en el campo “Nombre del paquete de Android´´, suele referirse al ID de aplicación. Este se encuentra en el archivo Gradle del proyecto.

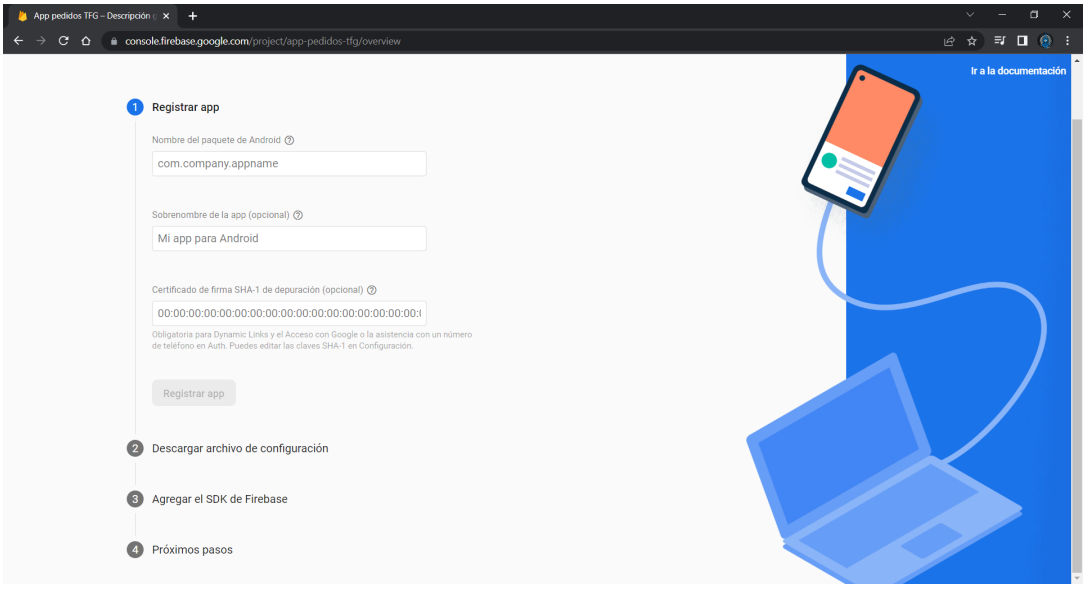


Imagen 16: Paso 3 de registro de app en Firebase

4. Haz clic en Registrar app.

Una vez está la app registrada, el siguiente paso es agregar un archivo de configuración de Firebase a la app y añadir los SDK.

1. Agrega el archivo de configuración de Firebase para Android a la app, para ello descarga *google-services.json* a fin de obtener el archivo de configuración de Firebase para Android y transfiere el archivo al directorio del módulo de la app.

2. Agrega el complemento de google-services, a tus archivos Gradle para de habilitar los productos de Firebase en tu app.

3. Declara las dependencias de los productos de Firebase que quieres usar en tu app.

```
implementation 'com.google.firebase:firebase-auth:21.0.1'
implementation 'com.google.firebase:firebase-firestore:24.0.1'
implementation 'com.google.firebase:firebase-storage:20.0.0'

implementation 'com.firebaseui:firebase-ui:0.6.0' //para el firebaseList (adapter del chat)
```

Imagen 17: Paso 3 de configuracion de Firebase en el proyecto

4. Sincroniza tu app para garantizar que todas las dependencias tengan las versiones necesarias.

6.4.2. Firebase Auth

Firebase Auth es un servicio que puede autenticar los usuarios utilizando únicamente código del lado del cliente. Incluye la autenticación mediante proveedores de inicio de sesión como Facebook, GitHub, Twitter, Google, Yahoo y Microsoft; así como los métodos clásicos de inicio de sesión mediante correo electrónico y contraseña. Además, incluye un sistema de administración del usuario por el cual los desarrolladores pueden habilitar la autenticación de usuarios con correo electrónico y contraseña que se almacenarán en Firebase.

Usamos este servicio para manejar todo el sistema de autenticación, ya que mejora la incorporación, acceso y seguridad para los usuarios respecto al desarrollar métodos de autenticación clásicos, ya que Firebase aporta de manera sencilla, eficaz y segura métodos para gestionar sus usuarios.

6.4.3. Realtime Database

Firebase proporciona una base de datos en tiempo real, back-end y organizada en forma de árbol JSON. El servicio proporciona a los desarrolladores de aplicaciones una API que permite que la información de las aplicaciones sea sincronizada y almacenada en la nube de Firebase. La compañía habilita integración con aplicaciones Android, iOS, JavaScript, Java, Objective-C, Swift y Node.js. La base de datos es también accesible a través de una REST API e integración para varios sistemas de Javascript como AngularJS, React, Ember.js y Backbone.js. La REST API utiliza el protocolo SSE (del inglés Server-Sent Events), el cual es una API para crear conexiones de HTTP para recibir notificaciones push de un servidor.

La sincronización en tiempo real de esta base de datos permite que los usuarios accedan a la información de sus datos desde cualquier dispositivo en tiempo real, compartiendo una instancia de Realtime Database, y cada vez que un usuario realice una modificación en esta, se almacena dicha información en la nube y se notifica simultáneamente al resto de dispositivos.

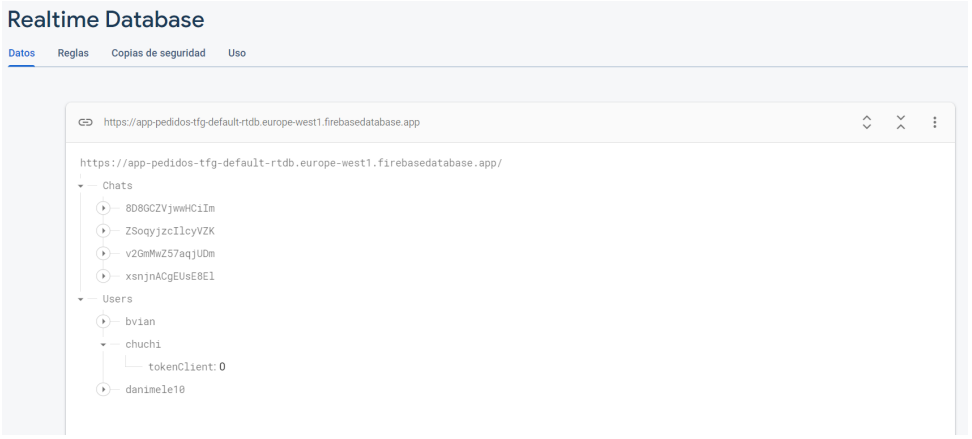


Imagen 18: Captura de realtime DB

En nuestro caso usamos esta funcionalidad de sincronización para administrar las notificaciones, cada usuario tiene un campo en la base de datos llamado *tokenClient* en el que se configura un "listener" donde se le envía información, como pueden ser detalles de un pedido cuando la tienda cambia el estado del mismo, para que se procese y muestre por el sistema de notificaciones (ver imagen [**19**]).

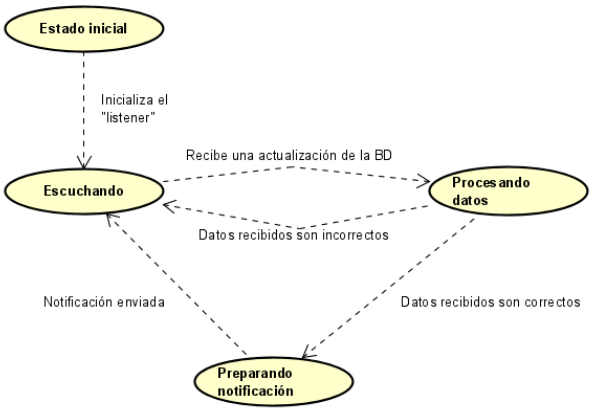


Imagen 19: Diagrama de estados configuracion de un listener

De un modo similar para notificar a un usuario de algún mensaje en alguno de sus pedidos, para cada pedido asignado al usuario se crea un oyente que reacciona cuando se escribe un mensaje.

6.4.4. Firebase Cloud Firestore

Cloud Firestore es un servicio de almacenamiento de datos derivado de Google Cloud Platform, adaptado a la plataforma de Firebase. Al igual que Realtime Database, es una base de datos NoSQL, aunque presenta diversas diferencias. Se organiza en forma de documentos agrupados en colecciones, y en ellos se pueden incluir tanto campos de diversos tipos (cadenas de texto, números, puntos geográficos, referencias a la propia base de datos, arrays, booleanos, marcas de tiempo, e incluso objetos propios) como otras subcolecciones, como se puede ver en [**20**].

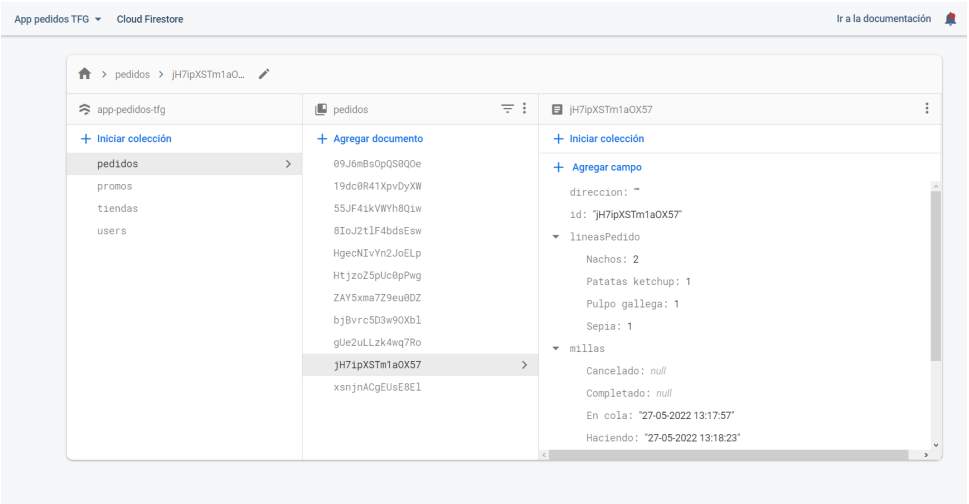


Imagen 20: Captura de Cloud Firestore

Esta servicio es el que usamos como base de datos principal, donde guardamos toda la información de usuarios, pedidos, tiendas, etc. definido en el apartado Diseño de la Base de Datos.

6.4.5. Firebase Storage

Firebase Storage proporciona cargas y descargas seguras de archivos para aplicaciones Firebase, sin importar la calidad de la red. En nuestro caso lo utilizamos para almacenar imágenes para el perfil de usuario y las tiendas, guardando en la Firestore y en el objeto el enlace como cadena de texto.

```
        Uri file = data.getData();                              //Uri de la foto

        guardarImagenPerfil(file);
    }
}

public void guardarImagenPerfil(Uri file) {
    StorageReference storageRef = storage.getReference();       //Referencia al almacenamiento

    HomeActivity act= (HomeActivity) this.getActivity();
    Usuario user=act.homePresenter.getUser();                   //Obtenemos los datos del usuario del presenter
    String email=user.getEmail();   //Usamos el email para identificar
    StorageReference userRef = storageRef.child("images/users/"+email);
    UploadTask uploadTask = userRef.putFile(file);              //Subir la foto

    uploadTask.continueWithTask(task -> {
        if (!task.isSuccessful()) {
            throw task.getException();
        }
        // Continue with the task to get the download URL
        return userRef.getDownloadUrl();
    }).addOnCompleteListener(task -> {  //On complete
        if (task.isSuccessful()) {
            Toast.makeText(this.getContext(),  text: "Foto de perfil actualizada con éxito.", Toast.LENGTH_SHORT).show();;   //Aviso por pantalla
            Uri downloadUri = task.getResult();

            //actualizar username en BD y en el usuario
            firestore.collection( collectionPath: "users").document(user.getEmail()).
                    update( field: "photo", String.valueOf(downloadUri));
            user.setPhoto(String.valueOf(downloadUri));

            //Actualizamos las fotos de la cabecera y el perfil
            act.loadHeader();
            loadPhoto();
        } else {
            // Handle failures
            Toast t = Toast.makeText(this.getContext(),  text: "Error al actualizar la foto de perfil.", Toast.LENGTH_SHORT);
            t.show();
        }
    });
}
```

Imagen 21: Código de ejemplo para guardar una foto de perfil en Firebase Storage

6.5. Consideraciones de Implementación

En este apartado se van a mostrar ciertas cuestiones clave sobre el funcionamiento del sistema.

6.5.1. Guardado de sesión. Archivo de preferencia

Para guardar la sesión de los usuarios, se guardan sus datos en un archivo por medio de las API de SharedPreferences. Un objeto SharedPreferences apunta a un archivo que proporciona métodos sencillos para leer y escribir donde se encuentra la información que queremos guardar. El framework administra cada archivo de SharedPreferences, que puede ser privado o compartido, en nuestro caso privado.

En el archivo guardamos la información de cada usuario en formato JSON, la cual de-serializamos por medio de la librería GSON. Con esto ya tenemos los datos del usuario, los cuales luego se comprueba que existan en la base de datos.

```
public Usuario loadSession(SharedPreferences prefs){
    Log.d( tag: "SA.Status", msg: "LOADING sesion");

    String userString = prefs.getString( s: "user",  s1: null);
    Gson gson = new Gson();
    Usuario user = gson.fromJson(
            userString,
            Usuario.class
    );
    return user;
}
```

Imagen 22: Código de lectura de sesión de usuario

El guardado de sesión se realiza automáticamente al iniciar la pantalla principal, es decir tras el inicio de sesión o registro del usuario. Para ello se escribe en el fichero la cadena “userString”, la cual contiene el objeto de usuario serializado en formato JSON.

```
SharedPreferences.Editor prefs = getSharedPreferences(
        "com.example.apppedidos.PREFERENCE_FILE_KEY"/*accedemos al fichero*/,
        Context.MODE_PRIVATE/*modo de acceso privado*/
).edit(); //edit para poner en modo de edicion a nuestro share preferences y añadir los datos
prefs.putString( s: "user", userString);
prefs.apply();
```

Imagen 23: Código de escritura de sesión de usuario

Si el usuario cierra o borra la cuenta, este fichero se “limpiará”, borrando todos los datos para no mantener la sesión abierta.

```
public void cerrarSesion() {
    //Borrado de datos
    SharedPreferences.Editor prefs = getSharedPreferences("com.example.apppedidos.PREFERENCE_FILE_KEY", Context.MODE_PRIVATE).edit();
    prefs.clear(); //Para borrar todas las preferencias que tenemos guardadas en la App
    prefs.apply();
```

Imagen 24: Código de cierre de sesión de usuario

6.5.2. Horario de negocios

Para controlar los horarios de negocios se ha usado un sistema basado en representar los horarios como cadenas de texto y luego procesar las fechas de forma interna. Esto se ha hecho para facilitar el procesar fechas al editar los datos.

La propia tienda convierte la cadena guardada en la base de datos a la lista de horarios [**25**]. Para cada tienda existe una lista de *Horario*, esta clase representa cada uno de los espacios de tiempo donde el negocio está abierto.

```
public void setHorarioFromStr(String horario) {
    ArrayList<Horario> list=new ArrayList<>();
    if(!horario.equals("")){
        String[] hDias=horario.split( regex: ",");
        for(String s:hDias){
            String[] values=s.split( regex: ";");
            Horario h=new Horario(values[0],values[1],Integer.parseInt(values[2]));
            list.add(h);
        }
    }
    this.horario=list;
    ordenaHorario();
}
```

Imagen 25: Código que procesa la cadena

Cada horario en la cadena tiene el formato “horaApertura” ;“horaCierre” ;“diaSemana” y se concatenan todos los horarios mediante comas. Por ejemplo la cadena “13:00;16:00;6,13:00;16:00;7” representa que el negocio está abierto de 13 a 16 sábados y domingos.

6.5.3. Sistema de promociones

Uno de los atractivos de la aplicación es el sistema de promociones, donde los clientes pueden obtener descuentos por usar la aplicación de forma seguida.

Para ello se recompensa al usuario por cada pedido completado con puntos, proporcionalmente al precio del pedido, los cuales se pueden acumular en la cuenta.

Estos puntos luego pueden ser usados para adquirir promociones a través de la pantalla de Interfaz de promociones, estas son descuentos especiales de tiempo limitado que funcionan en ciertas condiciones, por ejemplo puede ser canjeada exclusivamente en una tienda o estar disponible para un tipo concreto de tiendas (bar, alimentación, etc.). Cuando un usuario adquiere una de estas promociones, se le guarda un “ticket” de promoción, estos son acumulables y se pueden canjear cuando se quiera (estos no caducan) en una tienda siempre que se cumplan las condiciones de la promoción.

6.6. Organización Interna del Proyecto

En esta sección se describe la estructura organizativa interna del proyecto (en el entorno de programación), es decir la distribución de los archivos en sus respectivas carpetas, para facilitar la comprensión y localización de la implementación de cada una de las funcionalidades.

La estructura principal de la aplicación es la siguiente:

AppPedidos/app/src - Directorio principal del proyecto, dentro de este se encuentra:

- ../test y ../androidTest - Directorios con los archivos de pruebas
- ../main - Directorio con los recursos y código fuente. Este a su vez se puede dividir en:
  - ../AndroidManifest - El archivo de manifiesto describe información esencial de la aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play.
  - ../res - Directorio con todos los recursos usados por el proyecto, este se divide en varias carpetas creadas por el propio Android Studio, las usadas han sido:
    - ../layout - Contiene todos los archivos xml que definen los layout de las pantallas mostradas en la aplicación.
    - ../drawable - Contiene todas las imagenes e iconos usados en el proyecto. También existe una carpeta drawable-night la cual guarda versiones de imagenes usadas cuando el modo oscuro de la aplicación está activado.
    - ../menu - Contiene el xml usado para definir el menú de Interfaz del menú de la pantalla principal.
    - ../values - Contiene los archivos xml que definen los colores, los temas y las cadenas de texto predefinidas.
  - ../java/com/example/appPedidos - Contiene los archivos java encargados de la lógica. Esta a su vez contiene:
    - DBAccess y Utils - Archivos auxiliares usados en varios puntos de la lógica.
    - ../Actividades - Directorio con las vistas del patrón MVP. Dentro de este, aparte de los ficheros java de cada actividad, se encuentran:
      - ../Adapters - Contiene los ficheros java de los adaptadores usados en el proyecto.
      - ../DialogFragments - Contiene los ficheros java de los cuadros de dialogo usados en el proyecto.
      - ../HomeFragments - Contiene los ficheros java de los fragmentos del menú principal.
    - ../Modelo - Directorio con los archivos referentes al modelo del patrón MVP.
    - ../Presenters - Directorio con los archivos presentadores del patrón MVP.

7. Pruebas

7.1. Pruebas de caja blanca

El objetivo de este tipo de pruebas es validar las funciones internas de los módulos o subprogramas que se van a probar. El funcionamiento consiste en coger fragmentos de código y dividirlo en bloques, es decir cada bloque se trata como un proceso independiente y que se realiza de manera secuencial. Con esto se consigue rastrear todos los flujos de actuación que puede adoptar un programa para asegurarnos de que realiza las acciones de la forma esperada.

En nuestro proyecto las pruebas realizadas se han realizado durante el proceso de implementación, entre las pruebas realizadas se pueden destacar:

- Comprobación de las peticiones a la base de datos en tiempo de ejecución.
- Comprobación de todas las funcionalidades mostradas al usuario.
- Comprobación de carga de imágenes y archivos en tiempo de ejecución.
- Comprobación del funcionamiento del sistema en caso de que los datos a mostrar no se encuentren disponibles.
- Comprobación de la adecuación de las interfaces relativas a las diferentes funcionalidades.
- Comprobación de la adaptación de las interfaces a distintos dispositivos (diferentes resoluciones de pantalla y versiones).
- Comprobación de que los datos recuperados de la base de datos y los mostrados al usuario sean iguales.
- Comprobación de la conexión y comunicación del sistema con los agentes externos participantes del sistema (google maps).
- Comprobación del correcto funcionamiento de los archivos de preferencia de la aplicación almacenados en el propio dispositivo.

7.2. Pruebas de usuario

También durante el progreso del proyecto, sobre todo en etapas finales, se ha ido mostrando a usuarios la aplicación, para que pudieran opinar sobre distintas opciones de la app. En estas pruebas realizaban una serie de casos de uso y luego explicaban que se podría mejorar, normalmente referente a la interfaz de la aplicación. Algunas de estas mejoras fueron:

- Gif de carga en el inicio - En la Interfaz de inicio de sesión inicialmente se mostraban siempre los campos de texto. Tras enseñarlo se observó que, aunque estuviera cargando ya la siguiente pantalla, los usuarios seguían intentando introducir a mano el correo y la contraseña.

  Por ello se decidió que al iniciar con una sesión ya guardada no se mostrara estos campos de texto, si no un gif de carga durante los segundos en los que el sistema accede a la pantalla principal.

- Cambio de paleta - Durante las primeras iteraciones la paleta de colores era mucho más saturada, especialmente en botones. Tras enseñar la interfaz a varios usuarios, la mayoría comentaron esto, por lo que se decidió cambiar a colores menos saturados.

- Mejora en el catálogo - Una sugerencia que se dio al mostrar Interfaz de edición de catálogo de una tienda fue que se pudiera ordenar este, ya que para catálogos grandes sería difícil encontrar ciertos productos, también se sugirió añadir una descripción para cada producto aparte del nombre (no tenía).

- Simplificación de interfaz de compra - Tras varias pruebas con usuarios, se observó que muchos pulsaban el el cuadro de las tiendas en Interfaz de lista de tiendas, el cual no hacía nada (entonces no había un cuadro de dialogo), antes de llegar al botón de realizar pedido, ya que esta interfaz estaba muy sobrecargada.

  Para solucionar esto se decidió crear un cuadro de dialogo que diera “feedback” al pulsar en el cuadro, solucionando también la sobrecarga de estos cuadros, haciendo esta interfaz mejor. Aparte esto solucionó otro problema detectado, la carga de esta pantalla era bastante lenta por culpa de tener que cargar todas las imágenes, pero al mover estas a los cuadros de diálogo se solucionó ya que ahora se cargarían de forma individual.

Cabe destacar que se intentó realizar una fase de pruebas de la aplicación, donde usuarios reales podrían descargar la aplicación y realizar pedidos usando el negocio familiar como tienda a la que pedir, valorando la aplicación a través de un formulario externo.

Esta versión “Beta” de la aplicación no permitía crear tiendas a los usuarios, solo usar la existente por razones de seguridad y tampoco contaba con el sistema de promociones, ya que todavía no estaba implementado del todo. Lamentablemente no se han realizado pedidos en todo este periodo, por lo que no se ha podido obtener ninguna valoración real.

8. Instalación de la aplicación

A continuación se va a explicar como instalar la aplicación en cualquier teléfono que cumpla las características.

8.1. Instalación por APK

En este apartado se va a ver como instalar la aplicación a través de un apk. Un archivo con extensión .apk es un paquete para el sistema operativo Android.

Para ello primero hay que descargar el archivo, actualmente para ello hay un enlace de MEGA el cual se puede encontrar en Repositorio del proyecto y enlace de descarga.

Una vez descargado el archivo, hay que seguir los siguientes pasos:

1. Localizar el archivo - Normalmente los archivos se guardan en la carpeta de Descargas del teléfono, aunque puede estar en otra, en cualquier caso usando un explorador/gestor de archivos acceder a la carpeta donde esté el archivo.

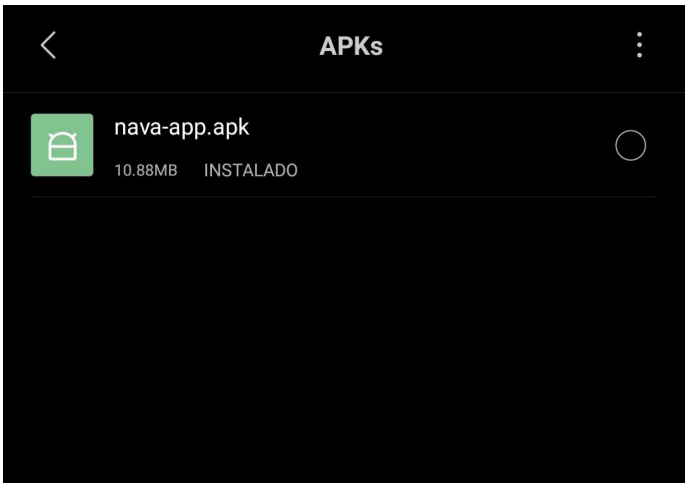


Imagen 26: Paso 1 instalación de APK

2. Seleccionar el archivo - Al pulsar el archivo saldrá un cuadro que preguntará si quieres instalar la aplicación.

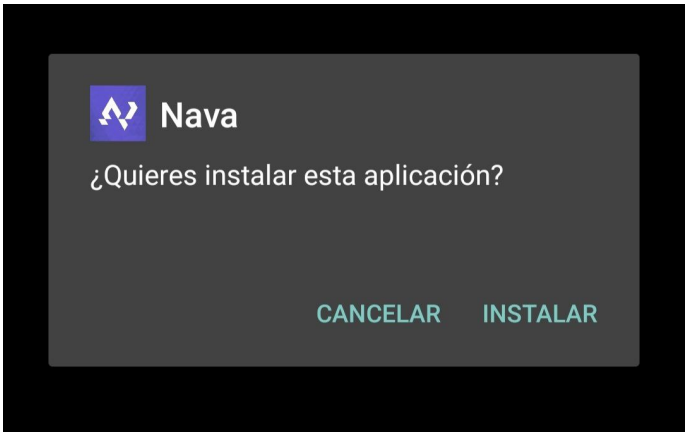


Imagen 27: Paso 2 instalación de APK

Puede ser que antes de dejar instalar la aplicación salga un cuadro de que no se permite instalar aplicaciones de fuentes desconocidas, si es así antes hay que permitir en los ajustes instalar aplicaciones de estas fuentes.

3. Instalación del archivo - Tras esto el archivo se ejecutará e instalará la aplicación en el sistema.

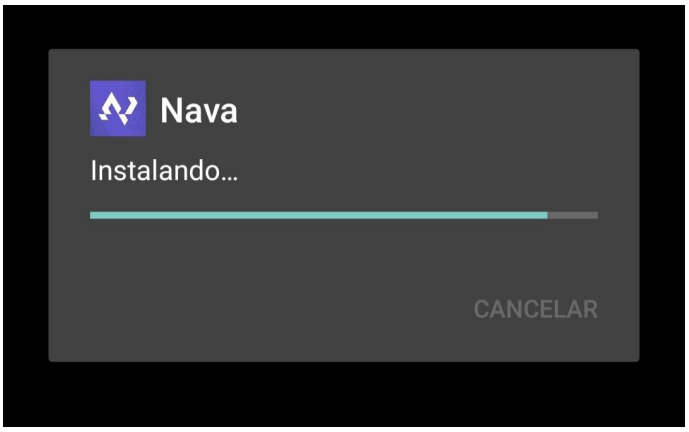


Imagen 28: Paso 3 instalación de APK

4. (opcional) Comprobación de la aplicación - Algunos dispositivos analizar la aplicación en busca de software malicioso por temas de seguridad al instalar la aplicación, una vez pasado este análisis ya se podría utilizar la aplicación.

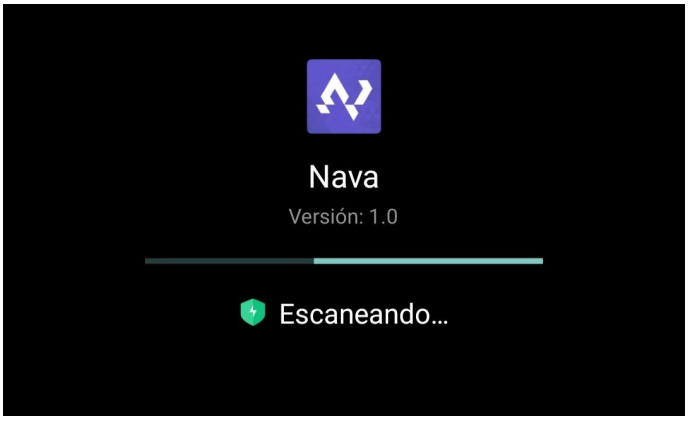


Imagen 29: Paso 4 instalación de APK

8.2. Instalación por Google Play

Aunque se pueda descargar por archivos apk, android tiene Google Play, una plataforma desde la que descargar aplicaciones de forma más sencilla.

Como ya se mencionó en Posibles costes futuros para poder subir aplicaciones a esta plataforma hay que pagar un registro, por lo que nuestra aplicación no está disponible.

Aun así, como es posible que se incluya esta forma de instalación en un futuro se va a incluir en este apartado.

1. Buscar la aplicación - Lo primero es encontrar la aplicación en el buscador del sistema, una vez encontrada la aplicación la pone en primer plano.

2. Instalar la aplicación - Tras esto se pulsa el botón de instalar y comienza la instalación automática de la aplicación.

3. (opcional) Comprobación de la aplicación - Aunque sea una fuente oficial, algunos dispositivos también analizan la aplicación, una vez pasado este análisis ya se podría utilizar la aplicación.

9. Posibles mejoras

A continuación se van a hablar de posibles mejoras de cara a un futuro en el caso de que el desarrollo de esta aplicación fuera más allá de este proyecto.

- Comprobación de tiendas - Una funcionalidad de la aplicación es que cualquier usuario registrado puede darse de alta como empresario y crear tiendas, aunque esto en principio es algo positivo, se podría usar de forma malintencionada.

  Para evitar estos malos usos, se podrían implementar 2 sistemas:

  - El primero sería un sistema de reporte, con este cualquier usuario podría detectar malos usos y reportar tiendas o usuarios. Esto requeriría de la participación de los propios usuarios de la aplicación, así como de alguien o algo que valorara estos reportes y actuara en base a ellos.
  - El segundo sería añadir algún tipo de verificación oficial, como podría ser pedir el NIF (número de identificación fiscal) de una empresa al darla de alta en el sistema y con ello comprobar la veracidad de esa empresa.

    Esto requeriría de una investigación sobre como obtener y verificar datos de estas empresas, así como actualizar el sistema para que permitiera esta verificación junto con los datos actuales que se guardan en el sistema.

- Pagos dentro de la aplicación - Como ya se comentó al comparar la aplicación con otras propuestas, se podría añadir un sistema de pagos desde la misma aplicación. Habría que investigar como implementar esto, tal vez mediante alguna API especifica para ello, la cual habría que valorar (en caso de que fuera de pago habría que añadir esto a los costes).

- Mejoras de la interfaz - Aunque la interfaz actual actual cumpla con lo necesario y es funcional, se podría mejorar añadiendo algunos elementos:

  - Distintos temas - Actualmente la aplicación cuenta con un solo tema (dos contando la versión oscura), una mejora podría ser permitir a los usuarios escoger entre varios temas, cambiando paletas de colores, fuentes de texto o fondos entre otras cosas.
  - Sonidos - Una mejora para el usuario podría ser añadir sonidos (podrían ser personalizados) a botones y menús para dar un mayor “feedback” al interactuar con ellos.
  - Animaciones - Otra mejora, en la linea con lo anterior, sería añadir animaciones en distintos puntos de la aplicación para mejorar la experiencia del usuario al usar la misma.

- Sistema de valoraciones - Como ya se ha mencionado en el apartado de Posibles costes futuros, se planteó implementar una API de Google capaz de obtener información de tiendas registradas en Maps para obtener las reseñas.

  Una opción alternativa a esto sería crear un sistema de valoraciones interno de la aplicación el cual permitiera valorar los aspectos de un pedido realizado (productos, tiempo, etc.) a un usuario, así como ver a otros las reseñas escritas para cada tienda.

- Realizar versiones WEB/IOS - Actualmente la aplicación solo funcionaría en Android, por lo que se podrían realizar en el futuro versiones tanto en una página web (para ordenadores) como para dispositivos con sistema IOS. Esto sería un proyecto completamente distinto y requeriría de bastante trabajo. Para ello habría que realizar clientes nuevos para estas versiones aunque se podría mantener el servidor actual de Firebase para todas las versiones.

10. Conclusiones

Primero hablando de lo “tecnico”, a pesar de haber trabajado ya en una aplicación móvil anteriormente en la asignatura de S. Móviles, este proyecto ha tenido un aprendizaje más grande de lo que creía en un principio, ya que he usado más elementos de trabajo del entorno de Android, los cuales eran más complejos de usar. Esto hizo que en varias ocasiones tuviera que parar el desarrollo, hacer pruebas para entender lo que quería añadir y finalmente aplicarlo al proyecto.

Algo que si ha sido completamente nuevo ha sido toda la parte de la base de datos, para la cual como ya se ha visto se han usado diferentes servicios de Firebase, y aunque también lo usamos en la aplicación de móviles, yo no me encargué de prácticamente nada, por lo que para este proyecto me ha tocado aprender a usarlo.

A nivel más personal, si he aprendido algo durante este proyecto ha sido a organizarme, desde siempre al hacer un trabajo de lo que fuera he sido de apurar las fechas, pero al tener tanto tiempo sin una fecha fija al iniciar el proyecto decidí organizarme para cumplir objetivos, lo cual derivó en la planificación por incrementos ya mencionada. Más o menos he sabido llevar esta planificación más o menos al día, lo cual me parece un logro a nivel personal.

11. Referencias

En primer lugar, quiero mencionar la asignatura de Sistemas Móviles. Al haber sido el proyecto una aplicación móvil, lo dado en esta asignatura ha sido de gran importancia para el desarrollo, sobre todo la práctica de la asignatura, la cual ha servido como base en partes de la creación de la aplicación.

11.1. Bibliografía

En este apartado se van a enumerar referencias obtenidas de libros y otros artículos, incluyendo apuntes de otras asignaturas usados durante el proyecto.

- Bob Hughes & Mike Cotterell - Software Project Management (5th edition)
- Arlow, Jim, Neustadt, Ila - UML 2
- Miguel A. Laguna - Apuntes de la asignatura *Modelado de Sistemas Sotfware*
- Yania Crespo - Apuntes de la asignatura *Diseño de Software*

11.2. Webgrafía

Aparte de la información obtenida ya mencionada, se han consultado diversas páginas web durante el desarrollo de la aplicación.

A continuación se van a enumerar estos artículos de información consultados. Primero se van a mostrar paginas web con documentación usada en múltiples ocasiones durante el proyecto, luego se van a mostrar enlaces con dudas concretas que se han considerado importantes enseñar.

11.2.1. Webs principales

- Documentación oficial de android studio: https://developer.android.com/studio
- Documentación oficial de uso de Firebase en entorno android: https://firebase.google.com/docs/android

11.2.2. Enlaces importantes

- Modelo vista presentador: https://www.develapps.com/es/noticias/modelo-vista-presentador-mvp-en-android
- Consultas firebase: https://stackoverflow.com/questions/38965731/how-to-get-all-childs-data-in-firebase-database
- Horario de tiendas (usado como ejemplo): https://github.com/dhatim/business-hours-java
- Navigation drawer:
  - https://guides.codepath.com/android/fragment-navigation-drawer
  - https://stackoverflow.com/questions/9853430/refresh-fragment-when-dialogfragment-is-dismissed/64479938#64479938

- Custom adapters:
  - https://stackoverflow.com/questions/8166497/custom-adapter-for-list-view
  - https://stackoverflow.com/questions/2618272/custom-listview-adapter-getview-method-being-called-multiple-times-and-in-no-co
- Comparadores - usado al ordenar listas: https://stackoverflow.com/questions/32995559/reverse-a-comparator-in-java-8
- Notificaciones: https://stackoverflow.com/questions/58817214/how-to-make-notification-appear-on-the-phone-for-android-studio
- Ajustar filtros: https://stackoverflow.com/questions/2505207/how-to-add-item-to-spinners-arrayadapter
- Ubicación:
  - https://android-doc.github.io/training/location/display-address.html
  - https://stackoverflow.com/questions/6205827/how-to-open-standard-google-map-application-from-my-application
- Chat:
  - https://code.tutsplus.com/es/tutorials/how-to-create-an-android-chat-app-using-firebase--cms-27397
  - https://stackoverflow.com/questions/4638832/how-to-programmatically-set-the-layout-align-parent-right-attribute-of-a-button
- Analizar requerimientos hardware: https://stackoverflow.com/questions/15358025/how-to-get-the-minimum-hardware-requirements-for-an-android-application

11.3. Repositorio del proyecto y enlace de descarga

Repositorio del proyecto: https://gitlab.inf.uva.es/danmele/tfg-danmele
Enlace de descarga: https://mega.nz/file/pBxwFLxL#4hZu_fWAbQ_aCw-Q-2nWS8frjPokPMfeOzcuzNmg6Zg