UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería del Software

Plataforma para Digitalización de Cartas y División de Gastos en Restaurantes

Platform for Menu Digitalization and Bill Splitting in Restaurants

Realizado por
Alba Ruiz Gutiérrez

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
Lenguajes y ciencias de la computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, septiembre 2025

Resumen

En la actualidad, los restaurantes buscan optimizar la experiencia de sus clientes mediante herramientas digitales que agilicen la gestión de pedidos y pagos. Uno de los problemas más comunes en reuniones de grupo es la dificultad para dividir la cuenta de manera justa, especialmente cuando cada persona ha consumido platos distintos o se han compartido ciertos ítems del menú. La falta de un sistema eficiente para realizar esta tarea genera confusión, retrasos en el pago y, en algunos casos, malentendidos tanto entre los comensales como con el restaurante.

Además, muchos restaurantes aún dependen de cartas físicas o menús en formatos poco accesibles, lo que conlleva ciertos inconvenientes. Entre ellos, destacan los riesgos de contagio en contextos sanitarios como los vividos durante la pandemia, la pérdida de tiempo mientras el camarero lleva la carta a la mesa y toma nota manualmente, y la dificultad para mantener los menús actualizados. Por otro lado, una carta digital no solo soluciona estos problemas, sino que también permite incluir imágenes de los platos y ofrecer información detallada sobre alérgenos e ingredientes, mejorando la experiencia del cliente y facilitando la toma de decisiones al pedir.

Este Trabajo de Fin de Grado propone el desarrollo de una plataforma que permita a los restaurantes digitalizar sus menús mediante escaneo OCR (Reconocimiento Óptico de Caracteres), generando una versión interactiva accesible desde una aplicación móvil. A partir de esta digitalización, los clientes podrán seleccionar los platos consumidos dentro de un grupo y dividir automáticamente el coste de la cuenta según distintos criterios, optimizando así la gestión de pagos en reuniones.

A diferencia de otras soluciones existentes que se centran únicamente en la división de gastos (como Splitwise o Tricount), esta propuesta integra la digitalización previa del menú como un paso esencial para mejorar la precisión y usabilidad del proceso. De esta manera, el proyecto no solo facilita la experiencia del usuario en el momento del pago, sino que también brinda a los restaurantes una herramienta para modernizar su servicio.

Palabras clave: OCR, React Native, Spring Boot, Digitalización de carta, División de gastos.

Abstract

Nowadays, restaurants seek to optimize their customers' experience through digital tools that streamline the management of orders and payments. One of the most common problems in group gatherings is the difficulty of fairly splitting the bill, especially when each person has ordered different dishes or certain menu items have been shared. The lack of an efficient system for this task often leads to confusion, payment delays, and, in some cases, misunderstandings among diners or with the restaurant itself.

In addition, many restaurants still rely on physical menus or poorly accessible formats, which come with several drawbacks. These include health risks in contexts such as the recent pandemic, wasted time while the waiter brings the menu to the table and takes notes manually, and difficulty keeping menus up to date. On the other hand, a digital menu not only solves these problems but also allows for the inclusion of images and detailed information about allergens and ingredients—enhancing the customer experience and helping them make informed decisions when ordering.

This Final Degree Project proposes the development of a platform that allows restaurants to digitize their menus through OCR (Optical Character Recognition) scanning, generating an interactive version accessible from a mobile application. Based on this digitalization, customers will be able to select the dishes consumed within a group and automatically split the bill according to different criteria, thereby optimizing payment management in group settings.

Unlike other existing solutions that focus solely on expense splitting (such as Splitwise or Tricount), this proposal integrates prior menu digitalization as an essential step to improve both the accuracy and usability of the process. In this way, the project not only simplifies the user experience during payment but also provides restaurants with a tool to modernize their service.

Keywords: OCR, React Native, Spring Boot, Menu Digitalizacion, Bill Splitting.

Índice

1

Introducción

1.1 Motivación

La motivación de este Trabajo Fin de Grado se sitúa en el contexto de la pandemia de la COVID-19, un momento que obligó al cierre temporal de bares y restaurantes. Cuando estos establecimientos pudieron reabrir sus puertas, lo hicieron bajo estrictas medidas de seguridad para reducir los riesgos de contagio. Una de las más visibles fue la sustitución de las cartas físicas por cartas digitales, habitualmente accesibles a través de códigos QR.

Aunque esta medida se adoptó de manera generalizada, pronto quedó en evidencia que no todos los restaurantes podían gestionarla con la misma facilidad. Muchos carecían de los recursos o las competencias digitales necesarias para diseñar una página web, estructurar un menú digital atractivo o mantenerlo actualizado. En consecuencia, se extendieron soluciones improvisadas: menús subidos como simples imágenes, documentos PDF poco legibles o enlaces difíciles de consultar. Estas limitaciones afectaron tanto a la experiencia de los clientes como a la imagen de los propios establecimientos.

Ante esta situación surge la idea de **CartaGo**, una aplicación pensada para ofrecer a los restaurantes una herramienta rápida y sencilla con la que digitalizar sus cartas físicas, sin necesidad de conocimientos técnicos en programación o diseño. Con este sistema, cualquier establecimiento puede transformar su menú en un formato digital claro y accesible, facilitando tanto su gestión como su consulta por parte de los clientes.

El proyecto, además, no se limita a resolver este problema inicial. También integra una funcionalidad orientada a mejorar la experiencia de los comensales: la división de gastos en grupo. Gracias a esta opción, los clientes pueden repartir la cuenta de forma equitativa y obtener instrucciones de pago claras, simplificando procesos habituales como realizar transferencias. De este modo, se combinan en una única herramienta dos necesidades cotidianas: la digitalización de menús y la gestión de pagos compartidos.

La motivación personal detrás de este trabajo parte de haber vivido de primera mano este cambio en la hostelería. Lo que en un principio fue una respuesta urgente a una crisis sanitaria se convierte aquí en la oportunidad de proponer una solución tecnológica más sólida, útil y adaptable. Con CartaGo se pretende facilitar a los restaurantes un acceso sencillo a la digitalización y, al mismo tiempo, mejorar la experiencia de los clientes en un entorno tan cotidiano como salir a comer en compañía.

1.2 Objetivos del proyecto

El desarrollo de este Trabajo Fin de Grado tiene como objetivo principal la creación de una aplicación móvil que facilite a los restaurantes la digitalización de sus cartas físicas y que, al mismo tiempo, ofrezca a los clientes funcionalidades adicionales que mejoren su experiencia, como la división de gastos en grupo y la obtención de instrucciones de pago.
La finalidad es aportar una solución práctica y accesible que responda tanto a las dificultades de digitalización de muchos establecimientos como a las necesidades cotidianas de los comensales.

Para lograr este objetivo general, se establecen los siguientes subobjetivos:

- Realizar un estudio de las aplicaciones existentes en el ámbito de la hostelería, identificando sus principales funcionalidades y carencias, con el fin de definir los elementos imprescindibles que debe incluir la aplicación.
- Investigar y seleccionar las tecnologías más adecuadas para el desarrollo de la aplicación móvil y del backend, teniendo en cuenta aspectos como la escalabilidad, la seguridad y la facilidad de uso.
- Diseñar un sistema que permita **digitalizar una carta física** de forma rápida y accesible, garantizando que los restaurantes puedan gestionar sus menús sin necesidad de conocimientos técnicos avanzados.
- Implementar un módulo de **gestión de gastos compartidos**, que permita a los clientes dividir la cuenta de manera clara y justa, generando además instrucciones de pago para facilitar procesos posteriores como transferencias.
- Establecer un modelo de almacenamiento de datos adecuado, que contemple la información de los restaurantes, las cartas digitalizadas y los registros de las divisiones de gastos, respetando criterios de seguridad y coherencia.
- Desarrollar una **interfaz de usuario sencilla e intuitiva**, que facilite tanto la labor de los restaurantes al subir o actualizar sus cartas como la experiencia de los clientes al consultarlas y utilizar las herramientas de pago.
- Realizar pruebas de funcionamiento y usabilidad para validar que la aplicación cumple con los objetivos propuestos, ofreciendo un servicio estable y una experiencia fluida.

1.3 Antecedentes

En esta sección se analizan algunas de las aplicaciones existentes en el mercado que guardan relación con las funcionalidades que se pretenden desarrollar en *CartaGo*. El objetivo es comprender cómo estas soluciones abordan necesidades similares y, a partir

de ahí, identificar las oportunidades de mejora y diferenciación que ofrece el presente proyecto.

Tricount [1]

- **Descripción:** Tricount es una aplicación enfocada en la gestión de gastos compartidos. Permite registrar gastos en grupos de amigos, viajes o actividades comunes, y calcular automáticamente quién debe pagar a quién para equilibrar las cuentas. Es muy popular para viajes o planes de ocio colectivos.
- **Características clave:** Creación de grupos de gasto, registro colaborativo de transacciones, cálculo automático de saldos pendientes, generación de resúmenes y enlaces compartibles.
- **Limitaciones respecto a *CartaGo*:** Aunque es una solución consolidada en el ámbito de dividir gastos, Tricount no está específicamente adaptada al contexto de la hostelería. No está integrada con cartas de restaurantes ni con procesos de consumo en tiempo real, por lo que si se quiera usar para dividir una cuenta grupal debería añadirse cada plato manualmente como un gasto individual. *CartaGo* toma como referencia este tipo de funcionalidades, pero las adapta al momento concreto de compartir una comida en un bar o restaurante, integrando el gasto directamente con los platos consumidos.

My Waiter [2]

- **Descripción:** My Waiter es una aplicación orientada al sector hostelero que permite a los clientes consultar menús digitales, realizar pedidos y gestionar la interacción con el personal de servicio desde su dispositivo móvil.
- **Características clave:** Visualización de la carta digital, pedidos desde la mesa, integración con la cocina y reducción del contacto físico con cartas tradicionales.
- **Limitaciones respecto a *CartaGo*:** Se centra principalmente en digitalizar la experiencia de pedidos dentro del restaurante, pero no contempla funcionalidades relacionadas con la división de la cuenta entre comensales ni con instrucciones de pago. *CartaGo* añade un valor diferencial al unir la digitalización de cartas con la gestión de gastos, ofreciendo así un servicio más completo para los clientes.

Uber Eats [3]

- **Descripción:** Uber Eats es una de las plataformas de reparto de comida a domicilio más utilizadas a nivel mundial. Permite al usuario consultar los menús de distintos restaurantes, realizar pedidos en línea y recibirlos en casa gracias a una red logística propia.
- **Características clave:** Amplia red de restaurantes asociados, sistema de pago integrado, promociones frecuentes, interfaz sencilla para explorar y pedir.
- **Limitaciones respecto a *CartaGo*:** Su modelo de negocio está orientado a la entrega a domicilio, no al consumo dentro del local. Además, impone

comisiones elevadas a los restaurantes, lo que supone una barrera de entrada para pequeños negocios. *CartaGo*, en cambio, está pensado para apoyar directamente al restaurante en su operativa diaria sin costes elevados, y para mejorar la experiencia de los clientes que acuden en persona.

TheFork [4]

- **Descripción:** TheFork (antes ElTenedor) es una plataforma de reservas de restaurantes. Permite a los usuarios descubrir nuevos locales, consultar menús, leer opiniones y reservar mesas en línea obteniendo descuentos adicionales y puntos canjeables en la aplicación.
- **Características clave:** Sistema de reservas online, descuentos y promociones, integración con reseñas de clientes, visibilidad para los restaurantes.
- **Limitaciones respecto a *CartaGo*:** Aunque mejora la gestión de reservas y la visibilidad de los restaurantes, TheFork no resuelve la digitalización de cartas ni la división de gastos. *CartaGo* se sitúa en otro momento de la experiencia del cliente: la interacción directa durante la comida y la gestión posterior de la cuenta.

Otras aplicaciones relevantes

Además de las anteriores, existen otras aplicaciones que aportan soluciones parciales:

- **Glovo [5]/ Just Eat[6]:** centradas en el reparto de comida a domicilio, con las mismas limitaciones que Uber Eats en cuanto a digitalización de cartas físicas y división de gastos.
- **Square POS[7] / Toast [8]/ TouchBistro[9]:** sistemas de punto de venta (POS) muy potentes para la gestión integral del restaurante, pero requieren infraestructura tecnológica avanzada y suelen suponer un coste elevado para pequeños establecimientos.

Aportación diferencial de *CartaGo*

De la comparación realizada se puede concluir que las aplicaciones existentes suelen estar especializadas en un ámbito concreto: reservas (TheFork), pedidos a domicilio (Uber Eats, Just Eat, Glovo), digitalización de pedidos en sala (My Waiter) o división de gastos genérica (Tricount). Sin embargo, no existe una solución que combine de forma sencilla y accesible la **digitalización de cartas físicas** con la **gestión de gastos compartidos** en un mismo entorno.

La novedad de *CartaGo* reside en:

- Proporcionar a los restaurantes una herramienta fácil de usar para digitalizar sus menús sin necesidad de conocimientos técnicos.
- Mejorar la experiencia de los clientes no solo en el acceso a la carta, sino también en el **momento crítico de pagar y dividir la cuenta**.
- Ofrecer una solución ligera, sin requerimientos de hardware específico ni comisiones elevadas, pensada especialmente para pequeños y medianos negocios que no disponen de recursos para invertir en plataformas más complejas.

En este sentido, *CartaGo* se posiciona como una propuesta innovadora que responde a las necesidades tanto de los establecimientos como de los clientes, diferenciándose de las aplicaciones competidoras actuales al cubrir de manera integrada dos problemas habituales en la hostelería moderna: **la digitalización de menús y la gestión de pagos en grupo**.

1.4 Estructura de la memoria

Este documento se organiza en ocho capítulos que recogen de manera ordenada todas las fases del desarrollo del proyecto, desde la motivación inicial hasta las conclusiones y posibles líneas futuras de mejora.

El **Capítulo 1** presenta la introducción al proyecto. En él se expone la motivación que da origen a la propuesta, se definen los objetivos planteados y se realiza un análisis de antecedentes y aplicaciones competidoras. El capítulo concluye con la explicación de la estructura general de la memoria.

En el **Capítulo 2** se describen las tecnologías, librerías y herramientas utilizadas durante el desarrollo. Se detallan tanto las tecnologías principales empleadas en el frontend y en el backend, como las librerías de apoyo y las herramientas de diseño, pruebas y despliegue.

El **Capítulo 3** recoge la especificación y análisis del sistema. En él se identifican los actores principales, se definen los requisitos funcionales y no funcionales, y se presentan los casos de uso asociados a cada tipo de usuario.

En el **Capítulo 4** se aborda el diseño del sistema. Se explica la arquitectura propuesta, se muestra el prototipada inicial realizado en Figma y se incluyen los modelos conceptuales, tales como el diagrama entidad-relación, los diagramas de casos de uso, los diagramas de procesos y de secuencias, que permiten comprender la interacción entre los distintos componentes del sistema.

El **Capítulo 5** está dedicado a la implementación del backend. Se detalla la creación y configuración inicial de la base de datos y del servidor backend, así como la estructura final del mismo. También se incluyen las APIs externas integradas en el proyecto, como el servicio de geocodificación de Nominatim o la gestión de imágenes mediante Cloudinary.

En el **Capítulo 6** se desarrolla la implementación del frontend. Se explica la estructura del proyecto y se describe la interfaz de usuario, diferenciando entre la vista correspondiente al restaurante y la del cliente.

El **Capítulo 7** trata sobre el despliegue y las pruebas del sistema. Se documenta el proceso de despliegue del frontend, backend y base de datos, además de presentar los principales casos de prueba para garantizar el correcto funcionamiento de la aplicación.

Finalmente, el **Capítulo 8** incluye las conclusiones del trabajo. Se revisan los objetivos alcanzados, las dificultades encontradas durante el desarrollo y se proponen posibles ampliaciones o mejoras para versiones futuras de la aplicación.

La memoria se complementa con los apartados de **referencias bibliográficas**, así como con dos apéndices: el **Manual de Usuario**, que sirve de guía práctica para el uso de la aplicación, y el **Manual de Instalación**, que describe el proceso necesario para la puesta en marcha del sistema.

2

Tecnologías, librerías y herramientas utilizadas

2.1 Tecnologías utilizadas

A continuación, se presentan las tecnologías utilizadas para desarrollar el proyecto:

2.1.1 React Native

Ilustración 1. Logo de React Native

React Native es un marco de desarrollo de código abierto que permite desarrollar aplicaciones móviles multiplataforma con una única base de código. Basado en JavaScript y React, facilita la creación de aplicaciónes tanto para iOS como para Android, utilizando componentes que se traducen directamente en elementos de interfaz nativos. Esto permite ofrecer una experiencia de usuario fluida y coherente en ambos sistemas operativos.

2.1.2 Expo

Ilustración 2. Logo de Expo

Expo es una plataforma de código abierto que facilita el desarrollo de aplicaciones móviles con React Native. Ofrece un conjunto de herramientas y servicios que permiten crear, compilar y probar aplicaciones de forma sencilla, sin necesidad de configurar entornos nativos. Con Expo, los desarrolladores pueden centrarse en escribir código en JavaScript y React, utilizando componentes listos para usar y compatibles tanto con iOS como con Android.

2.1.3 Java

Ilustración 3. Logo de Java

Java es un lenguaje de programación orientado a objetos y multiplataforma. Gracias a la Máquina Virtual de Java (JVM), permite ejecutar el mismo código en distintos sistemas. Es ampliamente usado en aplicaciones móviles, web y empresariales.

2.1.4 Spring Boot

Ilustración 4. Logo de Spring Boot

Spring Boot es un framework de Java que facilita el desarrollo de aplicaciones backend modernas. Proporciona una conIlustracióncIón automática y un entorno listo para ejecutar, lo que permite crear servicios web y APIs de forma rápida y eficiente. Es ideal para construir aplicaciones escalables, modulares y fáciles de mantener.

2.1.5 Git

Ilustración 5. Logo de GIT

Git es un sistema de control de versiones distribuido que permite gestionar y registrar los cambios en el código fuente de un proyecto. Es ampliamente utilizado en desarrollo de software para trabajar en equipo, mantener un historial de versiones y facilitar la colaboración a través de ramas y fusiones.

2.1.6 MySQL

Ilustración 6. Logo de MySQL

MySQL es un sistema de gestión de bases de datos relacional de código abierto. Se utiliza para almacenar, organizar y acceder a datos de forma estructurada mediante el lenguaje SQL. Es una de las bases de datos más populares en aplicaciones web y sistemas empresariales.

2.1.7 Figma

Ilustración 7. Logo de Figma

Figma es una herramienta de diseño y prototipado colaborativo basada en la nube. Permite crear interfaces, maquetas y prototipos interactivos en tiempo real, facilitando el trabajo en equipo entre diseñadores, desarrolladores y otros perfiles sin necesidad de instalar software.

2.2 Herramientas utilizadas

Ilustración 8. Logo de MySQL Workbench

2.2.1 MySQL Workbench

MySQL Workbench es una herramienta visual para el diseño, desarrollo y administración de bases de datos MySQL. Permite crear y modificar esquemas, ejecutar consultas SQL, gestionar usuarios y visualizar modelos de datos de forma intuitiva.

Ilustración 9. Logo de Docker Desktop

2.2.2 Docker Desktop

Docker Desktop es una aplicación que permite crear y gestionar contenedores de forma sencilla desde un entorno gráfico. Facilita el despliegue y la ejecución de aplicaciones en entornos aislados, garantizando la portabilidad y consistencia entre distintos sistemas. Es especialmente útil en el desarrollo y pruebas, ya que permite simular entornos de producción en local.

Ilustración 10. Logo de Postman

2.2.3 Postman

Postman es una herramienta diseñada para el desarrollo, prueba y documentación de APIs. Permite enviar peticiones HTTP, organizar colecciones de pruebas y automatizar escenarios de validación, lo que la convierte en un recurso esencial para comprobar el correcto funcionamiento de los endpoints de un backend.

Ilustración 11. Logo de IntelliJ Idea Comunity Edition

2.2.4 IntelliJ Idea Comunity Edition

IntelliJ IDEA Community Edition es un entorno de desarrollo integrado (IDE) especializado en proyectos Java. Ofrece funcionalidades como autocompletado de código, refactorización, integración con sistemas de control de versiones y soporte para frameworks como Spring Boot, facilitando así el proceso de programación y depuración.

Ilustración 12. Logo de Visual Studio Code

2.2.5 Visual Studio Code

Visual Studio Code es un editor de código multiplataforma ligero y flexible. Gracias a su amplia colección de extensiones, se adapta a distintos lenguajes y entornos de desarrollo. Es especialmente útil en proyectos frontend con tecnologías como JavaScript, TypeScript o React, e integra funciones de depuración, control de versiones y terminal en un solo entorno.

2.3 Librerías utilizadas

2.3.1 Backend

- Spring Boot Starter Web
  - Propósito: Crear APIs REST y gestionar peticiones HTTP.
  - Uso en el proyecto: Permite exponer los endpoints del backend y procesar las peticiones desde la aplicación móvil.
- Spring Boot Starter Data JPA
  - Propósito: Simplificar la persistencia de datos mediante JPA e Hibernate.
  - Uso en el proyecto: Gestiona el acceso a la base de datos MySQL a través de repositorios y entidades.
- Spring Boot Starter Security
  - Propósito: Proveer mecanismos de autenticación y autorización.
  - Uso en el proyecto: Protege los endpoints y gestiona la seguridad con JWT.
- Spring Boot Starter Validation
  - Propósito: Validar datos de entrada con Bean Validation (Jakarta).
  - Uso en el proyecto: Garantiza la integridad de los DTOs con anotaciones como @NotNull o @Email.
- Spring Boot Starter Actuator
  - Propósito: Monitorización y métricas del sistema.
  - Uso en el proyecto: Permite comprobar el estado y el rendimiento de la aplicación.
- Spring Boot Starter Mail
  - Propósito: Envío de correos electrónicos desde la aplicación.
  - Uso en el proyecto: Se emplea para enviar instrucciones de pago a los clientes.
- MySQL Connector/J
  - Propósito: Conectar la aplicación con bases de datos MySQL.
  - Uso en el proyecto: Permite la persistencia de datos en la base carta_go.
- Lombok
  - Propósito: Reducir código repetitivo mediante anotaciones.
  - Uso en el proyecto: Automatiza la generación de getters, setters y constructores.
- Java JWT (Auth0)
  - Propósito: Generación y validación de tokens JWT.
  - Uso en el proyecto: Implementa autenticación basada en tokens en el backend.
- Apache HttpClient 5
  - Propósito: Cliente HTTP para integraciones externas.
  - Uso en el proyecto: Facilita la comunicación con servicios externos como APIs de geocodificación.
- Google Cloud Vision
  - Propósito: Procesamiento de imágenes y reconocimiento de texto (OCR).
  - Uso en el proyecto: Digitaliza cartas físicas de restaurantes al extraer texto de imágenes.

2.3.2 Frontend

- React Native
  - Propósito: Framework para desarrollar aplicaciones móviles multiplataforma con JavaScript y React.
  - Uso en el proyecto: Base del frontend de la aplicación móvil, permitiendo construir pantallas y componentes que funcionan tanto en Android como en iOS.
- Expo
  - Propósito: Entorno que simplifica el desarrollo de aplicaciones en React Native, proporcionando herramientas y APIs listas para usar.
  - Uso en el proyecto: Facilita el despliegue, testing y uso de funcionalidades nativas como cámara, ubicación, fuentes o almacenamiento seguro.
- Expo Router
  - Propósito: Sistema de navegación basado en rutas de archivos para React Native.
  - Uso en el proyecto: Permite organizar la navegación de la aplicación de forma sencilla, gestionando pantallas y parámetros entre vistas.
- React Navigation (@react-navigation/native)
  - Propósito: Librería para manejar la navegación en aplicaciones móviles con pilas, tabs o drawer.
  - Uso en el proyecto: Gestiona la navegación entre pantallas como inicio de sesión, registro, carta del restaurante o salas de pago.
- React Query (@tanstack/react-query)
  - Propósito: Manejo de estados asincrónicos y cacheo de peticiones HTTP.
  - Uso en el proyecto: Optimiza las llamadas a la API del backend (ej. obtener cartas, restaurantes o salas de pago) almacenando los datos y evitando peticiones innecesarias.
- Axios
  - Propósito: Cliente HTTP para realizar peticiones a APIs.
  - Uso en el proyecto: Encargado de la comunicación con el backend (login, gestión de cartas, salas de pago, etc.).
- React Native Paper
  - Propósito: Conjunto de componentes UI basados en Material Design.
  - Uso en el proyecto: Mejora la apariencia de la app con botones, inputs, tarjetas y otros elementos visuales consistentes.
- React Native Maps
  - Propósito: Mostrar mapas y marcadores en aplicaciones móviles.
  - Uso en el proyecto: Visualiza la ubicación de los restaurantes en un mapa interactivo.
- Expo Location
  - Propósito: Acceso a la ubicación del dispositivo.
  - Uso en el proyecto: Obtiene la localización actual del cliente para buscar restaurantes cercanos.
- Expo Secure Store

    - Propósito: Almacenamiento seguro de datos sensibles.
    - Uso en el proyecto: Guarda tokens de autenticación y credenciales de usuario en el dispositivo de manera cifrada.
- Expo Image Picker
    - Propósito: Seleccionar imágenes de la galería o tomar fotos con la cámara.
    - Uso en el proyecto: Permite al restaurante subir imágenes de los platos de su carta.
- Expo Clipboard
    - Propósito: Interactuar con el portapapeles del dispositivo.
    - Uso en el proyecto: Facilita copiar datos como códigos de salas de pago.
- Expo Linear Gradient
    - Propósito: Renderizar fondos y elementos con degradados.
    - Uso en el proyecto: Mejora la estética visual de ciertas pantallas de la aplicación.
- React Native Reanimated
    - Propósito: Librería avanzada para animaciones fluidas y de alto rendimiento.
    - Uso en el proyecto: Añade animaciones suaves en la transición de pantallas y componentes interactivos.
- React Native Safe Area Context
    - Propósito: Manejo de las áreas seguras en iOS y Android (notch, barras de estado).
    - Uso en el proyecto: Garantiza que los contenidos no queden solapados por elementos del sistema operativo.
- React Native Screens
    - Propósito: Optimizar la navegación mediante componentes nativos de pantallas.
    - Uso en el proyecto: Mejora el rendimiento y la memoria al gestionar la pila de navegación.
- React Native Keyboard Aware Scroll View
    - Propósito: Ajustar automáticamente la vista cuando aparece el teclado.
    - Uso en el proyecto: Evita que los formularios de login y registro queden ocultos al escribir.
- @expo/vector-icons
    - Propósito: Conjunto de iconos populares para aplicaciones móviles.
    - Uso en el proyecto: Mejora la interfaz con iconos en botones, menús y secciones de la aplicación.
- @react-native-async-storage/async-storage
    - Propósito: Almacenamiento local persistente de datos.
    - Uso en el proyecto: Guarda información como configuraciones, historial o datos de usuario entre sesiones.

3

Especificación y análisis

3.1 Actores del Sistema

Dada la naturaleza de esta aplicación móvil y la necesidad de separar las funcionalidades visibles para los distintos usuarios de esta, se contemplan distintos roles de usuario. Estos roles son "cliente" y "restaurante" y cada uno de ellos contará con su propia interfaz gráfica independiente, así como permisos y opciones adaptadas a sus necesidades.

- **Cliente**: un cliente es cualquier persona que acuda a alguno de los restaurantes que estén registrados en la aplicación y que quiera utilizar las funciones de visualización de carta y división de cuentas que este le ofrece.
- **Restaurante**: un restaurante es cualquier establecimiento de restauración que utilice la aplicación para digitalizar su carta y mostrarla a sus clientes.

3.2 Requisitos Funcionales

En este apartado se detallarán todos los requisitos funcionales que se contemplan para cubrir todas las funcionalidades y necesidades de la aplicación. En la tabla 1 se puede observar cada requisito junto con su identificador y su descripción.

Tabla 1. Requisitos funcionales del sistema

Identificador de requisito	Nombre	Descripción
RF01	Registro de cliente	El sistema debe permitir el registro de un nuevo usuario, solicitando correo electrónico, nombre de

		usuario, contraseña y selección de rol como cliente.
RF02	Registro de restaurante	El sistema debe permitir el registro de un nuevo usuario, solicitando correo electrónico, nombre de restaurante, contraseña y selección de rol como restaurante.
RF03	Inicio de sesión	El sistema debe permitir el inicio de sesión de usuarios registrados mediante correo y contraseña.
RF04	Redirección de interfaz gráfica	El sistema debe redirigir al usuario a su interfaz correspondiente según el rol seleccionado al registrarse.
RF05	Escaneo de carta física	El restaurante debe poder escanear su carta física utilizando la cámara o importando imágenes desde la galería. La aplicación permite añadir varias páginas, que se procesan mediante OCR para generar una carta digital editable con platos en forma de tarjetas divididos en secciones.
RF06	Digitalización de carta escaneada	El sistema debe poder digitalizar cada elemento de la carta en formato de tarjeta editable.
RF07	Edición de carta digitalizada	El restaurante debe poder revisar y editar el contenido escaneado antes de publicarlo (nombre del plato, precio, imagen, alérgenos, etc.).
RF08	Organización de carta	El restaurante debe poder organizar las secciones de su carta digital. Puede crear nuevas secciones (que deben contener al menos un plato), renombrar secciones

		existentes y mover platos de una sección a otra.
RF09	Actualización de carta	El restaurante debe poder añadir, eliminar o modificar cualquier elemento de la carta, ya sea un plato o alguna información sobre este.
RF10	Subida de imágenes del plato	El restaurante debe poder añadir imágenes de cada plato cargándolas desde su dispositivo o tomándolas con la cámara.
RF11	Gestión del establecimiento	El restaurante debe poder gestionar la información pública de su establecimiento (nombre, dirección, horario, imagen).
RF12	Previsualización de carta	El restaurante debe poder previsualizar cómo se mostrará su carta a los clientes.
RF13	Búsqueda de establecimiento	El cliente debe poder buscar un restaurante registrado en la aplicación mediante su nombre y diversos criterios más como cercanía o abiertos ahora.
RF14	Selección de restaurante	El cliente debe poder seleccionar un restaurante y visualizar la información de este.
RF15	Consulta de carta	El cliente debe poder acceder y navegar a la carta de un restaurante.
RF16	Consulta de platos de una carta	El cliente debe poder consultar la información de cada plato de una carta (nombre, precio, descripción, alérgenos, imagen si está disponible).
RF17	Sala de pago	El cliente debe poder crear una sala de pago temporal en un restaurante concreto e invitar a otros

		usuarios mediante un código.
RF18	Selección de platos	El cliente debe poder seleccionar los platos que ha consumido dentro de una sala de pago y los clientes que han participado en cada uno de estos.
RF19	División de cuenta	El sistema debe calcular automáticamente la división de la cuenta entre los participantes de la sala, incluyendo platos compartidos y adaptándose a la forma de pago elegida por el cliente (personalizado o igualitario) .
RF20	Instrucciones de pago	El sistema debe generar instrucciones de pago indicando cuánto debe pagar cada usuario y un desglose por platos y consumiciones.

3.3 Requisitos no funcionales

En este apartado se detallarán todos los requisitos no funcionales que se contemplan para asegurar el correcto funcionamiento de la aplicación, así como el manejo de datos. En la tabla 2 se puede observar cada requisito junto con su identificador, tipo y descripción.

Tabla 2. Requisitos no funcionales del sistema

Identificador	Tipo de requisito	Nombre	Descripción
RNF1	Requisito de seguridad	Tratamiento seguro de datos	Los datos en la aplicación serán usados y almacenados de forma segura
RNF2	Requisito de producto	Compatibilidad multiplataforma	La aplicación estará desarrollada con React Native para asegurar su funcionamiento en dispositivos Android e iOS.

RNF3	Requisito de arquitectura	Arquitectura cliente-servidor	La aplicación funcionará bajo una arquitectura cliente-servidor, donde el cliente móvil realizará peticiones a un backend que gestiona la base de datos y la lógica de negocio.
RNF4	Requisito de usabilidad	Gestión de errores y feedback	La aplicación mostrará mensajes cuando se produzcan errores (por ejemplo, al fallar la carga del menú o el cálculo de la cuenta) e indicará al usuario cómo proceder.
RNF5	Requisito de rendimiento	OCR eficiente y rápido	El proceso de escaneo mediante OCR deberá completarse en un corto periodo de tiempo, proporcionando resultados legibles y editables.
RNF6	Requisito de disponibilidad	Acceso continuo	El servicio estará disponible de forma continua para permitir tanto la consulta de menús por parte de los clientes como la gestión de cartas por parte de los restaurantes.

3.4 Casos de uso

3.4.1 Casos de uso generales

Tabla 3. Caso de uso 1 - Registro del cliente

Titulo	Registro del cliente
Descripción	El cliente puede registrarse proporcionando correo electrónico,

	nombre, contraseña y seleccionando rol de cliente.
Pre-condición	El cliente no está registrado en la aplicación.
Post-condición	Se añade un nuevo usuario con un cliente asociado.
Escenario principal	
1. El cliente abre la aplicación 2. Se muestra una nueva pantalla con la opción de «iniciar sesión» o «registrarse» 3. El cliente selecciona la opción de «Registrarse» 4. El cliente introduce los datos necesarios y presiona el botón «Registrarse» 5. El sistema muestra un mensaje de éxito, guarda el cliente en la base de datos y redirige al cliente a la pantalla de principal de cliente.	
Escenario alternativo	
4.b. El cliente introduce los datos incompletos o inválidos. 5.b. El sistema muestra un mensaje de error. 6.b. Vuelta al punto 5	

Tabla 4. Caso de uso 2 - Registro del restaurante

Titulo	Registro de restaurante
Descripción	El restaurante puede registrarse proporcionando correo electrónico, contraseña, nombre del establecimiento, y rol como restaurante.
Pre-condición	El restaurante no está registrado en la aplicación.
Post-condición	Se añade un nuevo restaurante.
Escenario principal	
1. El restaurante abre la aplicación 2. Se muestra una pantalla con la opción de «iniciar sesión» o «registrarse» 3. El restaurante selecciona la opción de «Registrarse» 4. El restaurante introduce los datos necesarios y presiona el botón «Registrarse» 5. El sistema muestra un mensaje de éxito, guarda el restaurante en la base de datos y redirige al restaurante a la pantalla de Inicio de sesión	
Escenario alternativo	
4.b. El restaurante introduce los datos erróneos. 5.b. El sistema muestra un mensaje de error. 6.b. Vuelta al punto 5	

Tabla 5. Caso de uso 3 - Inicio de sesión

Titulo	Inicio de sesión
Descripción	El usuario puede iniciar sesión mediante su correo y contraseña.
Pre-condición	El usuario no está identificado en la aplicación.

Post-condición	El usuario está identificado en la aplicación.
Escenario principal	
1. El usuario abre la aplicación 2. El usuario selecciona la opción de «Iniciar sesión» 3. El usuario indica su correo electrónico y contraseña y presiona el botón «Iniciar sesión» 4. El sistema valida las credenciales y redirige al usuario a la pantalla principal correspondiente a su rol.	
Escenario alternativo	
3.b. El usuario introduce los datos erróneos. 4.b. El sistema muestra un mensaje de error. 5.b. Vuelta al punto 3	

3.4.2 Casos de uso del restaurante

Tabla 6. Caso de uso 4 - Escaneo de carta física

Titulo	Escaneo de carta física
Descripción	El restaurante puede escanear su carta física utilizando la cámara o importando imágenes desde la galería. La aplicación permite añadir varias páginas, que se procesan mediante OCR para generar una carta digital editable.
Pre-condición	El restaurante ha iniciado sesión en la aplicación.
Post-condición	La carta queda digitalizada y lista para ser editada en la aplicación.
Escenario principal	
1. El restaurante accede a la aplicación con su cuenta. 2. Desde el panel principal, selecciona la opción «Escanear carta». 3. La aplicación muestra una pantalla con dos opciones: «Usar cámara» e «Importar desde galería» 4. El restaurante elige «Usar cámara» y el sistema abre la cámara del dispositivo. 5. El restaurante toma una foto de la primera página de la carta. 6. La aplicación muestra la imagen como vista previa y permite confirmarla, eliminarla y añadir nueva página. 7. El restaurante repite el proceso hasta haber escaneado todas las páginas necesarias. 8. El restaurante pulsa «Digitalizar carta». 9. La aplicación procesa todas las imágenes mediante OCR y se genera la carta digital.	
Escenario alternativo	
5.c. Una imagen es ilegible o borrosa → La aplicación muestra un aviso y permite repetir la captura o cambiar la imagen. 8.d. El sistema no puede extraer correctamente el texto en alguna imagen → Muestra una advertencia y ofrece la opción de edición manual posterior.	

Tabla 7. Caso de uso 5 - Edición de carta

Titulo	Edición de carta
Descripción	El restaurante revisa y edita el contenido escaneado (nombre del plato, precio, imagen, alérgenos, etc.).
Pre-condición	La carta ha sido digitalizada mediante OCR y está disponible para su edición.
Post-condición	La carta queda actualizada con los cambios realizados por el restaurante.
Escenario principal	
1. El restaurante accede a la vista de edición tras finalizar el escaneo. 2. El sistema muestra una lista de platos generados a partir del OCR, cada uno como tarjeta editable. 3. El restaurante selecciona un plato para editarlo o eliminarlo o pulsa en el botón de «Añadir un plato». 4.Se abre un formulario con los campos editables: nombre, precio, descripción, imagen e información sobre alérgenos. 5. El restaurante introduce los campos que quiera modificar o añadir. 6. El restaurante presiona en el botón de «Guardar». 7. La aplicación guarda los cambios del plato o lo añade a la carta si es nuevo y vuelve a la lista general. 8. El restaurante repite el proceso hasta haber realizado todos los cambios necesarios. 9. El restaurante pulsa «Guardar cambios» para actualizar la carta con los cambios realizados.	
Escenario alternativo	
4.b. El restaurante cancela la edición sin guardar → El sistema descarta los cambios y vuelve a la lista.	

Tabla 8. Caso de uso 6 - Organización de carta

Titulo	Organización de carta
Descripción	El restaurante puede organizar su carta digital en categorías o secciones y asignar elementos a cada una de ellas.
Pre-condición	Existe una carta digital editable con al menos una sección y uno o varios platos.
Post-condición	La carta queda reorganizada según las modificaciones realizadas por el restaurante.
Escenario principal	
1. El restaurante accede a la vista de organización de carta desde el editor. 2. Se muestra la lista de secciones existentes con sus respectivos platos. 3. El restaurante puede realizar las siguientes acciones: 3.1. Crear nueva sección: 3.1.1. Pulsa «Añadir sección».	

3.1.2. Introduce un nombre para la nueva sección. 3.1.3. Añade uno o varios platos existentes a esa sección (es obligatorio). 3.1.4. Pulsa «Guardar». 3.2. Renombrar una sección existente: 3.2.1. Pulsa sobre el nombre de la sección. 3.2.2.Escribe el nuevo nombre. 3.2.3. Pulsa «Guardar cambios». 3.3. Mover platos entre secciones: 3.3.1. Selecciona un plato de una sección. 3.3.2. Elige la sección de destino. 3.3.3. El sistema mueve el plato y actualiza ambas secciones. 3.4. Eliminar sección vacía: 3.4.1. Si una sección no contiene platos, el restaurante puede seleccionarla y pulsar «Eliminar». 3.4.2. El sistema elimina la sección. 4. Una vez reorganizada la carta, el restaurante pulsa «Guardar organización». 5. El sistema guarda la nueva estructura de la carta.
Escenario alternativo
3.1.b. El restaurante intenta guardar una nueva sección sin platos asignados → El sistema muestra un mensaje de error y obliga al restaurante a añadir platos a la sección. 3.2.b. El restaurante deja el nombre en blanco al renombrar → El sistema muestra un aviso de campo obligatorio. 4.b. El restaurante sale sin guardar los cambios → El sistema solicita confirmación y advierte que se perderán las modificaciones.

Tabla 9. Caso de uso 7 - Subida de imágen de un plato

Titulo	Subida de imagen de un plato
Descripción	El restaurante puede añadir una imagen a un plato utilizando la cámara del dispositivo o importando una imagen desde la galería. Esta acción puede realizarse al crear o editar un plato.
Pre-condición	El restaurante se encuentra en el formulario de creación o edición de un plato.
Post-condición	La imagen queda vinculada al plato correspondiente.
Escenario principal	
1. El restaurante accede al formulario de edición o creación de un plato. 2. Pulsa el botón «Añadir imagen». 3. El sistema muestra dos opciones: «Tomar foto con la cámara» o «Seleccionar desde galería». 4. El restaurante elige una opción	

5. Si selecciona la cámara, se abre la aplicación de cámara y toma la foto, en caso de seleccionar la galería, se abre el selector de imágenes del dispositivo. 6. La imagen seleccionada se muestra en vista previa. 7. El restaurante confirma la selección. 8. El sistema asocia la imagen al plato y la muestra en el formulario.
Escenario alternativo
4.b. El sistema no tiene permisos para acceder a la cámara o galería → Muestra una advertencia y solicita al usuario activar los permisos desde ajustes. 5.c. La imagen seleccionada es de un formato no válido → El sistema muestra un mensaje de error y solicita una nueva selección. 6.d. El usuario cancela la selección → El sistema no guarda ninguna imagen.

Tabla 10 . Caso de uso 8 - Gestión de un establecimiento

Titulo	Gestión del establecimiento
Descripción	El restaurante puede editar la información pública de su establecimiento, incluyendo el nombre del local, la dirección, el horario de apertura y cierre, y la imagen representativa.
Pre-condición	El restaurante ha iniciado sesión en la aplicación.
Post-condición	La información del restaurante queda actualizada y visible para los clientes en su perfil público.
Escenario principal	
1. El restaurante pulsa el botón «Editar información del establecimiento» desde el panel principal. 2. El sistema muestra el formulario de edición con los campos actuales precargados. 3. El restaurante modifica los campos necesarios, incluyendo la subida o modificación de la imagen. 4. El restaurante pulsa el botón «Guardar cambios». 5. El sistema valida los datos introducidos. 6.Si la validación es correcta, el sistema guarda los cambios y muestra un mensaje de confirmación. 7. La nueva información queda reflejada en el perfil público del restaurante.	
Escenario alternativo	
4.b. El restaurante pulsa «Cancelar» → El sistema descarta los cambios y vuelve a la pantalla anterior. 5.c. Uno o más campos obligatorios están vacíos o son inválidos (por ejemplo, horario mal definido) → El sistema muestra un mensaje de error e impide guardar hasta que se corrija. 6.d. Se produce un error de conexión al guardar los datos → El sistema informa del error y permite reintentar más tarde.	

Tabla 11. Caso de uso 9 - Previsualización de carta

Titulo	Previsualización de carta
Descripción	El restaurante puede visualizar cómo se mostrará su carta digital a los clientes antes de publicarla. La previsualización refleja el diseño, el orden y la información de cada plato tal como la verá el cliente.
Pre-condición	El restaurante ha digitalizado su carta y accede a la vista de edición.
Post-condición	El restaurante ha visualizado la carta desde la perspectiva del cliente y puede decidir si continuar realizando cambios.
Escenario principal	
1. El restaurante accede al editor de carta digital. 2. Pulsa el botón «Previsualizar carta». 3. El sistema carga la carta en modo cliente (modo sólo lectura). 4. El restaurante puede navegar por la carta como si fuera un cliente: navegar entre secciones, consultar platos y ver imágenes e información de estos. 5. El restaurante pulsa «Salir de previsualización» para volver a la edición.	
Escenario alternativo	
4.b. Hay errores de formato (texto cortado, imagen que no carga…) → El sistema muestra advertencias sin impedir la visualización.	

3.4.3 Casos de uso del cliente

Tabla 12. Caso de uso 10 - Búsqueda de un establecimiento

Titulo	Búsqueda de un establecimiento.
Descripción	El cliente puede buscar restaurantes registrados en la aplicación utilizando distintos criterios de búsqueda como el nombre del establecimiento, ubicación o filtros por disponibilidad horaria.
Pre-condición	El cliente ha iniciado sesión en la aplicación.
Post-condición	Se muestra una lista de restaurantes que coinciden con los criterios introducidos por el cliente.
Escenario principal	
1. El cliente accede a la pantalla principal de búsqueda. 2. Introduce uno o más criterios de búsqueda, como nombre del restaurante, ubicación u horarios. 3. Pulsa el botón «Buscar». 4. El sistema consulta la base de datos y filtra los resultados. 5. Se muestra una lista de restaurantes que coinciden con la búsqueda. 6. El cliente puede pulsar sobre un restaurante para acceder a su perfil.	
Escenario alternativo	

2.b. El cliente no introduce ningún criterio → El sistema muestra una lista general de restaurantes cercanos o destacados. 3.c. Fallo de conexión al buscar → El sistema muestra un aviso y permite reintentar. 4.d. No se encuentran restaurantes que cumplan los criterios → El sistema muestra el mensaje "No se encontraron resultados" y sugiere modificar la búsqueda.

Tabla 13. Caso de uso 11 - Selección de un restaurante

Titulo	Selección de un restaurante
Descripción	El cliente selecciona un restaurante de los disponibles en una lista para visualizar su información y carta.
Pre-condición	El cliente ha realizado una búsqueda y tiene acceso a una lista de restaurantes disponibles.
Post-condición	El cliente accede al perfil público del restaurante seleccionado y puede visualizar su carta digital.
Escenario principal	
1. El cliente visualiza una lista de restaurantes como resultado de una búsqueda. 2. Pulsa sobre uno de los restaurantes listados. 3. El sistema carga el perfil público del restaurante. 4. Se muestra la información general del restaurante: nombre, imagen, dirección, horario de apertura. 5. El cliente puede pulsar el botón «Ver carta» para consultar la carta del establecimiento.	
Escenario alternativo	
3.b. El sistema no puede cargar la información del restaurante (por ejemplo, por error de red) → Muestra un mensaje de error e intenta recargar.	

Tabla 14. Caso de uso 12 - Consulta de carta

Titulo	Consulta de carta
Descripción	El cliente puede consultar la carta digital del restaurante seleccionado. La carta está organizada en secciones (por ejemplo, entrantes, principales, postres) y cada plato incluye su información detallada: nombre, precio, descripción, ingredientes, alérgenos e imagen si está disponible.
Pre-condición	El cliente ha seleccionado un restaurante que tiene su carta publicada.
Post-condición	El cliente visualiza la carta completa del restaurante, pudiendo consultar los detalles de cualquier plato.
Escenario principal	
1. El cliente pulsa el botón «Ver carta» en el perfil del restaurante.	

2. El sistema carga la carta digital publicada por el restaurante. 3. La carta se muestra organizada en secciones. 4. El cliente navega por las secciones de la carta. 5. Puede pulsar sobre cualquier plato para ver su información detallada: Nombre del plato, precio, descripción, alérgenos e imagen si está disponible.
Escenario alternativo
2.b. Error de carga de la carta → El sistema muestra un mensaje y ofrece reintentar. 5.c. La imagen de un plato no está disponible → El sistema muestra una imagen por defecto o indica que no hay imagen.

Tabla 15. Caso de uso 13 - Gestión de salas de pago compartido

Titulo	Gestión de salas de pago compartido
Descripción	El cliente puede crear una sala de pago temporal, que permite a varias personas seleccionar los platos que han consumido y dividir la cuenta. La sala puede compartirse con otros usuarios mediante un código.
Pre-condición	El cliente ha accedido a la carta de un restaurante publicado.
Post-condición	Se crea una sala de pago activa con un identificador único, lista para que los participantes seleccionen los platos consumidos.
Escenario principal	
1. El cliente pulsa el botón «Crear sala de pago» desde la vista de carta. 2. El sistema genera una sala temporal con un identificador único. 3. Se muestra un resumen con: Código para compartir la sala con otros participantes, nombre del restaurante y su carta. 4. El cliente comparte el código o enlace con los demás comensales. 5. Otros usuarios pueden unirse introduciendo el código. 6. Todos los participantes acceden a la sala, donde pueden visualizar la carta y seleccionar los platos que han consumido.	
Escenario alternativo	
2.b. El sistema no puede generar la sala por error de red → Muestra un aviso y permite reintentar. 5.c. Un participante introduce un código inválido → El sistema muestra un error y no permite acceder. 5.d. La sala ya ha expirado (por tiempo o acción del creador) → El sistema notifica que ya no está disponible.	

Tabla 16. Caso de uso 14 - Selección de platos en sala de pago

Titulo	Selección de platos en sala de pago
Descripción	Dentro de una sala de pago, cualquier participante puede registrar platos consumidos simplemente pulsando sobre

	ellos en la carta y seleccionando quiénes los han compartido. El sistema guarda esa información para realizar el cálculo de la cuenta.
Pre-condición	El cliente está dentro de una sala de pago activa y con otros participantes conectados.
Post-condición	La consumición queda registrada en la sala, asociada a una o varias personas según lo indicado.
Escenario principal	
1. El cliente accede a la sala de pago. 2. El sistema muestra la carta del restaurante asociada a esa sala. 3. El cliente pulsa sobre un plato de la carta. 4. El sistema abre una ventana emergente o pantalla donde puede seleccionar los clientes que han compartido ese plato. 5. El participante selecciona una o varias personas (incluyéndose o no a sí mismo). 6. El cliente pulsa el botón «Añadir». 7. El sistema registra la consumición en la sala con el nombre del plato, su precio y los participantes seleccionados. 8. La consumición queda visible en la lista compartida para todos los miembros de la sala.	
Escenario alternativo	
3.b. El plato ya ha sido añadido → El sistema permite volver a añadirlo (por ejemplo, si se pidió varias veces) o editarlo. 5.c. No se selecciona ningún participante → El sistema registra el plato a nombre de todos los clientes de la sala. 7.d. El usuario pulsa «Cancelar» → El sistema no guarda nada y vuelve a la carta. 8.e. El sistema permite editar o eliminar consumiciones previamente añadidas.	

Tabla 17. Caso de uso 15 - Cálculo automático de división de cuenta

Titulo	Cálculo automático de división de cuenta
Descripción	El sistema calcula automáticamente cuánto debe pagar cada participante de la sala de pago, en función de las consumiciones registradas y las personas asociadas a cada una.
Pre-condición	La sala de pago contiene una o más consumiciones con sus participantes registrados.
Post-condición	El sistema muestra el importe individual que debe pagar cada persona, calculado en base a su participación en los platos y envía un correo a cada uno de ellos con las instrucciones de pago detalladas.
Escenario principal	

1. Un cliente pulsa el botón «Calcular cuenta» desde la lista compartida en la sala de pago. 2.El sistema muestra una pantalla o ventana emergente con dos opciones de método de división: pago personalizado o igualitario. 3. El participante selecciona el modo de pago deseado. 4. El sistema recopila todas las consumiciones de la sala. 5. El sistema genera unas instrucciones de pago basándose en el modo de pago seleccionado: en el caso de división equitativa, se calcula el total de todos los platos y se divide entre el número de clientes de la sala de pagos. En el caso de división por participación, cada plato se divide entre los participantes seleccionados de ese plato. 6. El sistema muestra las instrucciones de pago con: nombre de cada participante, total individual a pagar y detalles del cálculo.
Escenario alternativo
2.b. El usuario cierra el selector sin elegir → El sistema cancela el cálculo. 4.c. No hay consumiciones registradas → El sistema muestra un mensaje de error.

4

Diseño del sistema

4.1 Arquitectura del sistema

El sistema se ha diseñado siguiendo una **arquitectura cliente-servidor**, en la que la interacción entre el cliente y el servidor se realiza mediante una **API REST** que centraliza la lógica de negocio y gestiona el acceso a los datos.

El **frontend** de la aplicación está desarrollado con **React Native** y utiliza **Expo Router** para la navegación, junto con librerías como **React Native Paper** para la interfaz de usuario. Gracias a esta tecnología, la aplicación es **multiplataforma**, pudiendo ejecutarse tanto en dispositivos Android como en iOS desde una única base de código.

El **backend** se ha implementado en **Spring Boot**, empleando una arquitectura modular basada en controladores, servicios y repositorios. La persistencia de datos se gestiona mediante **JPA/Hibernate** sobre una base de datos **MySQL**.

Durante el desarrollo en local, la base de datos se ha desplegado en un contenedor **Docker** con volumen persistente, lo que ha permitido **aislar el servicio de MySQL, facilitar su configuración y asegurar la conservación de los datos entre reinicios**. Este uso de Docker ha resultado especialmente útil para mantener un entorno de pruebas estable y reproducible.

En el entorno de producción, la base de datos se encuentra en un **servicio gestionado de MySQL en la nube**, concretamente en Railway, garantizando la disponibilidad y persistencia de los datos sin necesidad de mantener contenedores propios.

Por otra parte, el sistema integra **Cloudinary** para la gestión de imágenes (por ejemplo, fotos de restaurantes y platos), mientras que la **autenticación de usuarios** se realiza mediante **JWT (JSON Web Tokens)**, asegurando un inicio de sesión seguro y diferenciando los roles de cliente y restaurante.

En cuanto a la lógica principal, la arquitectura soporta dos grandes flujos:

- **Rol cliente**, que incluye la búsqueda de restaurantes, la visualización de cartas y la creación de salas de pago compartido.
- **Rol restaurante**, que contempla la gestión de la información del local, la digitalización de cartas mediante OCR y la administración de platos y secciones.

De esta forma, la arquitectura del sistema asegura una **separación clara de responsabilidades**, escalabilidad y facilidad de mantenimiento, sirviendo como base sólida para la implementación de las funcionalidades descritas en este proyecto.

4.2 Diagrama de arquitectura

En la Ilustración 13 se muestra un esquema general de la arquitectura planteada para el sistema. La aplicación móvil se conecta al backend mediante una API REST, que concentra la lógica de negocio organizada en distintos módulos: autenticación y roles, gestión de restaurantes y cartas, flujo de pago compartido e integraciones externas. La base de datos utilizada es MySQL, desplegada en contenedor Docker durante el desarrollo local y como servicio gestionado en Railway en producción. Además, se integran servicios externos como Cloudinary para el manejo de imágenes, OCR para la digitalización de cartas, geocodificación de direcciones y el envío de notificaciones por correo electrónico.

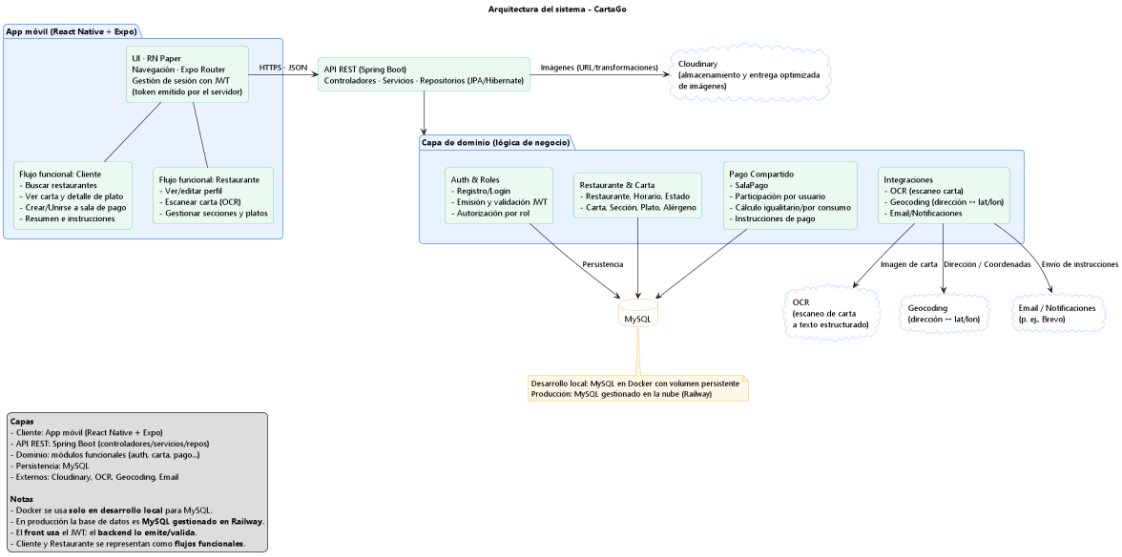


Ilustración 13. Diagrama de arquitectura

4.3 Diseño inicial de la base de datos

Para el almacenamiento de datos se ha decidido utilizar **MySQL**, un sistema de gestión de bases de datos relacional de código abierto ampliamente consolidado en la industria. Su elección se fundamenta en:

- **Madurez y fiabilidad**: MySQL ofrece estabilidad y rendimiento probado, con soporte en múltiples entornos (on-premise, cloud, contenedores), lo que asegura un funcionamiento predecible y adaptable al despliegue en Railway.
- **Modelo relacional adecuado**: El dominio del proyecto incluye entidades claramente definidas (usuarios, clientes, restaurantes, cartas, platos, etc.), lo que encaja de forma natural con un modelo relacional estructurado en tablas, claves foráneas e índices.

- **Integridad y consistencia**: La gestión de usuarios, menús y pagos requiere operaciones transaccionales y una fuerte integridad referencial, aspectos que MySQL maneja de forma eficiente.

En la Ilustración 14 se especifica el diseño de la base de datos.

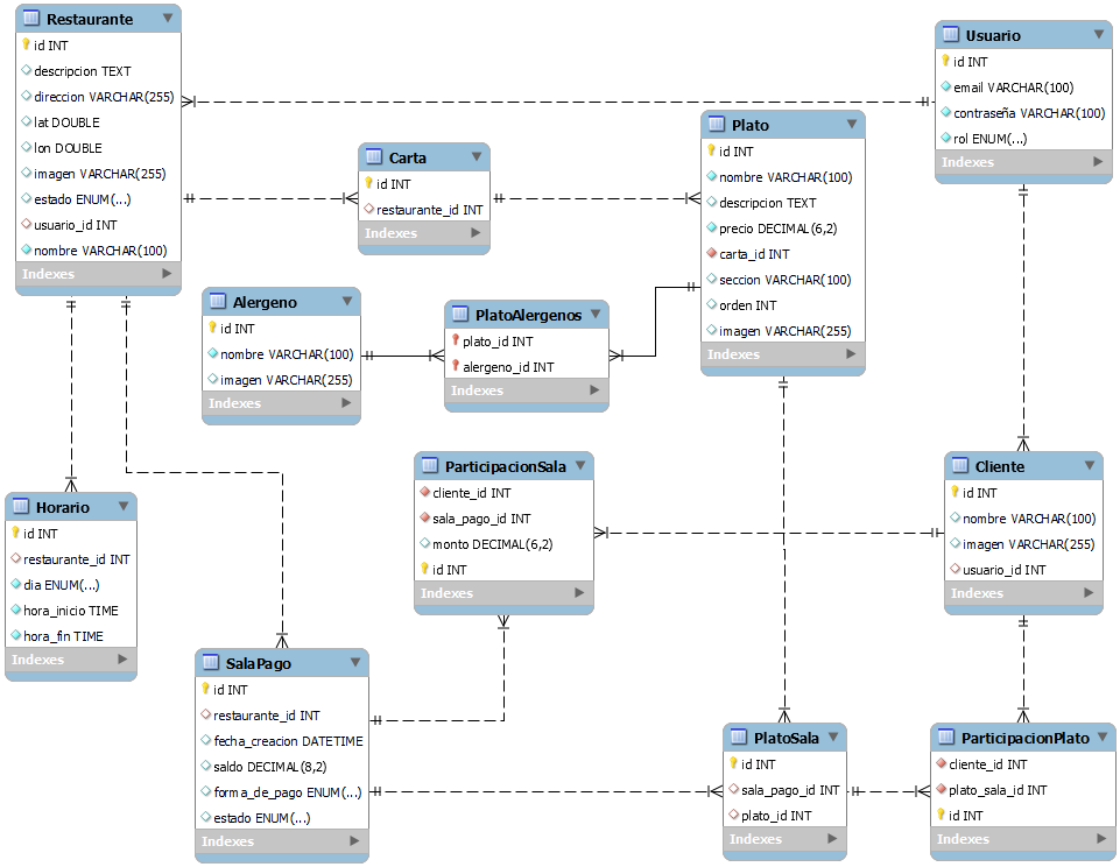


Ilustración 14. Diseño de la base de datos

La base de datos contiene tablas fundamentales como **Cliente**, **Restaurante**, **Carta** y **Plato**, que permiten gestionar la información principal relacionada con la lógica de negocio y los usuarios de la aplicación. Con el objetivo de aplicar una mejor separación de responsabilidades y seguir principios de normalización, se ha creado una tabla independiente llamada **Usuario** que almacena los datos relacionados con la autenticación (correo electrónico, contraseña y rol). De este modo, se desacoplan los datos de inicio de sesión de la información específica de los perfiles Cliente y Restaurante, facilitando la escalabilidad y el control de permisos en posibles futuras ampliaciones del sistema, así como facilitando el manejo de la información. Además, se incluyen tablas intermedias que permiten modelar relaciones más específicas o complejas, como las siguientes:

- **SalaPago**: almacena la información relacionada con las salas de pago creadas por los clientes dentro de un restaurante determinado.
- **PlatoSala**: representa un plato concreto que ha sido añadido a una sala de pago. Esta tabla es necesaria para distinguir distintas instancias del mismo plato (por

ejemplo, si varios clientes piden el mismo plato, pero deben pagar por separado). A su vez, esta entidad se relaciona con **ParticipacionPlato**.
- **ParticipacionPlato**: permite vincular cada plato añadido a una sala de pago con uno o varios clientes que lo hayan solicitado.
- **ParticipacionSala**: representa la participación de un cliente en una sala de pago concreta, junto con el importe que le corresponde.
- **PlatoAlergenos**: relaciona los platos de la carta con los distintos alérgenos que puedan contener, con el objetivo de ofrecer información relevante para los clientes.

4.4 Creación y configuración inicial del Backend

Para el desarrollo del backend se ha utilizado **Java** con el framework **Spring Boot**, una de las herramientas más consolidadas para la creación de aplicaciones empresariales en el ecosistema Java.
Su principal ventaja es la capacidad de generar un proyecto modular, escalable y de rápida configuración, gracias a su sistema de auto-configuración y a la amplia colección de dependencias disponibles mediante *starters*.

El proyecto se organiza en paquetes siguiendo una **separación por capas de responsabilidad**:
- **controller**: contiene los controladores REST que exponen los endpoints de la aplicación.
- **service**: incluye la lógica de negocio, separando la capa de presentación de la capa de datos.
- **entity**: define las entidades JPA que representan las tablas de la base de datos.
- **repository**: almacena las interfaces que extienden de JpaRepository para interactuar con la base de datos de forma sencilla.
- **dto**: agrupa los *Data Transfer Objects*, clases que definen cómo se estructuran los datos que entran y salen de la aplicación. Estos objetos permiten adaptar la información enviada al frontend según cada caso de uso, evitar exponer directamente las entidades de la base de datos y validar las peticiones entrantes antes de que lleguen a la capa de negocio.

4.5 Prototipado en Figma

Para el desarrollo de la aplicación se ha realizado un proceso de prototipado con Figma, una herramienta que permite diseñar interfaces de usuario de manera interactiva y colaborativa. A través de este prototipado se han definido tanto la estructura general de la aplicación como la disposición de los distintos elementos en cada pantalla, lo que ha facilitado la validación temprana de las funcionalidades principales y la coherencia visual del proyecto.

Este prototipo ha servido como guía para el posterior desarrollo del frontend, asegurando que la experiencia de usuario se mantenga alineada con los objetivos planteados.

En la siguiente sección se presentará con más detalle el prototipo diseñado, explicando las decisiones tomadas en cuanto a estilo, navegación y organización de la información.

4.5.1 Flujo de acceso común

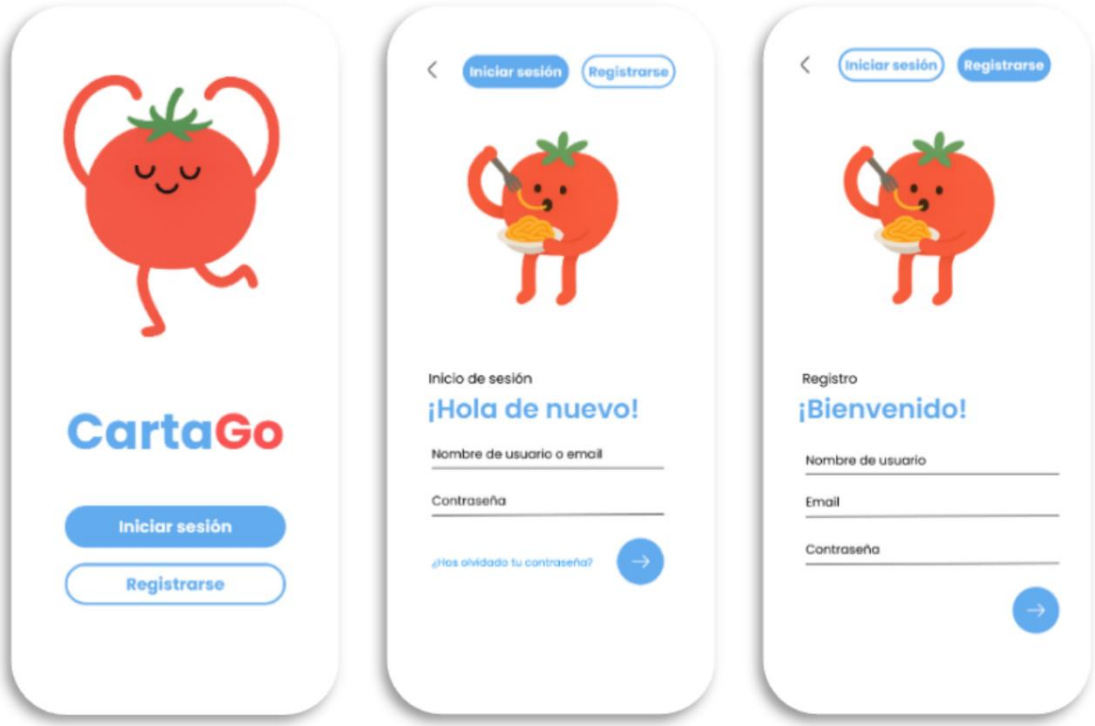


Ilustración 15. (Izda.) Prototipado de pantalla de bienvenida. (Centro) Prototipado de inicio de sesión. (Dcha.) Prototipado de pantalla de registro

Las primeras pantallas del prototipo corresponden al flujo de acceso común para clientes y restaurantes, ya que ambos perfiles de usuario comparten el mismo sistema de autenticación. Se diseñaron tres vistas principales:

- Pantalla de bienvenida (Ilustración 15 Izda.), con el logotipo de la aplicación "*CartaGo*" y un personaje ilustrado que aporta cercanía y un tono amigable. Se presentan dos botones principales —*Iniciar sesión* y *Registrarse*— en la parte inferior, con colores contrastados que permiten al usuario identificar rápidamente la acción que desea realizar.
- Pantalla de inicio de sesión (Ilustración 15 Centro), donde se mantiene la coherencia visual con el uso del mismo personaje ilustrado y un esquema cromático que combina el azul (asociado a confianza y seguridad) con detalles en rojo que refuerzan la identidad de la marca. La tipografía se mantiene clara y jerárquica, destacando el saludo inicial en un tono cercano ("¡Hola de nuevo!").
- Pantalla de registro (Ilustración 15 Dcha.), muy similar a la de inicio de sesión para reforzar la consistencia del diseño. Se emplean campos sencillos y bien delimitados para facilitar la introducción de datos, y un botón de acción en la parte inferior derecha que guía la navegación.

En cuanto a la paleta de colores, predominan el azul y el blanco como colores base, aportando limpieza y claridad visual, mientras que el rojo se reserva para el nombre de

la marca y algunos elementos gráficos, generando un contraste que facilita la recordación de la identidad visual.

El uso de formas redondeadas tanto en los botones como en los bordes de las pantallas busca transmitir una sensación de suavidad y accesibilidad, en línea con la experiencia de usuario deseada: intuitiva, cercana y sin fricciones. Asimismo, la inclusión de ilustraciones con personajes animados refuerza el carácter dinámico y moderno de la aplicación, diferenciándola de soluciones más frías o corporativas.

4.5.2 Flujo de cliente

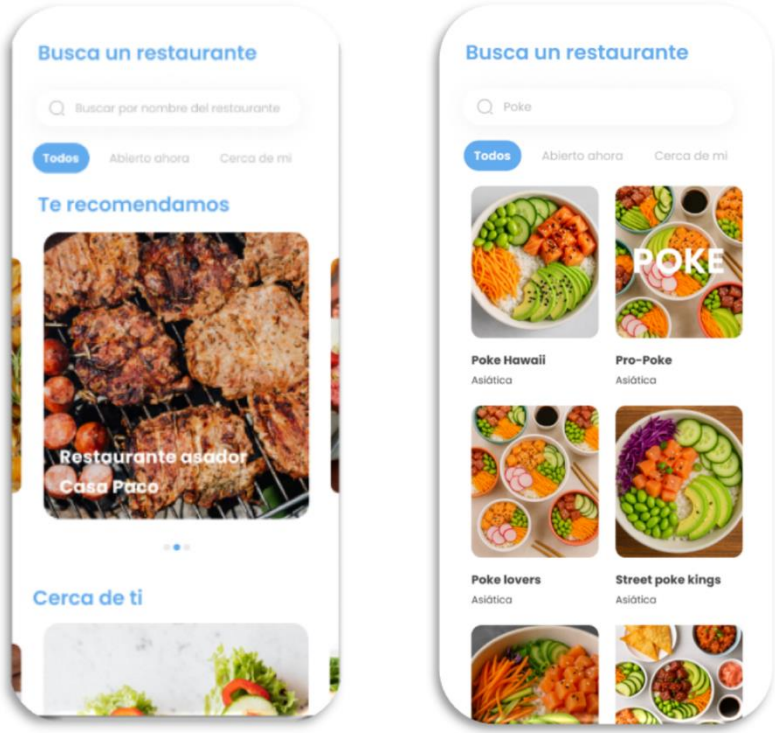


Ilustración 16. (Izda.) Prototipado de pantalla de vista principal de cliente. (Dcha.) Prototipado de buscador de restaurantes

En la **vista principal del cliente (Ilustración 16 Izda.)**, se organiza el acceso a los restaurantes disponibles en la aplicación. En la parte superior se sitúa un **buscador** que permite localizar restaurantes por nombre, acompañado de filtros rápidos (*Todos*, *Abierto ahora*, *Cerca de mí*). De este modo, el usuario puede ajustar los resultados de manera inmediata según sus necesidades, ya sea encontrar un restaurante abierto en ese momento o localizar opciones cercanas.

A continuación, se presentan dos propuestas de bloques diferenciados de información:

- **Recomendaciones**, que destacan restaurantes relevantes o populares mediante tarjetas visuales de gran tamaño, con imágenes llamativas que facilitan la identificación inmediata.
- **Cerca de ti**, que prioriza los restaurantes en función de la localización del usuario, reforzando la utilidad práctica de la aplicación para situaciones cotidianas.

En la segunda vista (Ilustración 16 Dcha.) se muestra cómo aparece la interfaz tras realizar una búsqueda específica (en este caso, restaurantes de tipo *poke*). Los

resultados se disponen en un formato de **tarjetas cuadradas más compactas**, con imagen principal y nombre del restaurante, lo que permite visualizar varias opciones en pantalla de manera simultánea y agilizar la toma de decisiones.
En conjunto, estas pantallas buscan facilitar la exploración del catálogo de restaurantes a través de un diseño claro, filtros accesibles y un uso intensivo de imágenes como elemento principal de referencia.

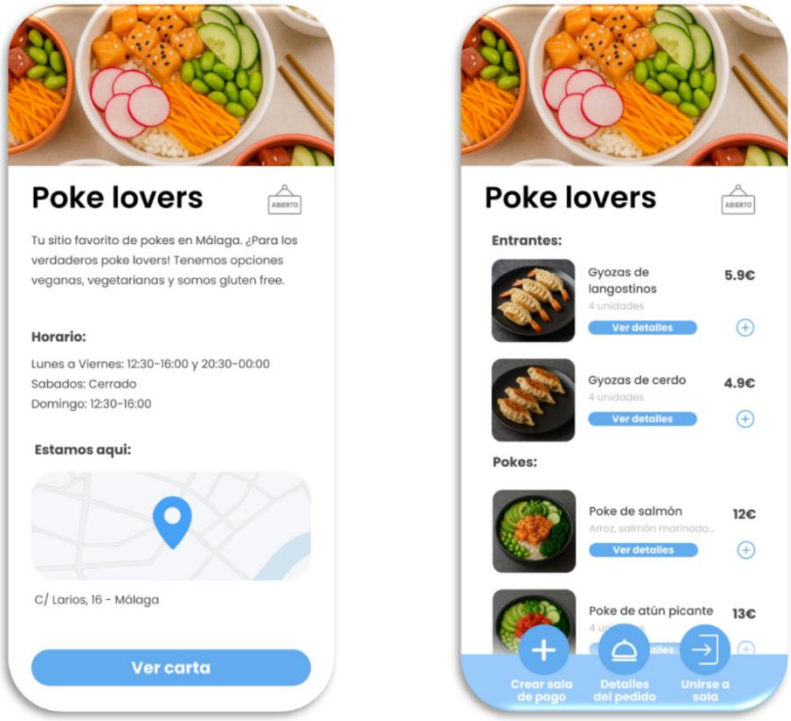


Ilustración 17. (Izda.) Prototipado de pantalla de información de restaurante. (Dcha.) Prototipado de carta

En la Ilustración 17 (Izda.) se muestra la información general del restaurante seleccionado. En la parte superior se incluye una imagen destacada como elemento visual principal, seguida del nombre y una breve descripción orientada a presentar la propuesta gastronómica del local. Justo debajo, se ofrece información práctica como el horario de apertura y un mapa con la dirección exacta, lo que facilita la localización física del restaurante. En la parte inferior se sitúa un botón de acción claro (*Ver carta*) que guía al usuario hacia la siguiente interacción lógica.

La Ilustración 17 (Dcha.) corresponde a la visualización de la carta del restaurante. Los platos se organizan por categorías (por ejemplo, *Entrantes* y *Pokes*), con cada elemento acompañado de una fotografía, nombre, breve descripción o unidades incluidas, y el precio. Además, se añaden botones de acción (*Ver detalles* y añadir al pedido con el símbolo "+" en caso de que el cliente este en una sala de pago), que permiten al usuario explorar más información o ir construyendo directamente su selección de platos.

En la parte inferior se incluye una barra de acciones rápidas orientada al proceso de pago compartido, con opciones como *Crear sala de pago*, *Ver detalles del pedido* o *Unirse a una sala*. Esto conecta la experiencia de consulta de la carta con la funcionalidad principal de la aplicación: facilitar la gestión de pedidos en grupo y la división de gastos.

En conjunto, estas pantallas refuerzan la idea de que el usuario pueda pasar de manera fluida de la búsqueda y selección de un restaurante a la consulta de su carta y posterior creación o unión a una sala de pago.

Ilustración 18. Prototipado de pantalla de vista detallada de un plato

La pantalla de la Ilustración 18 corresponde a la **vista detallada de un plato concreto** dentro de la carta del restaurante. Su objetivo es proporcionar al usuario toda la información necesaria para tomar una decisión informada antes de añadirlo a su pedido.

En la parte superior se presenta una **imagen destacada del plato**, ocupando un espacio central para transmitir de manera visual su aspecto y atractivo gastronómico. Debajo, se muestra el **nombre del plato acompañado de su precio**, con una tipografía clara y un contraste suficiente para resaltar esta información clave.

El apartado de descripción aporta un **texto explicativo** que detalla tanto la cantidad como las características del plato (en este caso, cuatro unidades de gyozas de cerdo, su preparación y acompañamiento). Esto no solo informa, sino que también contribuye a crear una experiencia más cercana y atractiva para el usuario.

Finalmente, se incluye una sección dedicada a los **alérgenos**, representados mediante iconos y etiquetas visuales. Este apartado cumple un papel fundamental en términos de **accesibilidad y seguridad alimentaria**, ya que permite identificar rápidamente posibles restricciones dietéticas.

Con este diseño, se busca que el usuario tenga una experiencia clara y completa: puede apreciar el plato visualmente, conocer su descripción, confirmar el precio y verificar los alérgenos antes de avanzar en el flujo de compra.

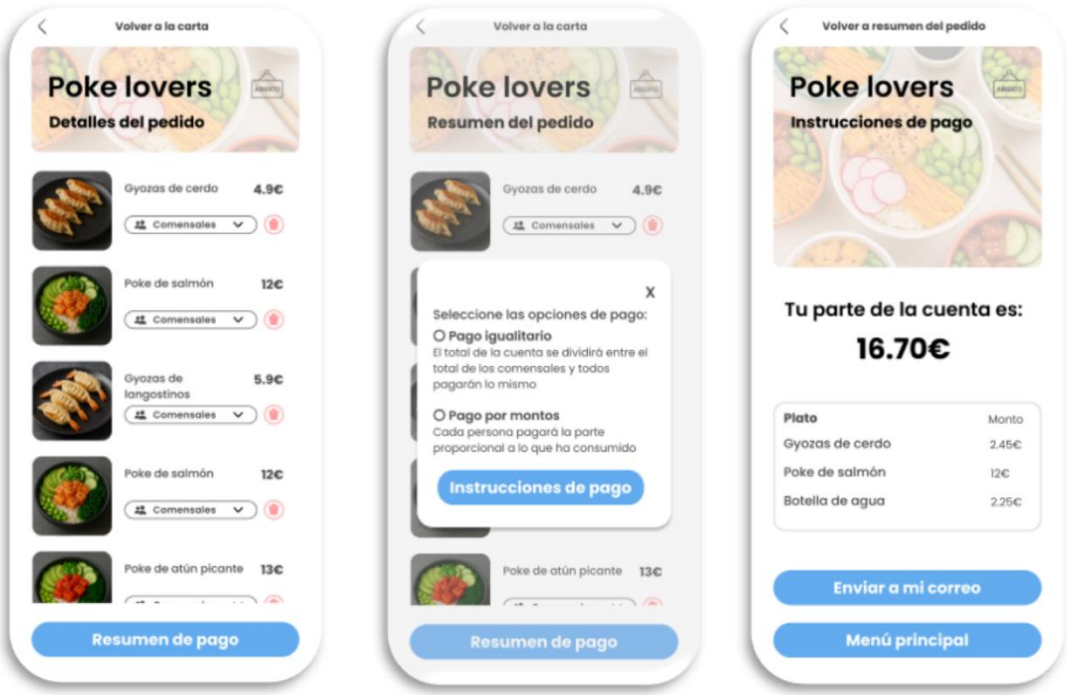


Ilustración 19. (Izda.) Prototipado de pantalla de detalles del pedido. (Centro) Prototipado de pantalla de selección de opciones de pago. (Dcha.) Prototipado de pantalla de instrucciones de pago.

Las siguientes pantallas corresponden al **tramo final del flujo de cliente**, donde se gestionan los detalles del pedido y la división de la cuenta:

En la primera vista se muestran los **detalles del pedido (Ilustración 19 Izda.)**, con todos los platos seleccionados, su precio y la asignación a los diferentes comensales. Esto permite identificar de manera clara qué ha pedido cada persona, preparando el terreno para la división del pago. En la parte inferior se incorpora un botón de acción (*Resumen de pago*) que conduce al siguiente paso.

La segunda pantalla recoge el **resumen del pedido**, incorporando además la elección de la modalidad de pago **(Ilustración 19 Centro).** Se presentan dos opciones principales:

- **Pago igualitario**, donde el importe total se divide de forma equitativa entre los comensales.
- **Pago por montos**, en el que cada persona abona únicamente lo correspondiente a lo que ha consumido.

De este modo, se abordan los dos escenarios más habituales al compartir gastos en un restaurante, aportando flexibilidad al usuario.

Finalmente, la tercera pantalla corresponde a las **instrucciones de pago personalizadas (Ilustración 19 Dcha.)**, donde se muestra la cantidad exacta que debe abonar el cliente, junto con un desglose de los platos asignados. Además, se incluyen opciones prácticas como "*Enviar a mi correo*" para recibir un recordatorio o comprobante, y "*Menú principal*" para regresar a la pantalla inicial del cliente.

En conjunto, estas pantallas materializan la funcionalidad diferencial de la aplicación: **simplificar la gestión de pedidos compartidos y automatizar la división de cuentas**, un proceso que en la práctica suele generar fricciones y cálculos manuales entre los comensales.

4.5.3 Flujo de restaurante

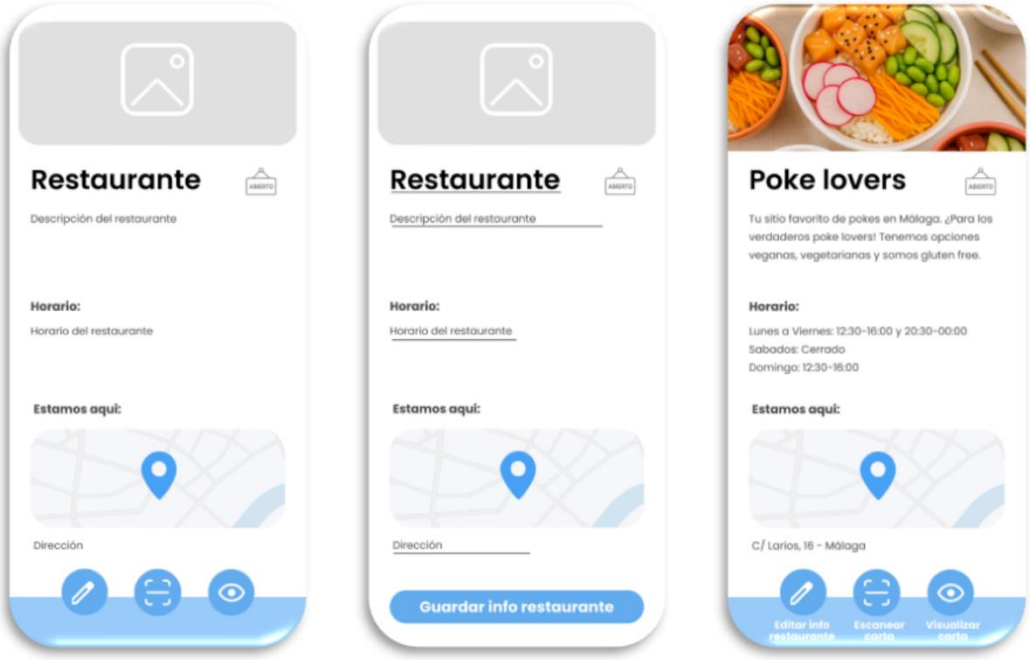


Ilustración 20. (Izda.) Prototipado de pantalla de información del restaurante vacía. (Centro) Prototipado de pantalla de edición de información del restaurante. (Dcha.) Prototipado de pantalla de información del restaurante completa.

Las siguientes pantallas corresponden al **rol de restaurante**, donde los usuarios con este perfil pueden gestionar la información básica de su local dentro de la aplicación:

Cuando un restaurante accede por primera vez tras completar el registro, se encuentra con su **pantalla de información vacía (Ilustración 20 Izda.)**. En este estado inicial aparecen los campos principales —nombre, descripción, horario y dirección— sin completar, junto con una imagen genérica.

la parte inferior se disponen tres botones de acción:

- **Editar información del restaurante**, que permite abrir un formulario para completar o modificar los datos (segunda imagen).
- **Escanear carta**, orientado a la digitalización automática de la carta mediante el uso de la cámara.
- **Visualizar carta**, que mostrará al restaurante su carta en el mismo formato en que la verán los clientes.

Si el restaurante selecciona la opción de **editar información (Ilustración 20 Centro.)**, se abre el formulario correspondiente donde puede añadir su descripción, horario y ubicación exacta. Una vez guardados los cambios, la pantalla se actualiza y pasa a

mostrarse como en la **tercera imagen (Ilustración 20 Dcha.)**. ya con los datos completados y listos para ser consultados por los clientes.

De este modo, el flujo inicial del rol restaurante asegura que cada local pueda configurar su perfil desde cero, y que la información mostrada sea siempre coherente y actualizada.

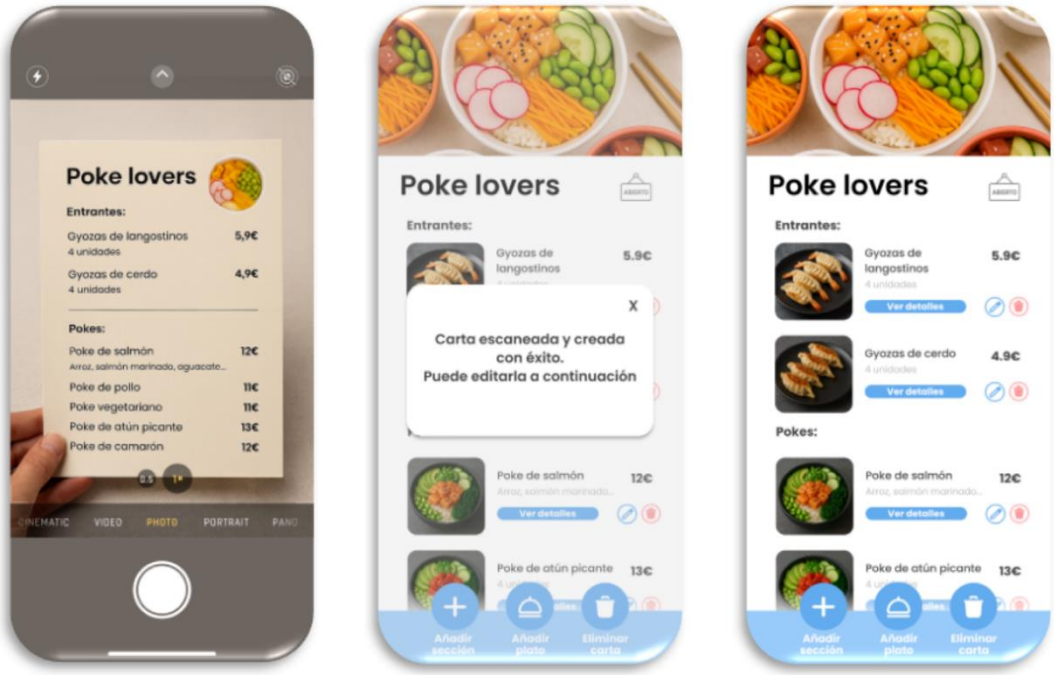


Ilustración 21. (Izda.) Prototipado de pantalla de escaneo de carta. (Centro) Prototipado de pantalla de confirmación de escaneo de carta. (Dcha.) Prototipado de pantalla de carta escaneada.

En el siguiente conjunto de pantallas se refleja la funcionalidad de **digitalización de la carta** por parte del restaurante:

En la primera vista se muestra el proceso de captura fotográfica de la carta física mediante la cámara del dispositivo **(Ilustración 21 Izda.).** Esta opción facilita a los restaurantes trasladar de manera rápida y sencilla su carta tradicional al entorno digital, evitando que tengan que introducir manualmente todos los platos desde cero.

Una vez realizada la captura, el sistema procesa la imagen y genera automáticamente una carta digital preliminar **(Ilustración 21 Centro).** En este paso se informa al usuario de que la carta ha sido escaneada con éxito y que puede **editarla a continuación** para revisar o ajustar los datos.

La tercera pantalla muestra la **carta ya digitalizada (Ilustración 21 Dcha.)** dentro de la aplicación, con los platos organizados en secciones y acompañados de sus precios e imágenes. Además, en la parte inferior aparecen los botones de acción que permiten al restaurante **añadir secciones**, **añadir platos nuevos** o **eliminar la carta completa**, ofreciendo así un control total sobre el resultado del escaneo.

Este flujo combina la **automatización** mediante el reconocimiento de texto con la **flexibilidad de edición manual**, lo que garantiza que la información final refleje fielmente la oferta gastronómica de cada restaurante.

Ilustración 22. (Izda.) Prototipado de pantalla de creación de un plato. (Dcha.) Prototipado de pantalla de creación de sección.

Las siguientes pantallas completan el flujo del **rol restaurante** en la gestión de la carta:

La primera vista corresponde al **formulario para añadir un nuevo plato (Ilustración 22 Izda.)**, donde el restaurante puede incluir una imagen representativa, el nombre del plato, su descripción y el precio. Además, se contempla la posibilidad de asociar los **alérgenos** con sus respectivos iconos, reforzando la transparencia y seguridad para los clientes con necesidades específicas.

En la segunda pantalla se muestra la acción de **añadir una nueva sección (Ilustración 22 Dcha.)** dentro de la carta (por ejemplo, “Postres” o “Bebidas”), lo que permite organizar los platos de manera estructurada y clara. Esta opción dota al restaurante de flexibilidad para gestionar su menú de acuerdo con la lógica de su oferta gastronómica.

De este modo, el flujo de restaurante cubre todas las operaciones principales: configurar su perfil, escanear la carta, añadir platos manualmente, organizar secciones y mantener su menú actualizado en todo momento.

Conclusión

El prototipo elaborado permite visualizar de forma integral el funcionamiento de la aplicación tanto para clientes como para restaurantes. A través de estas pantallas se han definido los **flujos principales de interacción**, validando la estructura de la aplicación antes de pasar al desarrollo.

El enfoque adoptado equilibra la **claridad funcional** (con procesos como búsqueda de restaurantes, gestión de pedidos o digitalización de cartas) con una **experiencia de usuario intuitiva**, apoyada en un diseño limpio, botones de acción claros y una organización coherente de la información.

En conclusión, el prototipado ha servido como una herramienta clave para **anticipar la experiencia final del usuario, detectar mejoras tempranas y establecer una guía visual y funcional** para el desarrollo posterior de la aplicación.

5

Implementación del backend

5.1 Gestión de dependencias y persistencia de datos

El backend de *CartaGo* se ha diseñado siguiendo una arquitectura modular y apoyándose en herramientas que facilitan tanto el desarrollo como el despliegue. A continuación, se describen las librerías principales y la configuración de la persistencia de datos.

Para la construcción y gestión de dependencias se utiliza **Maven**, lo que permite centralizar la configuración del proyecto en un archivo pom.xml. En él se declaran librerías fundamentales como:

- **spring-boot-starter-web**: para crear controladores REST.
- **spring-boot-starter-data-jpa**: para la integración con JPA e Hibernate.
- **spring-boot-starter-security**: para la gestión de autenticación y autorización.
- **mysql-connector-j**: para conectar con la base de datos MySQL.
- **lombok**: para reducir la verbosidad en las clases con anotaciones como @Getter, @Setter o @Builder.

En lo relativo a la persistencia de datos, la base de datos **MySQL** se ejecuta en un **contenedor Docker independiente**, configurado con credenciales seguras y una base de datos inicial llamada carta_go. Para garantizar la persistencia de la información, se define un volumen Docker que mantiene los datos, aunque el contenedor se reinicie o elimine.
La inicialización de la base de datos se realiza a través de un **script SQL de arranque**, que crea las tablas y relaciones definidas en el modelo entidad-relación, asegurando la coherencia entre la definición teórica y la implementación práctica.
El backend se conecta a la base de datos mediante el archivo de configuración application.properties, en el que se definen parámetros como el host, el puerto, el nombre de la base de datos, el usuario y la contraseña. Spring Boot gestiona

automáticamente la conexión a través de **Spring Data JPA**, lo que simplifica las operaciones CRUD y garantiza integridad y consistencia en las transacciones.

5.3 Estructura final del Backend

La organización del backend se ha diseñado con el objetivo de mantener una arquitectura clara, escalable y fácilmente mantenible. La estructura del proyecto se divide en dos niveles principales: la raíz del proyecto y la carpeta src, donde reside el código fuente y los recursos asociados.

5.3.1. Estructura raíz del proyecto

En la raíz del proyecto se encuentran varios archivos y carpetas clave que permiten gestionar la configuración, dependencias y despliegue del backend:

- **pom.xml**: archivo de configuración de Maven, en el que se definen todas las dependencias necesarias para el funcionamiento del proyecto (Spring Boot, Spring Data JPA, Spring Security, Lombok, MySQL Connector, entre otras).
- **.gitignore:** especifica qué archivos y carpetas deben excluirse del control de versiones (por ejemplo, la carpeta target/ o ficheros de configuración sensibles).
- **mvnw y mvnw.cmd**: *wrappers* de Maven que permiten ejecutar comandos de construcción sin necesidad de tener Maven instalado globalmente en el sistema.
- **.mvn/:** contiene archivos internos necesarios para el uso de Maven Wrapper.
- **docs/:** carpeta destinada a almacenar documentación auxiliar del proyecto.
- **target/:** carpeta generada automáticamente por Maven donde se almacenan los archivos compilados y el empaquetado final en formato .jar.
- **README.md:** documento de referencia para explicar la configuración y uso básico del backend.

5.3.2. Carpeta src: Estructura y componentes clave

Dentro de la carpeta src/main/java/com.cartaGo.cartaGo_backend se encuentra el código fuente del proyecto, organizado en paquetes siguiendo una separación por responsabilidades:

- **config**: incluye clases de configuración global, como la definición de *CORS*, beans personalizados y otros parámetros que afectan al comportamiento general del sistema.
- **controller**: contiene los controladores REST que exponen los diferentes endpoints de la API. Son responsables de recibir las peticiones HTTP del frontend y delegarlas a la capa de servicios.
- **dto**: agrupa los *Data Transfer Objects*, clases que definen la estructura de los datos que entran y salen del backend. Estos objetos permiten enviar al frontend solo la información necesaria, evitando exponer directamente las entidades de la base de datos.
- **entity**: define las entidades JPA que representan las tablas de la base de datos. Cada clase está anotada con @Entity y establece relaciones mediante anotaciones como @OneToMany, @ManyToOne o @ManyToMany.

- **mapper**: contiene clases y utilidades que facilitan la conversión entre entidades y DTOs, centralizando la lógica de mapeo y simplificando los controladores y servicios.
- **repository**: incluye las interfaces que extienden de JpaRepository, lo que permite realizar operaciones CRUD y consultas personalizadas sobre las entidades sin necesidad de escribir código SQL.
- **security**: implementa la lógica de autenticación y autorización de la aplicación. Aquí se definen la configuraciónde JWT, los filtros de seguridad y los roles de usuario (Cliente y Restaurante).
- **service**: contiene la lógica de negocio de la aplicación. Sus clases implementan las operaciones principales sobre las entidades y orquestan las reglas de negocio antes de devolver datos a los controladores.
- **utils**: incluye clases auxiliares y métodos de utilidad que se reutilizan en distintas partes del backend.
- Adicionalmente, la carpeta src/main/resources contiene archivos de configuración esenciales:
- **application.properties**: especifica parámetros de conexión a la base de datos (host, puerto, nombre, usuario y contraseña), así como otras variables relacionadas con seguridad, mensajería o almacenamiento en la nube.
- **static/ y templates/** (si se usan): directorios reservados para recursos estáticos o plantillas.

De este modo, la estructura final del backend (Ilustración 23) garantiza una separación clara de responsabilidades, facilitando la mantenibilidad del código, la incorporación de nuevas funcionalidades y la escalabilidad futura del sistema.

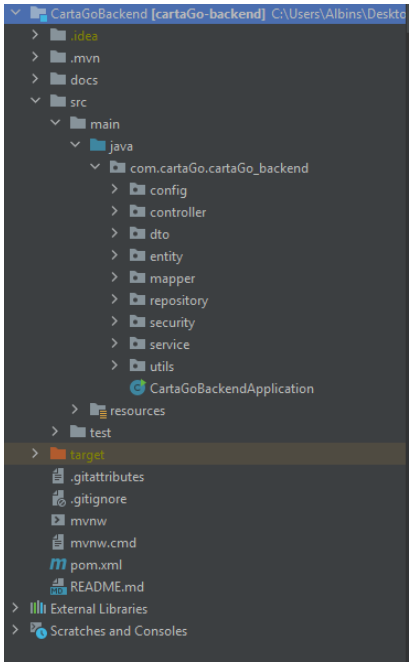


Ilustración 23. Carpetas del Backend

5.4 APIs utilizadas

Una **API (Application Programming Interface)** es un conjunto de reglas y definiciones que permite a dos aplicaciones comunicarse entre sí. En términos prácticos, una API actúa como un puente que facilita el intercambio de información o la utilización de servicios externos sin necesidad de implementarlos desde cero.
En este proyecto se han utilizado varias APIs externas para cubrir funcionalidades clave como la geolocalización, la gestión de imágenes, la digitalización de cartas y el envío de correos electrónicos. Gracias a estas integraciones, el sistema puede ofrecer servicios avanzados con un coste de desarrollo mucho menor, aprovechando soluciones ya existentes, fiables y escalables.

En los siguientes apartados se especifican las APIs externas utilizadas en este proyecto y su funcionamiento.

5.4.1. Geocodificación con Nominatim (geocode.maps.co)

Para la funcionalidad de localización de restaurantes se ha integrado el servicio externo **Nominatim**, a través de la API pública proporcionada por **geocode.maps.co**. Este servicio permite realizar operaciones de *geocoding* (obtener coordenadas a partir de una dirección) y *reverse geocoding* (obtener una dirección a partir de coordenadas).

El flujo de uso es el siguiente:

1. El cliente (aplicación móvil en React Native) obtiene la dirección escrita por el usuario o las coordenadas actuales mediante la geolocalización del dispositivo.
2. El cliente envía estos datos al backend (Spring Boot), que encapsula la lógica de acceso al servicio externo mediante un GeocodingService.
3. El backend construye la llamada HTTP al endpoint adecuado de Nominatim (/search para dirección → coordenadas o /reverse para coordenadas → dirección), incluyendo los parámetros necesarios como format=jsonv2.
4. El servicio externo responde en formato JSON con información como lat, lon, display_name y datos estructurados de la dirección.
5. El backend procesa la respuesta y devuelve un DTO con la información relevante (por ejemplo, la dirección formateada o las coordenadas del restaurante).
6. El cliente utiliza estos datos para mostrar en la interfaz la dirección textual, situar un marcador en el mapa o calcular qué restaurantes están dentro de un radio determinado.

Endpoints implementados en el backend:

- **GET /api/geocode/reverse?lat={lat}&lon={lon}**: devuelve la dirección correspondiente a unas coordenadas dadas.
- **GET /api/geocode/forward?q={direccion}**: devuelve una lista de posibles coordenadas para la dirección indicada (se conserva la primera).

5.4.2. Gestión de imágenes con Cloudinary

Para la gestión de imágenes en la aplicación se integró el servicio externo **Cloudinary**,

que permite almacenar ficheros multimedia en la nube y obtener una URL segura para su posterior uso en la interfaz.

El flujo de uso es el siguiente:

1. El cliente selecciona una foto (desde galería o cámara) y la sube directamente a Cloudinary mediante un *upload preset unsigned*.
2. Cloudinary devuelve una respuesta en formato JSON con la URL pública (secure_url) donde queda alojada la imagen.
3. El cliente envía esta URL al backend, que la guarda como un campo String en la entidad correspondiente (por ejemplo, Restaurante.imagen).
4. Al consultar un restaurante, el backend devuelve también la URL almacenada, que se usa directamente en el frontend para mostrar la imagen.

Endpoints implementados en el backend:

- **GET /restaurantes/{id}/imagen**: devuelve la URL de la imagen asociada a un restaurante o plato.
- **PUT /restaurantes/{id}/imagen**: permite asignar o reemplazar la URL de una imagen previamente subida a Cloudinary.
- **DELETE /restaurantes/{id}/imagen**: elimina la referencia a la imagen (pone el campo imagen a null).

5.4.3. Integración de OCR y parseo de menús con Google Cloud Vision

Para automatizar la digitalización de cartas, se integró **Google Cloud Vision** en modo *Document Text Detection*.
El flujo es el siguiente:

- La aplicación móvil sube la imagen de la carta a Cloudinary, que devuelve una URL pública segura.
- El backend recibe esa URL y, mediante el cliente oficial de Google Cloud Vision, solicita el OCR de la imagen. El servicio devuelve un bloque de texto plano con el contenido reconocido (nombres, precios y descripciones).
- A partir de ese texto, se aplica un *parser* propio que segmenta la información en secciones (Entrantes, Principales, Postres) y detecta platos y precios mediante expresiones regulares tolerantes a distintos formatos (12€, 5,9€, 13.50 €, etc.).
- Se identifican secciones tanto por delimitadores (:) como por heurísticas (encabezados cortos en mayúsculas o palabras clave).
- El resultado se transforma en una lista de PlatoRequestDTO con los campos seccion, nombre, precio y descripcion.
- El proceso es semi-automático: el restaurante puede revisar y editar la información después de la importación.

5.4.4. Gestión de correos con Brevo

Para la funcionalidad de envío de notificaciones automáticas se utilizó inicialmente el SMTP de Gmail, pero al desplegar el backend en Railway surgieron problemas de autenticación y bloqueo de credenciales, debido a las restricciones de seguridad de Google.

Para solventar este problema se migró a Brevo (antes Sendinblue), un servicio de correo transaccional que ofrece un plan gratuito y una integración sencilla mediante API HTTP o SMTP.

El flujo de uso es el siguiente:

1. El backend detecta un evento que requiere notificación (por ejemplo, el cambio de contraseña).
2. El MailService construye la petición HTTP hacia la API de Brevo con los parámetros necesarios: remitente, destinatario, asunto y cuerpo del mensaje.
3. Brevo procesa la solicitud y devuelve una respuesta con el estado del envío.
4. El backend confirma al cliente si el correo se envió correctamente.

Ejemplo de uso:

- Cambio de contraseña: cuando un cliente quiere recuperar su contraseña, se le envía un correo con el código que necesitará para cambiar su contraseña por una nueva.

Gracias a esta migración, el sistema de notificaciones por correo funciona de forma estable tanto en local como en producción.

6

Implementación del Frontend

6.1 Arquitectura y organización del proyecto

El frontend de la aplicación se ha desarrollado utilizando **React Native** con **Expo**, lo que permite crear aplicaciones móviles multiplataforma (Android e iOS) a partir de un único código base en JavaScript. La elección de esta tecnología responde a la necesidad de contar con un desarrollo ágil, componentes nativos y una integración sencilla con APIs externas.
Para gestionar la navegación se utiliza **Expo Router**, que implementa un sistema de enrutado basado en la estructura de carpetas (*file-based routing*). Gracias a esta característica, el flujo de pantallas se organiza de manera declarativa y escalable, simplificando la gestión de rutas para los dos roles principales de la aplicación: **Cliente** y **Restaurante**.

El proyecto se organiza en diferentes carpetas (Ilustración 24), cada una con una responsabilidad clara:

- **app/**: carpeta central que define la navegación de la aplicación. Contiene subcarpetas para los distintos módulos de la aplicación:
    - auth/: pantallas de autenticación, como login, registro y recuperación de contraseña.
    - cliente/: pantallas específicas para el rol cliente, como listado de restaurantes, detalle de carta o salas de pago.
    - restaurante/: pantallas específicas para el rol restaurante, como gestión de carta, subida de imágenes o visualización de salas activas.

  Además, dentro de app/ se incluyen archivos como _layout.jsx o not_found.jsx.

- **components/**: contiene componentes reutilizables de la interfaz, como tarjetas de restaurante, vistas de plato o barras de acción. De esta forma, se evita duplicar código y se mantiene una experiencia de usuario coherente en toda la aplicación.
- **lib/**: agrupa la lógica de comunicación con el backend, centralizada en diferentes APIs (por ejemplo, RestauranteAPI, FlujoPagoAPI). Gracias a esto, las llamadas HTTP quedan desacopladas de la capa de presentación.

Ilustración 24. Carpetas del Frontend

- **theme/**: define los estilos globales de la aplicación mediante *tokens* de color, tipografía y espaciado. En este proyecto se ha utilizado la librería **React Native Paper**, que facilita la creación de componentes visuales consistentes con soporte para temas claros y oscuros.

- **assets/**: carpeta que almacena recursos estáticos como imágenes e iconos que se utilizan en la interfaz.

- **constants/**: contiene constantes reutilizables de la aplicación, como valores de configuración o textos comunes.

- **dist/**: carpeta generada en el proceso de compilación, donde se almacena el código preparado para distribución.

En cuanto a librerías, las principales dependencias del proyecto declaradas en package.json son:

- **expo-router**: enrutado basado en carpetas, núcleo de la navegación.
- **react-native-paper**: biblioteca de componentes UI que respeta Material Design.
- **@tanstack/react-query**: para la gestión eficiente de datos asíncronos y caché de consultas.
- **axios**: cliente HTTP para llamadas al backend.
- **react-native-maps**: integración de mapas para mostrar restaurantes y ubicaciones.
- **expo-image-picker** y **expo-clipboard**: utilidades para seleccionar imágenes y gestionar texto en el portapapeles.
- **expo-secure-store**: almacenamiento seguro de datos sensibles como tokens de sesión.

Gracias a esta organización modular, el frontend se mantiene **escalable, reutilizable y fácil de mantener**, asegurando que cada rol (cliente o restaurante) tenga una experiencia de usuario diferenciada y coherente.

6.2 Interfaz de Usuario

Antes de acceder a las funcionalidades de la aplicación, los usuarios deben identificarse mediante un sistema de autenticación propio, implementado en el frontend con React Native y conectado al backend en Spring Boot. Este proceso asegura que solo usuarios registrados puedan acceder a la plataforma, diferenciando además entre los roles disponibles: **Cliente** y **Restaurante**.

El flujo de autenticación incluye las siguientes pantallas principales:

Pantalla de inicio (Ilustración 25)

Al abrir la aplicación, el usuario es recibido con la pantalla inicial, donde se muestra el logotipo de la aplicación junto con las opciones *Iniciar sesión* y *Registrarse*. Esta vista actúa como punto de entrada al sistema.

Ilustración 25. Pantalla de inicio

Registro de usuario (Ilustración 26)

En la pantalla de registro, el usuario introduce sus datos básicos (nombre, correo electrónico y contraseña). Un aspecto clave de esta vista es la selección del **rol** con el que el usuario se dará de alta, pudiendo elegir entre Cliente o Restaurante. Esta elección condiciona la experiencia de uso posterior dentro de la aplicación, ya que determinará a qué interfaz tendrá acceso.

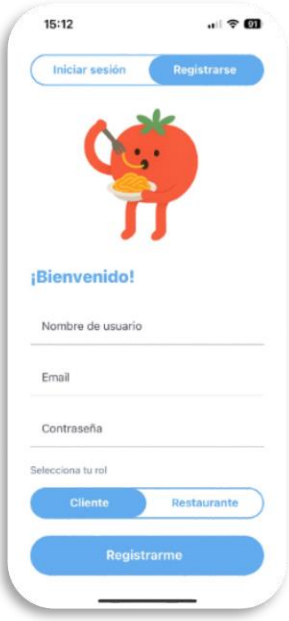


Ilustración 26. Pantalla de registro de usuario

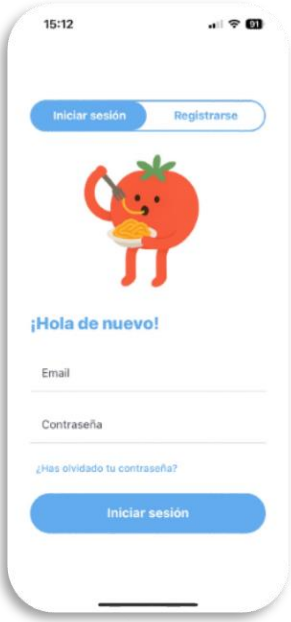


Ilustración 27. Pantalla de inicio de sesión

Inicio de sesión (Ilustración 27)

Los usuarios previamente registrados pueden acceder con su correo electrónico y contraseña. La vista incluye un acceso directo al flujo de recuperación de credenciales en caso de olvido.

Recuperación de contraseña (Ilustraciones 28 y 29)

El proceso de recuperación se compone de dos fases:

1. Introducción del correo electrónico asociado a la cuenta para solicitar un código de verificación.
2. Introducción del código recibido y definición de una nueva contraseña.

   Este mecanismo garantiza que únicamente el propietario del correo electrónico pueda restablecer el acceso a la cuenta.

Ilustración 29. Pantalla de recuperación de contraseña

Ilustración 28. Pantalla de confirmación de recuperación de contraseña

En conjunto, el flujo de autenticación proporciona una experiencia sencilla y segura, permitiendo a cada usuario acceder a la aplicación de acuerdo con su rol específico.

6.2.1 Interfaz del Restaurante

El rol de **Restaurante** está diseñado para que los dueños o gestores puedan administrar fácilmente la información de su establecimiento, así como gestionar la carta de platos y las salas de pago abiertas.

Ilustración 31. Pantalla inicial de restaurante vacío.

Ilustración 30. Pantalla de edición de la información del restaurante

Pantalla inicial del restaurante (Ilustración 31)

Una vez que un usuario se registra con el rol de restaurante, la aplicación le muestra la vista inicial de su perfil de establecimiento. En ella aparece el nombre registrado, un estado de apertura/cierre, y la posibilidad de añadir una imagen de portada y una descripción. También se incluye un bloque con el horario del restaurante, inicialmente vacío, que el usuario puede editar más adelante. En la parte inferior, se presentan accesos directos para editar la información, escanear una carta o visualizarla.

Edición de la información del restaurante (Ilustración 30)

Desde la vista inicial, el restaurante puede pulsar en *Editar info restaurante* para acceder a un formulario más completo. Aquí es posible añadir una imagen desde la cámara o galería, introducir una descripción más detallada y registrar la **dirección del establecimiento**. Además, se puede configurar el horario de apertura y cierre por cada día de la semana. Esta funcionalidad permite que el perfil del restaurante quede totalmente personalizado y que la información mostrada a los clientes sea precisa y actualizada.

Ilustración 32. Pantalla de gestión de horarios

Ilustración 33. Pantalla de información de restaurante completa

Gestión de horarios (Ilustración 33)

En el mismo formulario de edición de la información, el restaurante puede configurar sus horarios de apertura y cierre de forma detallada. La interfaz permite añadir múltiples franjas horarias por día, así como editar o eliminar las existentes.
Una vez guardados los cambios, la vista del restaurante se actualiza mostrando los horarios consolidados para cada día de la semana, de manera clara y accesible para los clientes.

Vista con información cargada (Ilustración 32)

Una vez guardados los cambios, la pantalla principal del restaurante refleja toda la información introducida: imagen de portada, descripción, dirección y horarios completos. Esta vista constituye el punto de entrada al resto de funcionalidades del rol de restaurante, proporcionando un resumen visual del perfil y manteniendo accesibles las opciones de edición, escaneo y visualización de la carta.

Escaneo de carta (Ilustraciones 34 y 35)

Además de poder editar manualmente los datos de su establecimiento, el rol de restaurante cuenta con la funcionalidad de **digitalizar su carta a partir de imágenes**.

Al seleccionar la opción *Escanear carta* desde la pantalla principal, se accede a una vista específica donde se invita al usuario a subir fotografías claras de su carta (Ilustración 34 Izda.). Para ello, se ofrecen dos opciones:

- **Galería**: seleccionar una imagen ya existente en el dispositivo.
- **Cámara**: tomar una nueva fotografía en el momento.

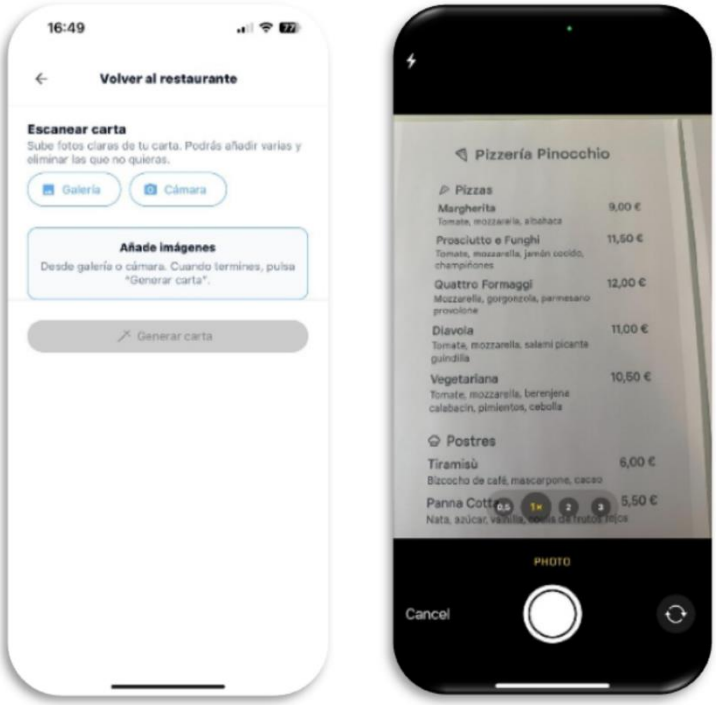


Ilustración 34. (Izda.)Pantalla de carga de imágenes para escaneo de carta. (Dcha.) Pantalla de uso de la cámara para escaneo de carta.

Las imágenes seleccionadas quedan listadas en esta vista, pudiendo eliminar aquellas que no sean válidas. Una vez añadidas, el usuario puede pulsar en *Generar carta*, lo que envía las imágenes al backend. Allí, mediante la integración con **Google Cloud Vision OCR**, se reconoce el texto, y un parser propio estructura los datos en secciones, nombres de platos, descripciones y precios.

De este modo, la aplicación permite a los restaurantes importar de manera semiautomática sus cartas físicas al sistema digital, agilizando el proceso de alta y minimizando el esfuerzo manual.

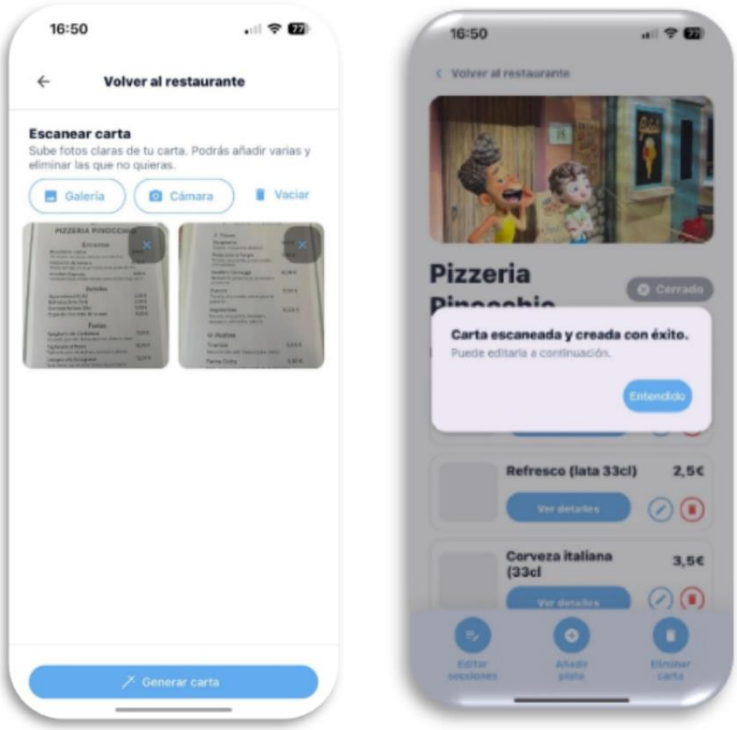


Ilustración 35. (Izda.)Pantalla de imágenes de carta cargadas . (Dcha.) Pantalla de confirmación de carta correctamente escaneada.

Una vez este termine se le redirigirá automáticamente a la pantalla de edición de carta.

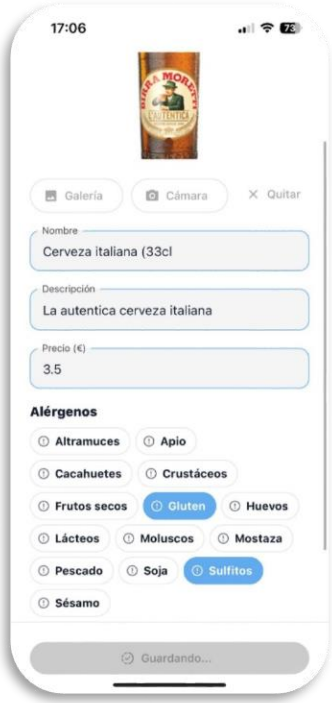


Ilustración 36. Pantalla de creación o edición de plato

Vista de creación/edición de plato (Ilustración 36)

En esta pantalla el restaurante puede dar de alta o modificar un plato de su carta digital. Se incluyen los campos principales de gestión: imagen del producto (con opción de subir desde galería o cámara), nombre, descripción y precio. Además, se ofrece la posibilidad de seleccionar de manera interactiva los alérgenos correspondientes mediante una lista de opciones predefinidas en formato de chips. Esta vista permite al restaurante mantener actualizada la información de cada plato de forma clara y estructurada, garantizando tanto la presentación visual como el cumplimiento de requisitos de información alimentaria.

Vista de gestión de carta (Ilustración 37)

Esta pantalla muestra la carta del restaurante organizada por secciones (entrantes, pastas, bebidas, etc.), con cada plato representado mediante una tarjeta que incluye imagen, nombre, breve descripción y precio. Desde esta vista, el restaurante dispone de accesos directos para **ver detalles, editar o eliminar**

cada plato. En la parte inferior, se ofrecen las acciones principales de gestión: **editar secciones**, **añadir un nuevo plato** o **eliminar la carta completa**. De este modo, se proporciona una herramienta centralizada e intuitiva para mantener la carta actualizada y perfectamente estructurada.

Ilustración 37. Pantalla de gestión de carta.

Ilustración 38. Pantalla de gestión de secciones.

Vista de edición de secciones (Ilustración 38)

En esta pantalla el restaurante puede gestionar la organización de su carta dividiendo los platos en distintas secciones (por ejemplo, bebidas, entrantes, pastas, etc.). Cada sección muestra el número total de platos que contiene y un listado resumido de sus nombres. Para cada una se ofrecen opciones de **renombrar la sección, mover platos entre secciones o eliminarla por completo**.

Además, en la parte superior se encuentra la acción de **crear una nueva sección**, lo que permite adaptar la estructura de la carta a las necesidades del restaurante en todo momento.

6.2.2 Interfaz del Comensal

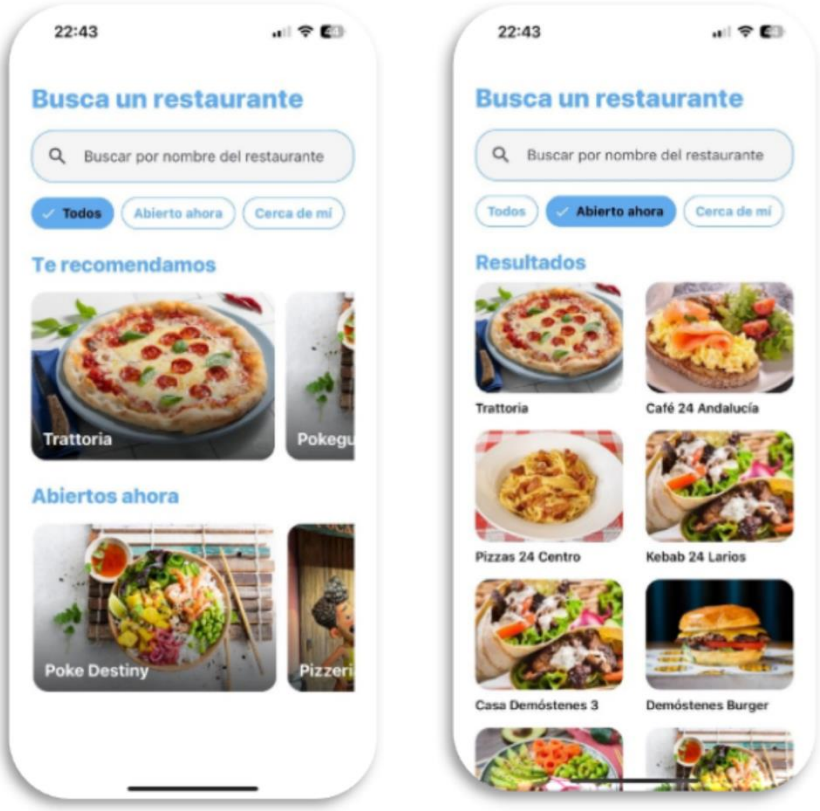


Ilustración 39. (Izda.)Pantalla de exploración de restaurante. (Dcha.) Pantalla de resultados de filtro "abierto ahora"

Vista de exploración de restaurantes (Ilustración 39 Izda.)

En esta pantalla el cliente puede buscar restaurantes a través de un buscador por nombre y mediante filtros rápidos: todos, abiertos ahora y cerca de mí. Además, se muestran secciones destacadas como Te recomendamos o Abiertos ahora, donde aparecen tarjetas visuales con el nombre y la imagen representativa de cada restaurante. Esta vista constituye el punto de entrada al descubrimiento de la oferta disponible en la aplicación.

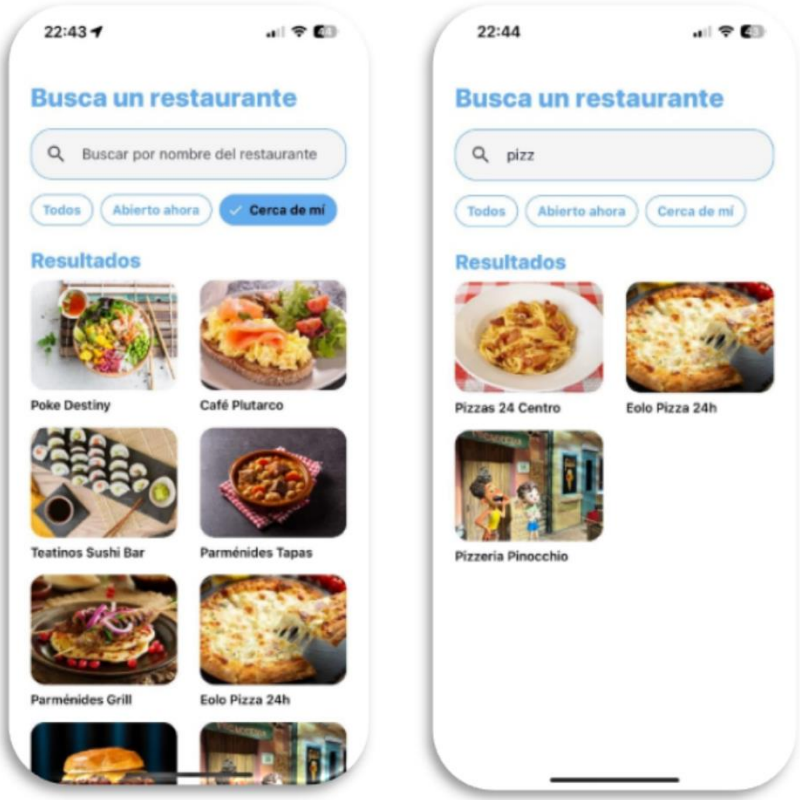


Ilustración 40. (Izda.)Pantalla de resultados de filtro "cerca de mí". (Dcha.) Pantalla de resultados de filtro por nombre

Vista de resultados filtrados (Ilustraciones 39 Dcha. y 40)

Al aplicar un filtro, el cliente accede a un listado de restaurantes que cumplen con el criterio seleccionado (por ejemplo, abiertos en el momento de la consulta). Los resultados se presentan en tarjetas que combinan imagen y nombre del restaurante, facilitando la comparación y selección. De esta forma, el usuario puede localizar rápidamente restaurantes que se ajusten a sus necesidades inmediatas.

Ilustración 41. Pantalla de detalles del restaurante

Vista de detalle de restaurante (Ilustración 41)

En esta pantalla el cliente accede a la información completa de un restaurante concreto. Se muestra la imagen de portada junto con el nombre, descripción breve y estado de apertura en tiempo real. A continuación, se incluye el **horario semanal de apertura**, especificando los días y franjas horarias en las que el establecimiento permanece operativo. Finalmente, se ofrece un mapa interactivo con la ubicación exacta del restaurante, lo que facilita al cliente la localización física del mismo.

Desplazándonos hacia abajo en esta pantalla encontramos el botón de "Ver carta" que nos permite consultar la carta del restaurante.

Esta vista constituye el punto de partida para consultar la carta y el resto de funcionalidades asociadas al restaurante seleccionado.

Vista de carta del restaurante (Ilustración 42)

En esta pantalla el cliente puede consultar la carta completa del restaurante, organizada por secciones (entrantes, principales, etc.). Cada plato se muestra con imagen, nombre, breve descripción y precio, además de la opción de ver detalles o añadirlo al pedido. En la parte inferior se encuentra la barra de acciones, que permite **crear una sala de pago, unirse a una sala existente o acceder a los detalles del pedido**.

Ilustración 42. Pantalla de carta del restaurante

Vista de creación y unión a sala de pago (Ilustración 43)

Al seleccionar la opción de crear una sala, la aplicación genera un **código único** que puede compartirse con otros comensales para que se unan a la misma mesa. Una vez dentro de una sala de pago, el botón de **detalles del pedido** se activa como acceso directo principal, mientras que el resto de opciones se deshabilitan. De esta forma, se centraliza la gestión del pedido compartido y se impide volver atrás al listado de restaurantes hasta que la sala sea cerrada, garantizando la coherencia del flujo de pago.

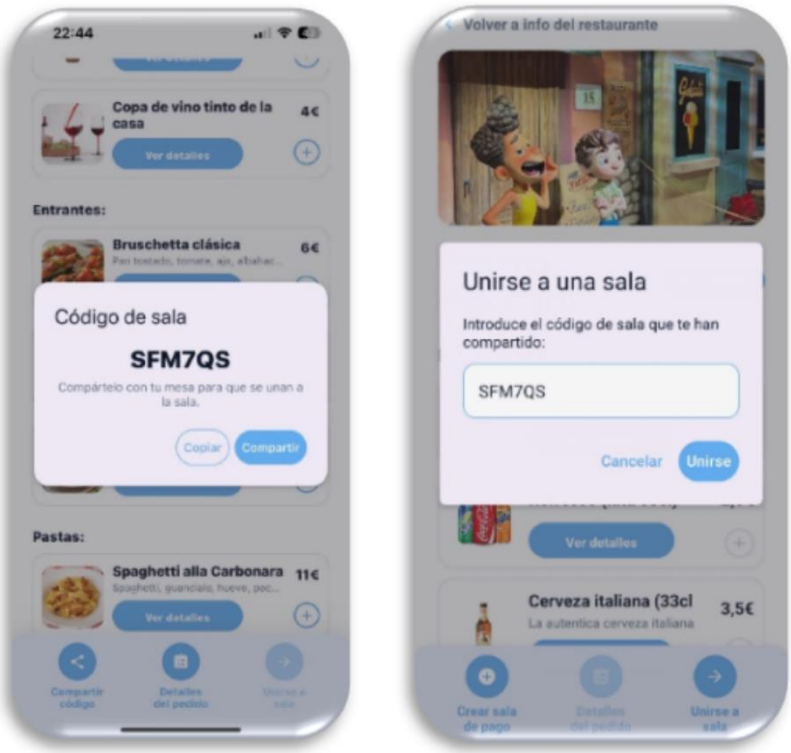


Ilustración 43. (Izda.) Pantalla de creación de sala de pago. (Dcha.) Pantalla de código para unirse a la sala de pago.

Ilustración 44. Pantalla de detalles de un plato

Vista de detalles de un plato (Ilustración 44)

En esta pantalla el cliente puede consultar de forma individual la información completa de un plato específico de la carta. Se muestra la imagen en primer plano junto al nombre, precio y una breve descripción de sus ingredientes principales. Además, se destacan los **alérgenos asociados** mediante iconos visuales (por ejemplo, gluten, huevos, lácteos), lo que garantiza que el usuario disponga de la información necesaria antes de realizar su pedido. Esta vista complementa la experiencia de navegación por la carta, ofreciendo un nivel de detalle más profundo y facilitando la toma de decisiones de forma informada.

Ilustración 45. Pantalla de asignación de comensales a un plato

Vista de asignación de comensales a un plato (Ilustración 45)

Al pulsar el botón **"+"** en cualquier plato de la carta, aparece una ventana emergente que permite seleccionar qué comensales de la sala de pago van a participar en ese plato. El cliente puede marcar uno o varios usuarios de la mesa y confirmar la acción mediante el botón **"Añadir al pedido"**, quedando registrado el reparto inicial del plato. Esta asignación puede modificarse más adelante desde los detalles del pedido, facilitando la **gestión flexible y colaborativa del consumo compartido**.

Vista de detalles del pedido (Ilustración 46 Izda.)

Esta pantalla muestra un resumen del pedido compartido dentro de una sala de pago activa. En la parte superior se identifican el nombre del restaurante, el número y el código de la sala, junto con la lista de participantes. A continuación, se presentan los platos añadidos al pedido, cada uno con su imagen, nombre, precio y número de comensales asignados.

Desde esta vista, los usuarios pueden **modificar los participantes de cada plato o eliminarlo del pedido** mediante las acciones disponibles.
Finalmente, se ofrece la opción de **generar instrucciones de pago**, lo que permite calcular y distribuir los importes entre los participantes, garantizando una gestión transparente y organizada del gasto compartido.

Vista de selección de opciones de pago (Ilustración 46 Dcha.)

Al finalizar la configuración del pedido compartido, la aplicación solicita a los comensales elegir cómo desean dividir la cuenta. Se ofrecen dos modalidades:

- **Pago personalizado**: cada participante paga únicamente la parte correspondiente a los platos en los que ha participado.
- **Pago igualitario**: el importe total de la cuenta se reparte a partes iguales entre todos los integrantes de la sala.

Una vez seleccionada la opción deseada, se generan y envían las **instrucciones de pago** a todos los participantes, tras lo cual la sala queda cerrada. Esta funcionalidad asegura una gestión flexible y justa del reparto económico entre los comensales.

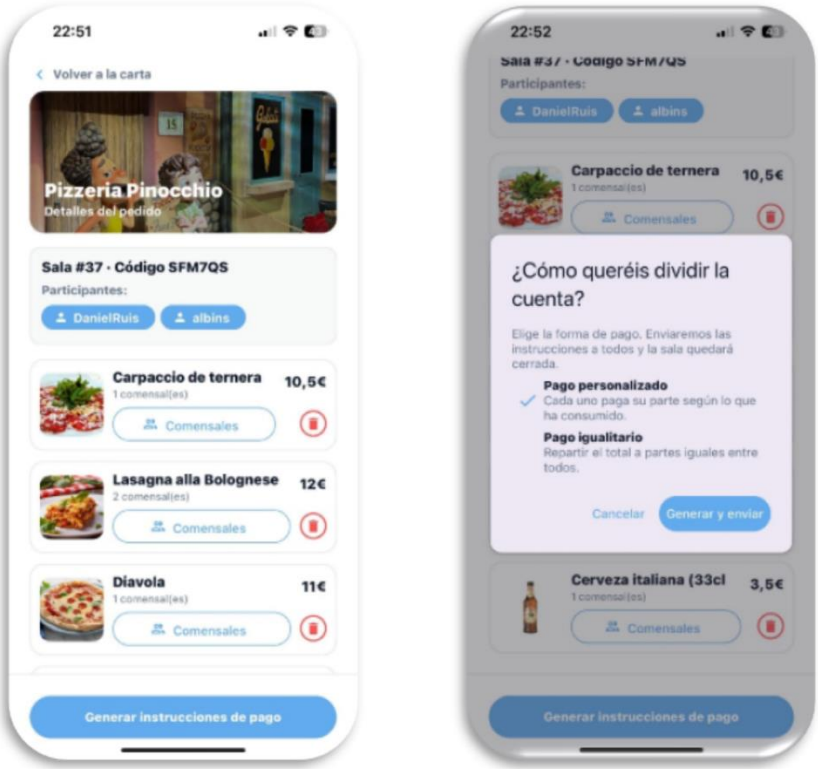


Ilustración 46. (Izda.) Pantalla de detalles del pedido. (Dcha.) Pantalla de selección de opciones de pago.

Ilustración 47. Pantalla de instrucciones de pago

Vista de instrucciones de pago (Ilustración 47)

Una vez seleccionada la forma de dividir la cuenta y confirmada la acción, todos los integrantes de la sala son redirigidos automáticamente a esta pantalla. Aquí se muestra de manera individualizada **la parte correspondiente a cada comensal**, indicando el importe total que debe abonar junto con el desglose de los platos asignados.
De esta forma, cada usuario recibe sus propias **instrucciones de pago personalizadas**, garantizando claridad y evitando confusiones en el reparto. Esta vista marca el cierre de la sala y constituye el paso final en el flujo de división de gastos.

Confirmación de pago por correo (Ilustración 48)

Además de mostrar las instrucciones en la propia aplicación, el sistema envía automáticamente un **correo electrónico personalizado a cada participante** con el desglose de su parte del pedido. En el mensaje se incluyen los datos principales: restaurante, fecha, modo de pago seleccionado, los platos consumidos con su precio correspondiente y el **total a pagar por el comensal**.
De esta manera, los usuarios disponen de un **respaldo adicional y accesible** de su gasto, lo que facilita tanto la transparencia del proceso como la consulta posterior fuera de la aplicación.

En conjunto, el flujo del cliente ofrece una experiencia completa que abarca desde la exploración inicial hasta el cierre del pago, combinando **usabilidad, claridad en la información y una gestión eficiente del gasto compartido**.

Ilustración 48. Ejemplo de confirmación de pago por correo

7

Despliegue y pruebas

7.1 Despliegue del Frontend

Durante la fase de despliegue del frontend se valoraron diferentes alternativas para la publicación de la aplicación en dispositivos móviles. En el caso de **iOS**, la distribución a través de la App Store requiere una cuenta de desarrollador de Apple con coste anual, lo que supone una barrera económica añadida. Por ello, esta vía fue descartada en el contexto del proyecto.

En **Android**, el proceso resulta más accesible y flexible. Además de la posibilidad de publicar en Google Play, se ofrece la opción de generar directamente un archivo **APK** que puede instalarse en cualquier dispositivo de forma local. Esta alternativa fue la elegida, ya que permite probar y distribuir la aplicación de manera rápida sin depender de procesos de validación externos.

En consecuencia, el frontend se encuentra empaquetado en un APK funcional, lo que facilita su instalación y evaluación en dispositivos Android.

7.2 Despliegue del Backend y Base de Datos

Para el despliegue del backend y la base de datos se optó por la plataforma **Railway**, que ofrece un entorno en la nube sencillo de configurar y mantener. El backend, desarrollado en Spring Boot, fue desplegado en un servicio de Railway, mientras que la base de datos MySQL también se aloja en la misma plataforma, garantizando así la disponibilidad de los datos y la conectividad con la aplicación móvil.

Railway resultó ser una opción adecuada por su facilidad de uso, integración con repositorios externos y la posibilidad de gestionar tanto el backend como la base de datos desde un mismo panel. De esta forma, el despliegue queda centralizado y simplificado, permitiendo dedicar más esfuerzo al desarrollo funcional del sistema en lugar de a la configuración de infraestructura compleja.

7.3 Casos de Prueba

En esta sección se presentan los principales casos de prueba diseñados para validar el correcto funcionamiento de la aplicación. Cada caso se ha elaborado a partir de los casos de uso definidos previamente, cubriendo tanto las funcionalidades del rol restaurante como las del rol cliente, así como el flujo completo de registro, gestión de carta y proceso de pago compartido.

Tabla 18. Caso de prueba 01

Titulo	**Registro de cliente válido**
Objetivo	Verificar que un cliente puede registrarse correctamente.
Precondiciones	No existe un usuario con ese correo.
Datos de prueba	Correo válido, nombre, contraseña fuerte, rol=cliente.
Pasos	1. Abrir aplicación. 2. Seleccionar “Registrarse”. 3. Introducir correo, nombre, contraseña y rol cliente. 4. Pulsar “Registrarse”.
Resultados esperados	Se crea el usuario cliente, se muestra mensaje de éxito, se inicia sesión automáticamente y se redirige a la pantalla principal de cliente.

Tabla 19. Caso de prueba 02

Titulo	**Registro de restaurante válido**
Objetivo	Verificar que un restaurante puede registrarse.
Precondiciones	No existe un usuario con ese correo.
Datos de prueba	Correo válido, contraseña, nombre del establecimiento, rol=restaurante.
Pasos	1. Abrir aplicación. 2. Seleccionar “Registrarse”. 3. Introducir datos y rol restaurante. 4. Pulsar “Registrarse”.
Resultados esperados	Se crea el usuario restaurante, aparece mensaje de éxito, se inicia sesión automáticamente y se redirige a la pantalla principal del restaurante.

Tabla 20. Caso de prueba 03

Titulo	**Inicio de sesión correcto**
Objetivo	Validar que un usuario accede con credenciales válidas.

Precondiciones	Usuario registrado en el sistema.
Datos de prueba	Email y contraseña correctos.
Pasos	1. Abrir aplicación. 2. Pulsar “Iniciar sesión”. 3. Introducir credenciales válidas. 4. Pulsar “Iniciar sesión”.
Resultados esperados	El usuario inicia sesión y accede a la pantalla principal de su rol.

Tabla 21. Caso de prueba 04

Titulo	**Escaneo de carta física**
Objetivo	Digitalizar carta a partir de fotos mediante OCR.
Precondiciones	Restaurante ha iniciado sesión correctamente en la aplicación.
Datos de prueba	2 imágenes legibles de carta.
Pasos	1. Pulsar “Escanear carta”. 2. Seleccionar “Usar cámara”. 3. Capturar imágenes de las páginas. 4. Pulsar “Digitalizar carta”.
Resultados esperados	El sistema procesa las imágenes, genera platos y muestra la vista de edición.

Tabla 22. Caso de prueba 05

Titulo	**Edición de un plato**
Objetivo	Modificar información de un plato existente.
Precondiciones	Carta digitalizada con al menos un plato.
Datos de prueba	Nuevo nombre, precio y alérgenos.
Pasos	1. Abrir lista de platos. 2. Seleccionar un plato. 3. Editar los campos necesarios. 4. Guardar cambios.
Resultados esperados	El sistema actualiza el plato y vuelve a la lista general.

Tabla 23. Caso de prueba 06

Titulo	**Subida de imagen a plato**
Objetivo	Asociar imagen a un plato desde cámara o galería.
Precondiciones	Restaurante en pantalla de edición/creación de plato.
Datos de prueba	Foto válida en galería o tomada con cámara.
Pasos	1. Pulsar “Añadir imagen”.

	2. Seleccionar cámara o galería. 3. Confirmar selección.
Resultados esperados	El sistema asocia la imagen al plato y la muestra en el formulario.

Tabla 24. Caso de prueba 07

Titulo	**Organización de carta en secciones**
Objetivo	Reestructurar la carta creando y moviendo secciones.
Precondiciones	Carta con varios platos existentes.
Datos de prueba	Nombre válido de sección.
Pasos	1. Abrir editor de secciones. 2. Crear nueva sección y asignar platos. 3. Mover un plato de una sección a otra. 4. Guardar organización.
Resultados esperados	El sistema refleja la nueva organización de la carta.

Tabla 25. Caso de prueba 08

Titulo	**Gestión de establecimiento**
Objetivo	Editar información pública del restaurante.
Precondiciones	Restaurante ha iniciado sesión.
Datos de prueba	Nueva dirección, horario válido y foto de portada.
Pasos	1. Pulsar “Editar establecimiento”. 2. Modificar los campos deseados. 3. Guardar cambios.
Resultados esperados	La información se actualiza y aparece en el perfil público del restaurante.

Tabla 26. Caso de prueba 09

Titulo	**Previsualización de carta**
Objetivo	Visualizar carta como la vería un cliente.
Precondiciones	Carta digitalizada y con platos.
Datos de prueba	-
Pasos	1. Pulsar “Previsualizar carta”. 2. Navegar por secciones y platos. 3. Salir de la previsualización.
Resultados esperados	Se muestra la carta en modo solo lectura y se retorna al editor sin cambios.

Tabla 27. Caso de prueba 10

Titulo	**Búsqueda de restaurante**
Objetivo	Localizar restaurante por nombre o filtro.

Precondiciones	Cliente ha iniciado sesión.
Datos de prueba	Nombre parcial de restaurante.
Pasos	2. Introducir texto de búsqueda. 1. Pulsar “Buscar”.
Resultados esperados	El sistema muestra lista de restaurantes coincidentes.

Tabla 28. Caso de prueba 11

Titulo	**Detalle de restaurante**
Objetivo	Consultar información pública del restaurante.
Precondiciones	Cliente con listado de restaurantes visible.
Datos de prueba	Selección de un restaurante válido.
Pasos	1. Seleccionar restaurante en la lista. 2. Abrir su perfil público.
Resultados esperados	Se muestran descripción, horarios y ubicación en mapa, junto al acceso a la carta.

Tabla 29. Caso de prueba 12

Titulo	**Consulta de carta**
Objetivo	Ver carta organizada y detalle de platos.
Precondiciones	Restaurante con carta publicada.
Datos de prueba	Plato con descripción e imagen.
Pasos	1. Pulsar “Ver carta”. 2. Navegar entre secciones. 3. Seleccionar un plato.
Resultados esperados	Se muestran nombre, precio, descripción, alérgenos e imagen.

Tabla 30. Caso de prueba 13

Titulo	**Crear sala de pago**
Objetivo	Iniciar pedido compartido.
Precondiciones	Cliente en carta de restaurante.
Datos de prueba	-
Pasos	1. Pulsar “Crear sala de pago”. 2. Confirmar creación.
Resultados esperados	Se genera código único de sala y se activa barra con “Detalles del pedido”

Tabla 31. Caso de prueba 14

Titulo	**Añadir plato a sala**
Objetivo	Registrar un plato con comensales asociados.
Precondiciones	Sala de pago activa con participantes.

Datos de prueba	Selección de 2 comensales para un plato.
Pasos	1. Pulsar “+” en un plato. 2. Seleccionar comensales. 3. Confirmar
Resultados esperados	El plato queda registrado en la sala y visible para todos los participantes.

Tabla 32. Caso de prueba 15

Titulo	**Cálculo y envío de división de cuenta**
Objetivo	Generar instrucciones de pago individuales.
Precondiciones	Sala activa con consumiciones registradas.
Datos de prueba	Opción “Pago personalizado” e “Igualitario”.
Pasos	1. Pulsar “Calcular cuenta”. 2. Seleccionar método de pago. 3. Confirmar.
Resultados esperados	Cada usuario recibe en pantalla sus instrucciones de pago personalizadas y además un correo con el desglose de su parte.

La ejecución de estos casos de prueba garantiza que las funcionalidades clave del sistema cumplen con los requisitos planteados y que la aplicación ofrece una experiencia fiable y coherente para ambos tipos de usuarios.

7.4 Colección de Postman

Con el fin de facilitar la validación del backend, se entrega junto a la memoria una **colección de Postman** que incluye los distintos endpoints implementados.

Esta colección permite ejecutar pruebas de manera directa sobre la API, verificando operaciones como la autenticación de usuarios, la gestión de restaurantes y cartas, así como la creación y uso de salas de pago compartido.

De esta forma, Postman se convierte en una herramienta complementaria a los casos de prueba descritos previamente, proporcionando un medio práctico para comprobar el correcto funcionamiento del sistema y simplificando futuras tareas de verificación o mantenimiento.

8

Conclusiones y Líneas Futuras

8.1 Conclusiones y objetivos cumplidos

El desarrollo de **CartaGo** ha permitido cumplir con los objetivos principales planteados al inicio del proyecto: digitalizar la carta de los restaurantes, ofrecer una interfaz clara y accesible para su gestión, y proporcionar a los clientes una experiencia sencilla para consultar menús, organizar pedidos compartidos y dividir pagos de manera justa.

Se ha logrado implementar una aplicación móvil funcional, con backend y base de datos desplegados en la nube, integrando características como escaneo mediante OCR, edición de platos con alérgenos, organización de cartas y un sistema de salas de pago con reparto automático. Todo ello valida la viabilidad técnica y práctica de la propuesta.

8.2 Dificultades Encontradas

A lo largo del desarrollo surgieron varias dificultades que condicionaron el avance del proyecto:

- **Complejidad en el despliegue del frontend**, ya que, aunque la generación de un APK en Android fue relativamente sencilla, la publicación oficial en iOS resultó inviable por los requisitos y costes de la App Store.
- **Procesamiento de resultados del OCR**, puesto que la API de Google Cloud Vision devolvía la carta como un bloque de texto continuo. Fue necesario diseñar un proceso de parseo capaz de identificar qué fragmentos correspondían a platos, descripciones, precios o secciones, para convertirlos en una estructura usable dentro de la aplicación.
- **Bloqueos y limitaciones de APIs externas en producción**, que dificultaron la estabilidad de algunas funcionalidades del backend y obligaron a plantear alternativas o gestionar excepciones.

- **Restricciones al probar en Android con Expo Go**, ya que ciertas funcionalidades, como la integración de mapas, no pueden visualizarse o validarse correctamente sin generar un build completo de la aplicación.

A pesar de estos obstáculos, se adoptaron soluciones prácticas para mantener la continuidad del desarrollo, priorizando la funcionalidad esencial frente a características secundarias.

8.3 Posibles Ampliaciones

El proyecto abre la puerta a futuras mejoras que podrían enriquecer su alcance:

- Publicación oficial en **Google Play y App Store**, una vez solventados los requisitos de distribución.
- Inclusión de **métodos de pago integrados** en la propia aplicación (Bizum, PayPal, Stripe), eliminando el paso manual de transferencias.
- Incorporación de **estadísticas y analítica** para restaurantes (platos más vendidos, horas de mayor actividad, etc.).
- Mejora de la experiencia del cliente con **sistemas de valoración y reseñas** de restaurantes.

Bibliografía

Block, Inc. (2025). *Square for Restaurants POS*. Square. https://squareup.com/us/en/point-of-sale/restaurants

Expo. (2025). *Expo documentation*. Expo. https://docs.expo.dev/

Expo. (2025). *Expo Secure Store*. Expo. https://docs.expo.dev/versions/latest/sdk/securestore/

Expo. (2025). *Expo Location*. Expo. https://docs.expo.dev/versions/latest/sdk/location/

Expo. (2025). *Expo Image Picker*. Expo. https://docs.expo.dev/versions/latest/sdk/imagepicker/

Expo. (2025). *Expo Clipboard*. Expo. https://docs.expo.dev/versions/latest/sdk/clipboard/

Expo. (2025). *Expo Linear Gradient*. Expo. https://docs.expo.dev/versions/latest/sdk/linear-gradient/

GlovoApp23, S.L. (2025). *Glovo – Food and more, delivered*. Glovo. https://glovoapp.com/

Google Cloud. (2025). *Cloud Vision API documentation*. Google Cloud. https://cloud.google.com/vision

Just Eat Takeaway.com N.V. (2025). *Just Eat – Order food online*. Just Eat. https://www.justeat.com/

Lombok Project. (2025). *Project Lombok*. https://projectlombok.org/

MyWaiter. (2025). *MyWaiter – Digital ordering solution*. MyWaiter. https://mywaiterapp.com/

OpenJS Foundation. (2025). *Axios HTTP client*. Axios. https://axios-http.com/

React Navigation. (2025). *React Navigation documentation*. https://reactnavigation.org/

React Query. (2025). *TanStack Query*. https://tanstack.com/query/latest

React Native Community. (2025). *React Native Maps*. https://github.com/react-native-maps/react-native-maps

React Native Paper. (2025). *React Native Paper components*. https://callstack.github.io/react-native-paper/

React Native Reanimated. (2025). *React Native Reanimated documentation*. https://docs.swmansion.com/react-native-reanimated/

React Native Safe Area Context. (2025). *Safe area handling in React Native*. https://github.com/th3rdwave/react-native-safe-area-context

React Native Screens. (2025). *Native navigation optimizations*. https://github.com/software-mansion/react-native-screens

React Native Training. (2025). *Keyboard Aware ScrollView*. https://github.com/APSL/react-native-keyboard-aware-scroll-view

TheFork. (2025). *TheFork – Online restaurant reservations*. TheFork. https://www.thefork.com/

Toast, Inc. (2025). *Toast POS – Point of sale system*. Toast. https://pos.toasttab.com/

TouchBistro, Inc. (2025). *TouchBistro POS – Restaurant management system*. TouchBistro. https://www.touchbistro.com/

Tricount. (2025). *Tricount – Expense sharing made easy*. Tricount. https://www.tricount.com/

Uber Technologies Inc. (2025). *Uber Eats – Food delivery*. Uber Eats. https://www.ubereats.com/

VMware. (2025). *Spring Boot reference documentation*. Spring. https://spring.io/projects/spring-boot

VMware. (2025). *Spring Data JPA documentation*. Spring. https://spring.io/projects/spring-data-jpa

VMware. (2025). *Spring Security reference documentation*. Spring. https://spring.io/projects/spring-security

VMware. (2025). *Spring Validation (Bean Validation)*. Spring. https://spring.io/guides/gs/validating-form-input/

VMware. (2025). *Spring Boot Actuator reference documentation*. Spring. https://docs.spring.io/spring-boot/docs/current/actuator-api/htmlsingle/

VMware. (2025). *Spring Boot Mail starter documentation*. Spring.

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#io.email

Auth0. (2025). *Java JWT library*. Auth0. https://github.com/auth0/java-jwt

Apache Software Foundation. (2025). *Apache HttpClient 5*. Apache. https://hc.apache.org/httpcomponents-client-5.0.x/

Oracle. (2025). *MySQL Connector/J documentation*. MySQL. https://dev.mysql.com/downloads/connector/j/

Meta Platforms, Inc. (2025). *React Native documentation*. React Native. https://reactnative.dev/

@expo. (2025). *@expo/vector-icons*. Expo. https://docs.expo.dev/guides/icons/

@react-native-async-storage. (2025). *@react-native-async-storage/async-storage*. GitHub. https://github.com/react-native-async-storage/async-storage

Apéndice A

Manual de Instalación

A.1 Requerimientos:

Para poder ejecutar la aplicación **CartaGo** en un dispositivo móvil es necesario contar con:

- Node.js 18+ y npm (o pnpm/yarn).
- npx disponible (viene con npm).
- Expo CLI (se usa vía npx expo start).
- Móvil y ordenador en la misma red Wi-Fi (solo en iOS con Expo Go).
- Expo Go instalado en iPhone (App Store).
- Permisos de instalación de APK en Android.
- Back y base de datos ya desplegados en Railway (no se requiere levantar en local).

A2. Instalación en el dispositivo IOS

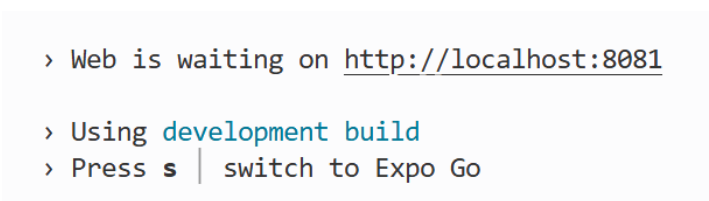
Descargar la aplicación **Expo Go** desde la App Store.
Clonar e instalar dependencias en el ordenador

```
git clone https://github.com/Alba-rg-14/CartaGoFrontend.git
cd CartaGoFrontend
npm install
```

Lanzar el frontend introduciendo este comando en la terminal

```
npx expo start -c
```

En la terminal si pulsamos “s” cambiamos entre “Development build” o “Expo Go”, hay que asegurarse de estar en “Expo Go”.



Seguidamente escaneamos el código QR que aparece en la terminal desde la cámara de nuestro dispositivo y la aplicación se abrirá en nuestra aplicación de ExpoGo.

A3. Instalación en el dispositivo Android

Para permitir la instalación de APK fuera de Play Store:

- Abrir Ajustes en tu móvil
- Ir al apartado de Apps o Aplicaciones
- Toca el menú de **3 puntos** → **Acceso especial** (o **Acceso a apps especial**).
- Entra en **Instalar apps desconocidas**.
- Elige la aplicación desde la que vas a abrir el APK (p. ej., **Archivos**, **Descargas** o **Chrome**).
- Activa **Permitir desde esta fuente**.

O bien al abrir el .apk, Android le mostrará un aviso → pulse **Ajustes** → active **Permitir desde esta fuente** → vuelva atrás y **Instalar**.

Desde su dispositivo móvil, descargue la APK desde el siguiente enlace pulsando en Install:

https://expo.dev/accounts/albaruiz/projects/cartago-app/builds/385b225f-d525-4830-b41f-272a6597d770

La aplicación estará disponible en su dispositivo.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA