IES EL GRAO
VALENCIA

RESERVFLIGHT

Ciclo Formativo de Grado Superior
DAM

2024-2025

AUTOR: Fernando Vicent Torres Serra

TUTOR: Marcos Gallach Pérez

FERNANDO VICENT TORRES SERRA

Firmado digitalmente por FERNANDO VICENT TORRES SERRA
Fecha: 2025.05.30 10:15:24 +02'00'

RESÚMENES

Resumen (Español)

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una herramienta digital funcional, intuitiva y de fácil uso que permita al usuario registrar de manera ordenada y eficaz todos sus viajes, garantizando la recopilación y conservación de la información más relevante. La aplicación propuesta busca simplificar el proceso de documentación personal de experiencias viajeras, integrando datos fundamentales como fechas, destinos y gastos. A través de un diseño centrado en la usabilidad, la herramienta permite al usuario visualizar y gestionar sus viajes de forma clara, fomentando una experiencia organizada y personalizada. Como resultado, se presenta una solución que combina funcionalidad, diseño atractivo y eficiencia en el manejo de datos, ofreciendo un recurso práctico para uso personal. Las conclusiones obtenidas reflejan el cumplimiento de los objetivos propuestos y abren la puerta a futuras mejoras que permitan ampliar las capacidades de la herramienta, adaptándola a nuevos contextos y tecnologías emergentes.

Resum (Valenciano)

L'objectiu principal d'aquest Treball de Fi de Grau és desenvolupar una eina digital funcional, intuïtiva i fàcil d'utilitzar que permet a l'usuari registrar de manera ordenada i eficaç tots els seus viatges, garantint la recopilació i conservació de la informació més rellevant. L'aplicació proposada busca simplificar el procés de documentació personal de les experiències viatgeres, integrant dades fonamentals com dates, destinacions i despeses. A través d'un disseny centrat en la usabilitat, l'eina permet a l'usuari visualitzar i gestionar els seus viatges de manera clara, fomentant una experiència organitzada i personalitzada. Com a resultat, es presenta una solució que combina funcionalitat, disseny atractiu i eficiència en la gestió de dades, oferint un recurs pràctic per a l'ús personal. Les conclusions obtingudes reflecteixen l'acompliment dels objectius proposats i obrin la porta a futures millores que permeten ampliar les capacitats de l'eina, adaptant-la a nous contextos i tecnologies emergents.

Abstract (Inglés)

The main objective of this Final Degree Project is to develop a digital tool that is functional, intuitive, and easy to use, allowing the user to record all their trips in an orderly and efficient manner, while ensuring the collection and preservation of the most relevant information. The proposed application aims to simplify the process of personal travel documentation by integrating essential data such as dates, destinations, and expenses. Through a design focused on usability, the tool enables users to clearly visualize and manage their travels, promoting an organized and personalized experience. As a result, the solution presented combines functionality, attractive design, and efficiency in data management, offering a practical resource for personal use. The conclusions reached reflect the fulfillment of the proposed objectives and open the door to future improvements that may expand the tool's capabilities, adapting it to new contexts and emerging technologies.

ÍNDICE

JUSTIFICACIÓN

La elección de este proyecto surge de una experiencia personal durante los dos últimos años en los que, por motivos académicos, residí fuera de mi vivienda habitual, lo que implicó la necesidad de realizar numerosos desplazamientos en avión. En uno de esos trayectos surgió la curiosidad por conocer cuántos vuelos había realizado y cuánto había gastado en total, constatando al mismo tiempo que la mayoría de las aerolíneas no ofrecen herramientas que permitan al usuario gestionar o consultar fácilmente esta información. Esta carencia motivó el desarrollo de una solución que cubra dicha necesidad, permitiendo a los usuarios, tanto a nivel personal como profesional, llevar un control completo y detallado de sus viajes. Aunque el proyecto está orientado a personas que viajan con cierta frecuencia, este grupo representa un público amplio y diverso, lo que refuerza su utilidad práctica. A diferencia de otras aplicaciones disponibles, esta herramienta ofrece la posibilidad de centralizar toda la información relativa a los vuelos, independientemente de la compañía aérea, permitiendo consultar gastos totales, aerolínea más utilizada, destinos más frecuentes y otros datos relevantes, aportando así un valor añadido en términos de organización y análisis personalizado de la experiencia viajera.

OBJETIVOS

El objetivo principal de esta aplicación es proporcionar a los usuarios una herramienta funcional, intuitiva y fácil de usar, que permita llevar un registro completo y ordenado de todos los viajes realizados, abarcando tanto la información básica como los datos más detallados relacionados con cada desplazamiento. La aplicación tiene como finalidad facilitar la gestión y el análisis de los viajes de manera eficiente, integrando datos relevantes tales como fechas de los vuelos, destinos, aerolíneas utilizadas y gastos totales.

Uno de los objetivos secundarios es permitir que la herramienta sea útil tanto para usuarios con fines personales como profesionales, ya que puede adaptarse a diversos perfiles de viajeros, como estudiantes, trabajadores frecuentes o personas que realizan viajes esporádicos. A través de su diseño centrado en la usabilidad, se busca que cualquier tipo de usuario pueda acceder y gestionar la información de manera clara y eficaz.

Adicionalmente, la aplicación pretende centralizar todos los datos relacionados con los vuelos de distintas aerolíneas en un solo lugar, evitando que los usuarios tengan que consultar múltiples plataformas para obtener la información deseada. Asimismo, se busca proporcionar una visión detallada del comportamiento del usuario, permitiéndole identificar patrones en sus hábitos de viaje, como la aerolínea más utilizada, el destino más frecuente o el gasto total acumulado, facilitando así una toma de decisiones más informada. En última instancia, el objetivo es ofrecer una herramienta que optimice la experiencia viajera, mejorando la organización personal y proporcionando un análisis de datos accesible y comprensible para el usuario.

PLANIFICACIÓN DE PROYECTO

Planificación

1. **Análisis de viabilidad del proyecto**
   En esta fase se evaluará la viabilidad técnica y funcional del proyecto, asegurándose de que los objetivos son alcanzables dentro de los plazos establecidos y con los recursos disponibles. Se realizarán investigaciones previas para determinar las herramientas, tecnologías y enfoques más adecuados para el desarrollo de la aplicación.

2. **Elaboración de los bocetos del diseño de la aplicación**
   Se desarrollarán los primeros prototipos y bocetos del diseño de la interfaz de usuario (UI) y experiencia de usuario (UX). Esto incluye la planificación de la estructura visual de la aplicación, las pantallas principales y cómo el usuario interactuará con la herramienta.

3. **Desarrollo de la base de datos**
   Se comenzará con el diseño de la base de datos que sustentará la aplicación.

4. **Modelo Entidad-Relación (ER)**: Se creará un modelo visual que refleje las entidades y sus relaciones.

5. **Modelo lógico**: Se estructurará la base de datos de acuerdo con los requisitos del proyecto, definiendo tablas, campos y restricciones de integridad.

6. **Modelo físico**: Se definirá la implementación real de la base de datos en un sistema de gestión PostgreSQL.

7. **Investigación de herramientas de desarrollo útiles**
   Se investigarán y seleccionarán las herramientas, lenguajes de programación y frameworks más adecuados para el desarrollo de la aplicación, considerando las necesidades del proyecto y la compatibilidad entre las distintas tecnologías.

8. **Formación sobre las herramientas seleccionadas**
   Se llevará a cabo un proceso de aprendizaje y formación sobre las herramientas elegidas para el desarrollo de la aplicación. Esto puede incluir tutoriales, documentación oficial y la realización de proyectos pequeños para familiarizarse con las tecnologías seleccionadas.

9. **Desarrollo de la aplicación**
   En esta fase se procederá con la implementación técnica de la aplicación, siguiendo el diseño previo y utilizando las herramientas de desarrollo seleccionadas. Esto incluirá la creación de la interfaz de usuario, la lógica de negocio, la integración de la base de datos y la implementación de las funcionalidades principales.

10. **Pruebas de la aplicación**
    Se realizarán pruebas rigurosas tanto durante el proceso de desarrollo como después de la implementación final de la aplicación.

11. **Pruebas de desarrollo**: Pruebas internas para comprobar que todas las funcionalidades de la aplicación se desarrollan correctamente y no contienen errores.

12. **Corrección de errores y optimización**
    Tras la fase de pruebas, se procederá a corregir los errores detectados, optimizar el rendimiento de la aplicación y realizar mejoras en la interfaz según la retroalimentación obtenida durante las pruebas.

13. **Finalización del proyecto**
    Una vez corregidos los errores y optimizada la aplicación, se procederá a la finalización del proyecto, preparando toda la documentación final y asegurando que todos los objetivos establecidos en el TFG se han cumplido.

14. **Observación de futuras mejoras**
    Después de la entrega del proyecto, se reflexionará sobre las posibles mejoras y nuevas funcionalidades que podrían implementarse en el futuro, basándose en las tendencias tecnológicas emergentes y las necesidades del usuario. Esta fase también puede incluir la planificación de versiones futuras o actualizaciones.

HERRAMIENTAS UTILIZADAS

Kotlin con Jetpack Compose

Kotlin es un lenguaje de programación moderno, utilizado para el desarrollo de aplicaciones móviles en Android, y completamente interoperable con Java, lo que facilita su integración en proyectos existentes. Jetpack Compose es un framework de UI desarrollado por Google que simplifica la creación de interfaces de usuario de manera declarativa en Kotlin. Gracias a su diseño, Jetpack Compose permite crear interfaces interactivas con menos código y mayor flexibilidad. Esta herramienta ha sido clave en el desarrollo de la interfaz de la aplicación, asegurando una experiencia de usuario fluida y accesible.

- **Licencia:** Open Source (Apache 2.0)
- **Coste:** Gratuito
- **Alternativas:** Flutter (también open source, desarrollado por Google), Java con XML (más tradicional pero menos moderno).

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, reconocido por su robustez y fiabilidad, especialmente en entornos que requieren operaciones complejas y gran volumen de datos. En este proyecto, se utiliza para almacenar información relacionada con los usuarios, sus reservas y los vuelos asociados. La base de datos se ejecuta en un servidor local, arrancado mediante Node.js, actuando como backend intermedio entre la aplicación y los datos persistentes. Para la gestión visual de la base de datos, se ha utilizado la herramienta **pgAdmin**, lo cual ha facilitado tanto la consulta como la administración de tablas y relaciones.

- **Licencia:** Open Source (PostgreSQL License, similar a MIT)
- **Coste:** Gratuito
- **Alternativas:** MySQL, SQLite (menos potente para relaciones complejas), Oracle DB (de pago).

Firebase (Auth y Crashlytics)

- **Firebase Auth** se ha integrado para facilitar la autenticación de usuarios, permitiendo el acceso mediante correo electrónico y contraseña. Esta solución proporciona una forma segura y rápida de gestionar sesiones de usuario y garantiza que cada usuario esté correctamente identificado.

- **Firebase Crashlytics** proporciona análisis detallados de fallos y errores en la aplicación. Ayuda a identificar y solucionar problemas en tiempo real, mejorando la estabilidad general del sistema.

- **Licencia:** Propietaria (Google)
- **Coste:**
    - Firebase Auth: Gratis hasta cierto número de usuarios/funciones.
    - Crashlytics: Gratuito en el plan básico.
- **Alternativas:** Auth0 (freemium, con opciones avanzadas de pago), Sentry (para crash reporting).

Retrofit

Retrofit es una librería que simplifica la comunicación entre la aplicación y las API RESTful. A través de Retrofit, la app puede hacer peticiones HTTP para interactuar con servicios externos, obtener información y enviar datos al servidor de forma confiable. Esta herramienta es esencial para la integración con APIs externas, como la de aviación.

- **Licencia:** Open Source (Apache 2.0)
- **Coste:** Gratuito
- **Alternativas:** Volley (menos moderno), OkHttp (usado internamente por Retrofit), Fuel.

Aviationstack API

Aviationstack es una API REST que proporciona datos en tiempo real sobre vuelos, aerolíneas y aeropuertos. Se emplea para obtener información actualizada sobre vuelos, como horarios, estados y rutas. Su integración permite que los vuelos mostrados en la app sean reales y actualizados, aunque con limitaciones de uso gratuitas.

- **Licencia:** Propietaria (REST API)
- **Coste:**
  - Plan Gratuito: 500 llamadas al mes con limitaciones.
  - Planes de pago disponibles desde 29,99 $/mes.
- **Alternativas:** OpenSky Network, FlightAware (más completos pero de pago).

Postman

Postman es una plataforma de desarrollo de APIs que permite realizar pruebas de peticiones HTTP de manera sencilla. Se utilizó para probar los endpoints de la Aviationstack API, así como para verificar respuestas, parámetros y estructura de datos antes de su implementación dentro de la aplicación. Ha sido fundamental durante la fase de pruebas de la integración con servicios externos.

- **Licencia:** Freemium (propietaria)
- **Coste:** Gratuito en su versión básica
- **Alternativas:** Insomnia, Hoppscotch.

Entornos de desarrollo utilizados

Android Studio: IDE principal utilizado para el desarrollo de la aplicación móvil en Kotlin. Ofrece herramientas avanzadas como el emulador de dispositivos Android, que ha sido utilizado como máquina virtual para probar la aplicación de forma continua durante el desarrollo.

- **Licencia:** Gratuita (proporcionada por Google)
- **Coste:** Gratuito
- **Alternativas:** IntelliJ IDEA (versión Community gratuita, Ultimate de pago)

Visual Studio Code + Node.js: Se ha utilizado Visual Studio Code como editor ligero para el desarrollo del servidor backend en Node.js, el cual actúa como conector entre la aplicación, la base de datos PostgreSQL y la API de vuelos.

- **Licencia:**
  - VS Code: Open Source (MIT), mantenido por Microsoft
  - Node.js: Open Source (MIT)
- **Coste:** Gratuito
- **Alternativas:** WebStorm (de pago), Eclipse + plugins

pgAdmin: Herramienta de gestión gráfica para PostgreSQL, utilizada para consultas, creación de tablas y administración general de la base de datos de forma más eficiente y visual.

- **Licencia:** Open Source (PostgreSQL Licence)
- **Coste:** Gratuito
- **Alternativas:** DBeaver, Adminer

Relación entre las herramientas

- **Kotlin + Jetpack Compose**: Desarrollo de la interfaz de usuario, con un diseño declarativo, moderno y eficiente.
- **Firebase Auth + Crashlytics**: Autenticación de usuarios y análisis de errores, garantizando seguridad y estabilidad.
- **PostgreSQL + pgAdmin**: Almacenamiento estructurado de vuelos, reservas y usuarios, administrado con pgAdmin.
- **Retrofit**: Comunicación con servicios externos como APIs REST.
- **Aviationstack API**: Proveedor de datos en tiempo real sobre vuelos, integrados en la app a través de Retrofit.
- **Postman**: Pruebas de integración con la API, verificación de respuestas y debugging previo al desarrollo.

- **Aviationstack API**: Proveedor de datos en tiempo real sobre vuelos, integrados en la app a través de Retrofit.

- **Postman**: Pruebas de integración con la API, verificación de respuestas y debugging previo al desarrollo.

- **Node.js + Visual Studio Code**: Backend local que conecta la app con la base de datos y la API externa.

- **Android Studio + emulador**: Entorno de desarrollo principal y simulador de la app para pruebas móviles.

Estas herramientas actúan de forma conjunta y sinérgica para garantizar una aplicación estable, funcional y bien estructurada, con datos en tiempo real y una experiencia de usuario moderna y completa.

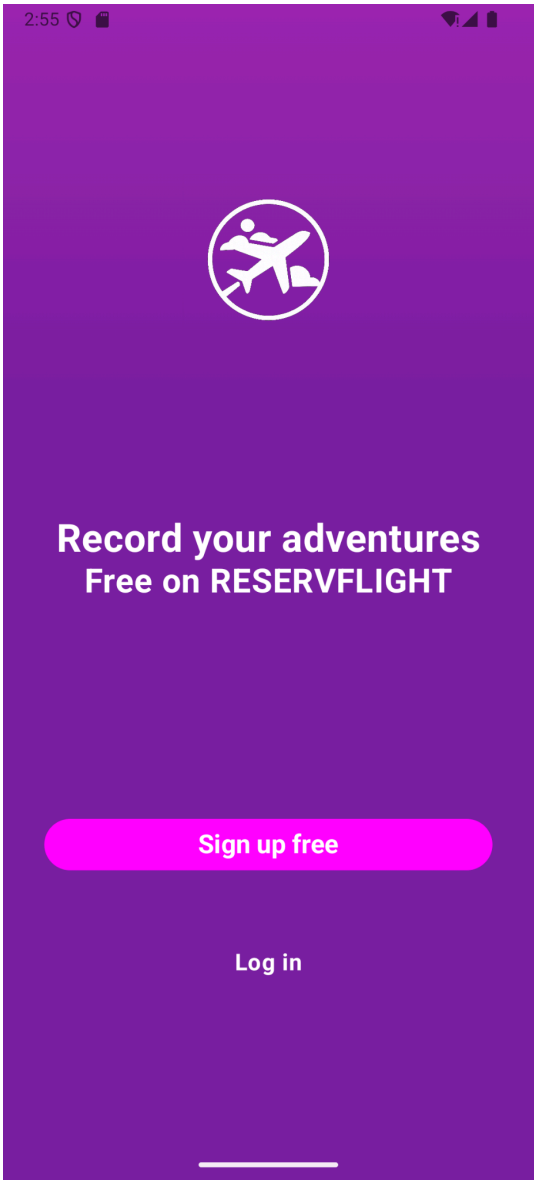
DESCRIPCIÓN DE PROYECTO

Análisis

Pantalla inicial (Initial screen)

La **pantalla inicial** es la primera interfaz que el usuario observa al abrir la aplicación por primera vez. Su diseño tiene como objetivo proporcionar una experiencia de usuario sencilla y clara, facilitando el acceso a las funcionalidades principales de la aplicación.

Componentes de la pantalla:



- **Logo de la aplicación:** El logo se encuentra ubicado en la parte superior o central de la pantalla, ofreciendo una representación visual de la identidad de la aplicación. Este componente no solo refuerza la marca, sino que también proporciona una sensación de profesionalismo y confianza al usuario desde el primer momento.

- **Botón de registro (Sign Up):** El botón de **registro** está destacado para facilitar la navegación de los usuarios que aún no tienen una cuenta en la aplicación. Al pulsar este botón, el usuario es redirigido a la pantalla "Sign Up", donde podrá completar su proceso de registro. El diseño del botón debe ser intuitivo y accesible, garantizando que sea fácil de encontrar y utilizar.

- **Enlace de inicio de sesión (Log In):** Un texto con el enlace **"Log In"** se muestra en la parte inferior de la pantalla, dirigido a los usuarios que ya tienen una cuenta. Este enlace les permitirá acceder a la pantalla "Log In", donde podrán introducir sus credenciales y acceder a la aplicación.

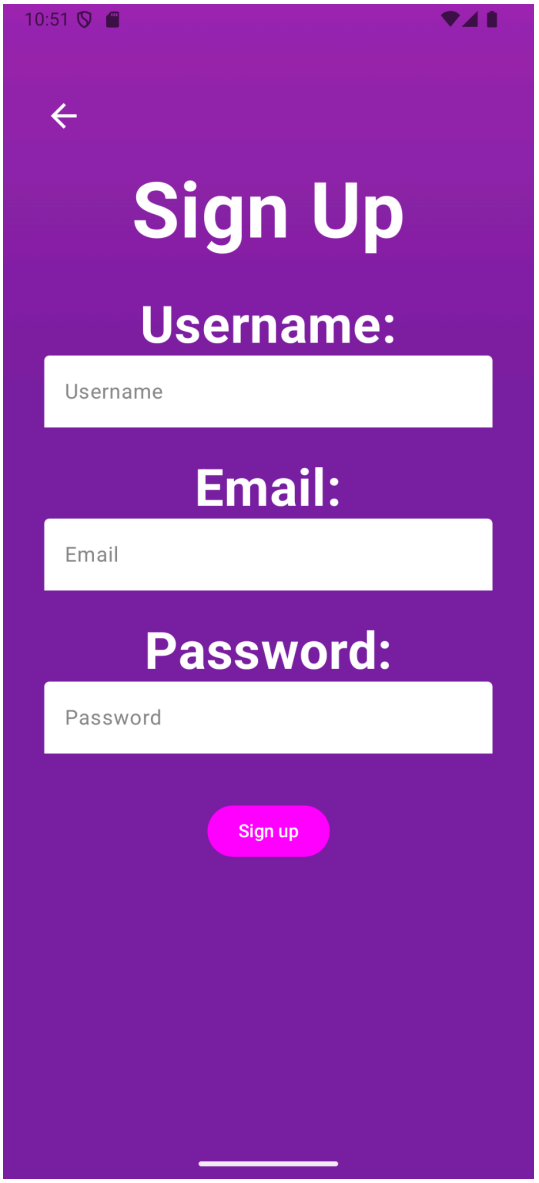
Relación con la aplicación:

La función principal de esta pantalla es facilitar la navegación inicial, proporcionando opciones claras para que el usuario pueda registrarse o iniciar sesión dependiendo de su situación. La interfaz es simple, lo que evita la sobrecarga de información y asegura una transición fluida hacia el uso de la aplicación.

De esta manera, la **pantalla inicial** actúa como punto de entrada, guiando al usuario hacia el flujo de navegación adecuado, según si es nuevo en la aplicación o si ya tiene una cuenta. Una vez que el usuario complete el registro o inicio de sesión, podrá comenzar a interactuar con las funcionalidades principales de la aplicación.

Pantalla de Registro (Sign Up)

La pantalla de registro es una interfaz simple y funcional, diseñada para permitir a los usuarios crear una cuenta en la aplicación de manera rápida y segura.



Componentes de la pantalla:

Campos de entrada (Inputs): La pantalla contiene tres campos de entrada principales:

- **Nombre de usuario (Username):** Este campo permite al usuario establecer un nombre de usuario personalizado. Este identificador mejora la experiencia de uso y puede emplearse como referencia visual dentro de la aplicación. Debe cumplir ciertos criterios de formato y unicidad.

- **Email:** Campo destinado a la introducción de una dirección de correo electrónico válida. Este dato se emplea como identificador principal del usuario en el sistema y es necesario para el acceso y recuperación de la cuenta.
- **Contraseña (Password):** Campo para establecer una contraseña segura. Se requieren criterios mínimos como longitud adecuada y combinación de caracteres para garantizar la protección de la

cuenta.

Botón de registro (Sign up):

Una vez completados los tres campos, el usuario puede pulsar el botón de registro. Este botón activa el proceso de validación y autenticación, el cual es **gestionado completamente por Firebase Auth**.

Firebase se encarga de verificar que:

- El **email tenga un formato válido**.
- La **contraseña cumpla con los requisitos de seguridad**.
- El **usuario no esté ya registrado previamente**.

Si todas las validaciones son superadas con éxito, el sistema registra al usuario tanto en **Firebase Auth**, como en la base de datos propia de la aplicación.

Posteriormente, el usuario es redirigido automáticamente a la pantalla **"Home"**, donde podrá comenzar a utilizar las funcionalidades principales de la aplicación.

Relación con la aplicación:

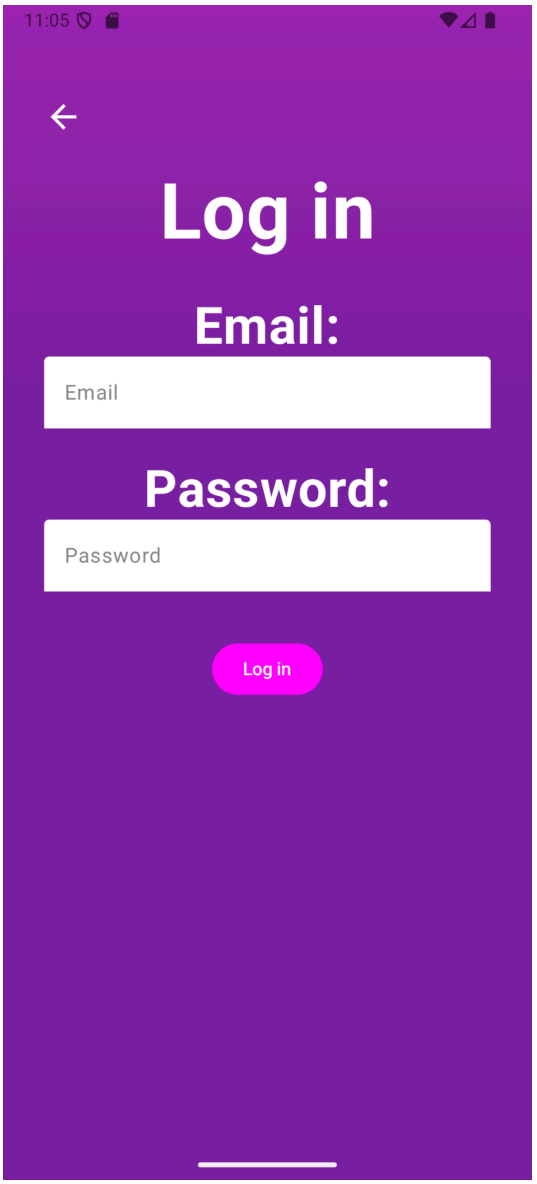
La pantalla de registro cumple un papel esencial en la **fase de incorporación de usuarios**, ya que permite crear cuentas personales de manera segura y eficiente. Gracias a la integración con **Firebase Auth**, el proceso de validación y autenticación es **seguro, robusto y centralizado**, lo que mejora la fiabilidad y escalabilidad del sistema.

El diseño se mantiene minimalista y centrado en los datos imprescindibles (nombre de usuario, email y contraseña), permitiendo una experiencia ágil y sin distracciones.

Una vez registrado, el usuario accede a una experiencia completamente personalizada, pudiendo gestionar y consultar su historial de viajes desde la interfaz principal.

Firebase no solo garantiza la **seguridad y validez de los datos**, sino que además facilita la gestión posterior de usuarios, recuperación de cuentas y sincronización entre dispositivos.

Pantalla de Inicio de Sesión (Login)



La pantalla de inicio de sesión es una interfaz sencilla que permite a los usuarios acceder a su cuenta previamente registrada dentro de la aplicación. Su diseño sigue la misma línea visual que la pantalla de registro, pero está enfocada en la autenticación de usuarios existentes.

Componentes de la pantalla:

Campos de entrada (Inputs):

- **Email:** Campo donde el usuario introduce su dirección de correo electrónico registrada previamente. Este dato actúa como identificador único del usuario en el sistema.
- **Contraseña (Password):** Campo destinado a ingresar la contraseña asociada al correo electrónico introducido. Debe coincidir con la registrada al momento de la creación de la cuenta.

Botón de inicio de sesión:

Una vez introducidos ambos campos, el usuario puede pulsar el botón de inicio de sesión. Al hacerlo, se activa un proceso de validación mediante **Firebase Auth**, el cual verifica que las credenciales sean correctas. Si los datos coinciden con un usuario registrado, se concede el acceso a la aplicación y el usuario es redirigido automáticamente a la pantalla **"Home"**.

Persistencia del estado de sesión:

Una de las ventajas clave de utilizar **Firebase Auth** es que el sistema **mantiene de forma automática el estado de sesión del usuario**, incluso si la aplicación se cierra completamente. Esto significa que, salvo que el usuario **se desconecte manualmente** o **desinstale la aplicación**, **seguirá autenticado la próxima vez que abra la aplicación**, lo que ofrece una experiencia de uso fluida y continua sin necesidad de iniciar sesión repetidamente.

Relación con la aplicación:

La pantalla de inicio de sesión cumple una función crucial en el ciclo de vida del usuario dentro de la aplicación, ya que garantiza que solo aquellos con credenciales válidas puedan acceder a su cuenta y a la información personal asociada a sus viajes.

Gracias a la integración con **Firebase Auth**, se asegura un **control de acceso robusto, seguro y centralizado**, al mismo tiempo que se mejora la experiencia del usuario al gestionar automáticamente la persistencia de la sesión.

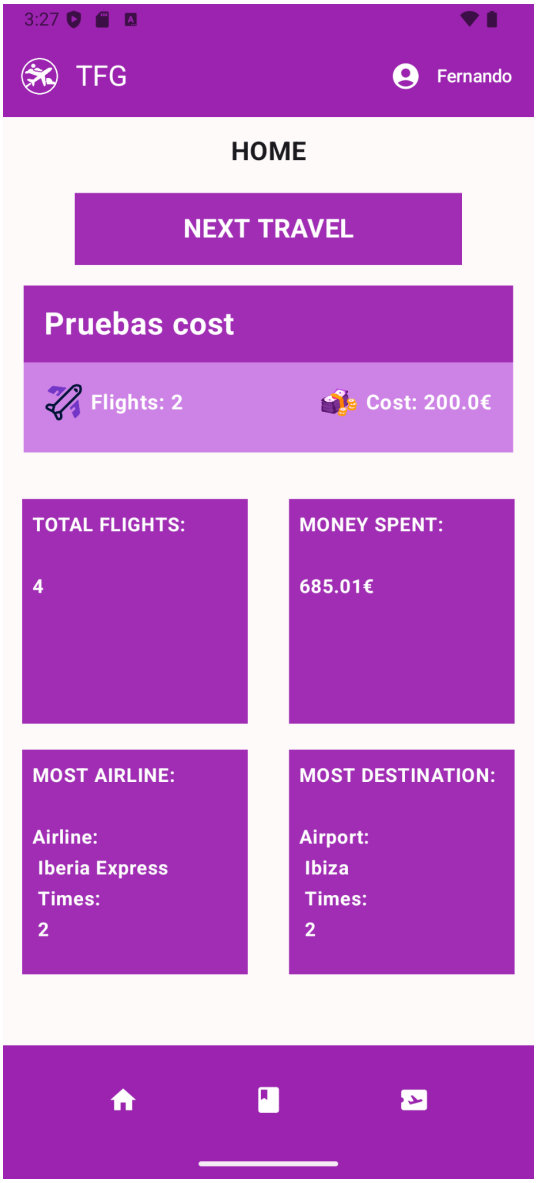
Una vez autenticado, el usuario es redirigido a la pantalla principal **"Home"**, donde puede comenzar a utilizar todas las funcionalidades de la aplicación. El diseño limpio y enfocado permite un acceso rápido, minimizando fricciones innecesarias y manteniendo la seguridad como prioridad.

Pantalla Principal (HomeScreen)

La **pantalla principal (HomeScreen)** es la primera interfaz que los usuarios visualizan tras iniciar sesión correctamente o al abrir la aplicación si su sesión permanece activa, gracias al sistema de persistencia de sesión proporcionado por Firebase Auth. Esta pantalla está diseñada para ofrecer un acceso inmediato a la información más relevante del usuario, como su **reserva más próxima**, estadísticas de uso y un acceso fluido a las secciones principales de la aplicación.

Además, utiliza un componente de **Scaffold reutilizable**, que se emplea de forma coherente en otras pantallas para asegurar una navegación consistente y una experiencia de usuario uniforme.

Componentes de la pantalla:



Top App Bar:

- **Nombre de la aplicación**: Se encuentra posicionado en la parte izquierda superior de la interfaz.
- **Nombre de usuario**: Ubicado en la parte derecha de la barra superior, se muestra dinámicamente el nombre del usuario autenticado en ese momento, facilitando una referencia clara de identidad. Al hacer clic sobre el nombre, se despliega un menú con la opción *Sign out*, que permite cerrar la sesión de forma segura.
- **Icono de la aplicación**: También situado en la parte derecha de la barra superior. Este icono varía en función de la pantalla en la que se encuentre el usuario. En las pantallas principales, se muestra el icono representativo de la aplicación. Sin embargo, en pantallas secundarias o no principales, el icono se sustituye por una flecha de retroceso (*back*), que al pulsarla permite regresar a la pantalla anterior, facilitando la navegación dentro de la app.

Bottom App Bar:

Incluye una serie de **IconButtons** que permiten la navegación directa entre las pantallas más relevantes de la aplicación:

- HomeScreen
- ReservationScreen
- FlightScreen

Este patrón de navegación permite un desplazamiento intuitivo y constante entre secciones sin recargar visualmente la interfaz.

Contenido principal de la HomeScreen:

Tarjeta de la última reserva:

Se presenta una tarjeta destacada que muestra los detalles de la **reserva más próxima en fecha**. Esto permite al usuario acceder rápidamente a la información de su próximo viaje sin tener que buscar entre todas sus reservas.

Panel de resumen estadístico:

Bajo la tarjeta de la última reserva, se muestran **cuatro paneles cuadrados** con métricas clave del historial del usuario. Estas estadísticas permiten una visualización rápida del comportamiento y preferencias de viaje:

1. **Cantidad total de vuelos realizados:**
   Muestra el número acumulado de vuelos registrados. Esto proporciona al usuario una visión clara de su actividad aérea.

2. **Cantidad total de dinero gastado:**
   Indica el gasto total en todos los vuelos. Esta información puede ser útil para control financiero personal o para gestión de viajes de trabajo.

3. **Aerolínea más frecuente:**
   Informa sobre la aerolínea más utilizada por el usuario, junto con la cantidad de veces que ha viajado con ella. Ayuda a detectar patrones de preferencia o fidelidad a ciertas compañías.

4. **Destino más habitual:**
   Muestra el destino al que el usuario ha viajado más veces, incluyendo el número de visitas. Esta métrica ofrece una visión de los destinos favoritos o frecuentes, útil para futuras planificaciones o recordatorios.

Relación con la aplicación:

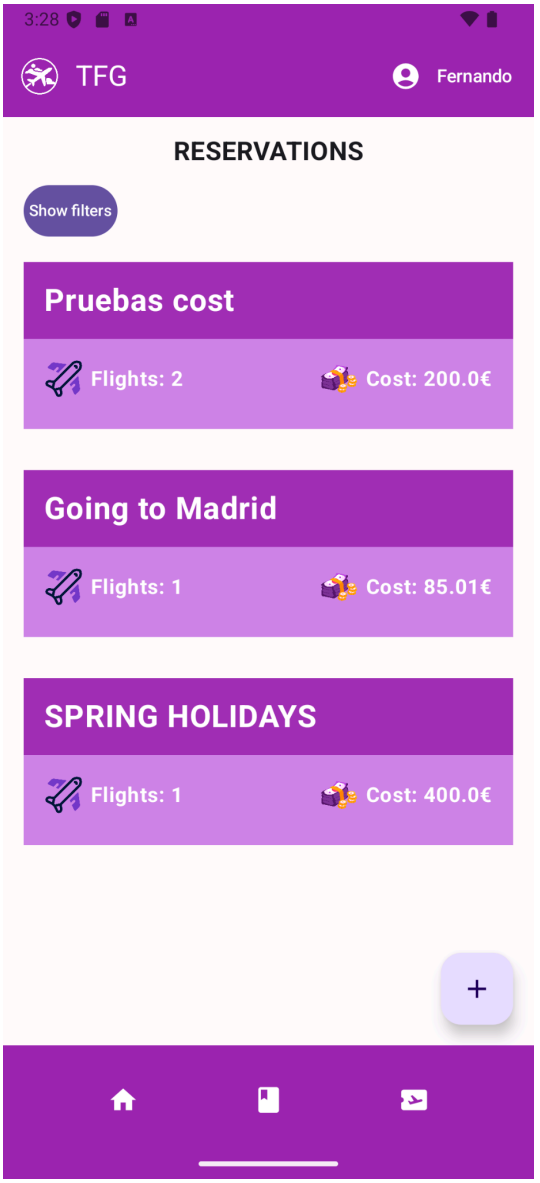
La **HomeScreen** funciona como el **centro de operaciones del usuario**, integrando información útil, navegación rápida y una experiencia personalizada. Su diseño modular y basado en Scaffold favorece la mantenibilidad del código y la consistencia visual a lo largo de toda la aplicación.

Además, el enfoque en la **visualización de estadísticas** mejora la utilidad práctica de la pantalla, transformándola no solo en una puerta de entrada, sino en una herramienta activa de consulta y organización del historial de viajes.

La futura integración del **menú desplegable superior** aumentará aún más la eficiencia de navegación y consolidará esta pantalla como el núcleo funcional del entorno de usuario.

Pantalla de Reservas (ReservationScreen)

La **pantalla de reservas (ReservationScreen)** permite a los usuarios visualizar de manera estructurada y eficiente todas las reservas de vuelos que han creado a través de la aplicación. Esta pantalla utiliza el diseño de **Scaffold unificado**, presente en las pantallas principales, lo cual garantiza una experiencia de navegación coherente, fluida y familiar para el usuario.



Componentes de la pantalla:

Botón flotante de acción (+):

Ubicado en la parte inferior derecha, este botón con el icono de “+” permite al usuario crear una nueva reserva.

- Al pulsarlo, se redirige a la pantalla **AddReservationScreen**, donde podrá introducir los datos necesarios para registrar una nueva reserva de vuelo.

Lista de reservas:

Se muestra una lista **scrollable** de **tarjetas (Cards)**, cada una representando una reserva individual creada por el usuario.
El estilo visual de estas tarjetas es consistente con el diseño empleado en la **HomeScreen**, particularmente en la presentación de la reserva más cercana.

Cada tarjeta de reserva incluye:

- **Nombre de la reserva:** Etiqueta personalizada definida por el usuario, útil para identificar rápidamente cada viaje.
- **Cantidad de vuelos asociados:** Número total de vuelos agrupados dentro de la reserva, facilitando la comprensión de su tamaño y complejidad.
- **Coste total de la reserva:** Suma de los precios de todos los vuelos que forman parte de la reserva. Este dato permite al usuario llevar un control financiero detallado.

Además, **al pulsar sobre una tarjeta**, el usuario es dirigido a la pantalla **InfoReservation**, donde podrá ver en detalle toda la información de esa reserva específica.

Botón "Show Filters":

En la parte superior derecha de la pantalla se encuentra un botón con el texto **"Show Filters"**.
Al pulsarlo, se despliega un menú que ofrece **opciones de filtrado y ordenación**, mejorando la capacidad del usuario para organizar la información según sus necesidades.

Opciones de filtrado y ordenación disponibles:

1. **Filtrado por nombre:**
    - Se presenta un campo de texto donde el usuario puede introducir el nombre (o parte del nombre) de una reserva para buscarla directamente.
2. **Ordenación mediante botones:**

- **Newer:** Ordena las reservas según su fecha de creación, alternando entre más reciente y más antigua.
- **Flights:** Ordena por la cantidad total de vuelos que contiene cada reserva.
- **Cost:** Ordena por el coste total de cada reserva.

3. Estos tres botones alternan entre orden ascendente y descendente al ser presionados sucesivamente. Una **flecha indicadora** acompaña cada botón para señalar la dirección actual del orden (↑ ascendente, ↓ descendente).

Relación con la aplicación:

La **ReservationScreen** cumple un rol clave al ofrecer al usuario una visión clara y centralizada de todas sus reservas.
La posibilidad de **añadir nuevas reservas**, **consultar reservas anteriores**, **filtrarlas por nombre** o **ordenarlas por diferentes criterios** aporta una **gran flexibilidad y control** sobre la información.

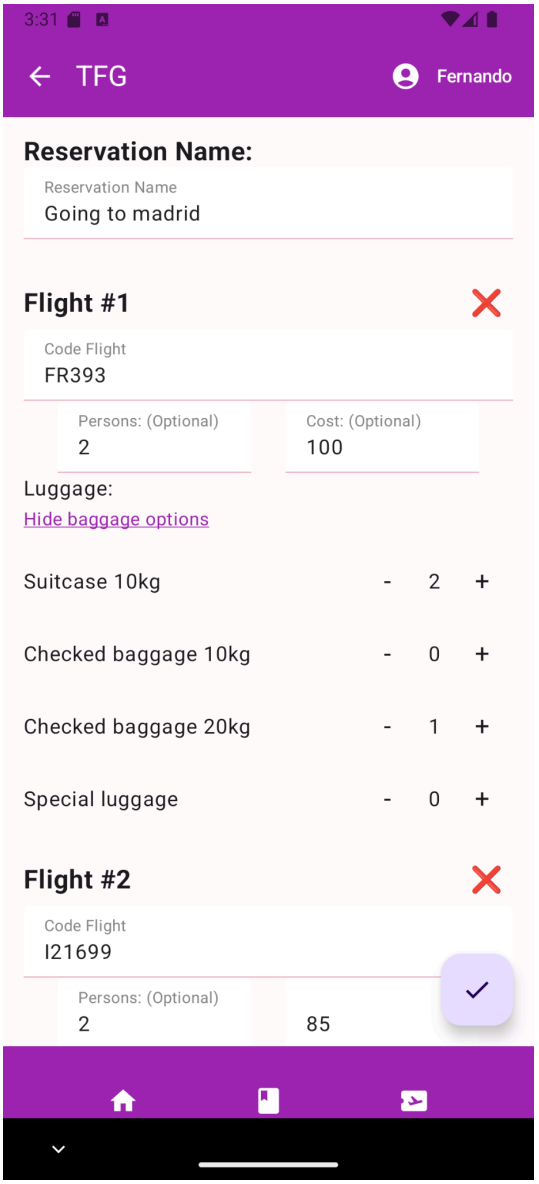
La integración del botón “Show Filters” mejora notablemente la usabilidad, especialmente en casos donde el número de reservas es elevado.
El acceso directo a la **pantalla InfoReservation** desde cada tarjeta también optimiza el flujo de navegación, permitiendo al usuario profundizar en los detalles con tan solo un clic.

Finalmente, el uso consistente del **Scaffold estándar** mantiene la experiencia de usuario coherente con el resto de la aplicación, reforzando una interfaz intuitiva y bien estructurada.

Pantalla de Añadir Reserva (AddReservationScreen)

La **pantalla de añadir reserva (AddReservationScreen)** permite a los usuarios crear nuevas reservas de vuelos de forma flexible y dinámica. Utiliza el diseño de **Scaffold estándar**, garantizando una estructura coherente y una navegación fluida, en línea con el resto de la aplicación.

Componentes de la pantalla:



Reservation Name (Nombre de la reserva):

En la parte superior de la pantalla se encuentra un campo de texto destinado a ingresar el **nombre de la reserva**. Este identificador personalizado facilita la organización y posterior consulta de las reservas por parte del usuario.

Campos de entrada por vuelo:

Cada vuelo que se desee añadir a la reserva incluye los siguientes campos:

- **Código de vuelo (Flight Code):** Campo de texto donde el usuario introduce el identificador del vuelo. Este puede corresponder al código **iata** proporcionado por la aerolínea. Es un dato clave para la validación y posterior gestión.
- **Número de personas (Persons):** Campo numérico que permite indicar cuántas personas viajan en ese vuelo.
- **Coste del vuelo (Cost):** Campo numérico que registra el precio total pagado por el vuelo.

- **Equipaje (Luggage):**
  A través de una línea de texto con la opción **"Show baggage options"** o **"Hide baggage options"**, el usuario puede desplegar o esconder las opciones de equipaje según su necesidad.
  Si está habilitado, se permite seleccionar tipos de equipaje, con botones "+" y "–" para ajustar la cantidad.

  - Si no se selecciona ningún equipaje (cantidad cero en todos), no se guardará información de equipaje en la base de datos.

Gestión de múltiples vuelos:

- Cada vez que el usuario pulsa el botón **"ADD FLIGHT +"**, se generan nuevos campos como los descritos anteriormente, permitiendo añadir múltiples vuelos dentro de una misma reserva (por ejemplo, vuelo de ida y vuelta o conexiones intermedias).

- Una vez se añade un nuevo vuelo, aparece una **cruz (✖) a la derecha** de su encabezado. Esta funcionalidad permite eliminar dicho vuelo antes de guardar la reserva, ofreciendo flexibilidad y control total al usuario.

Botón de guardar (✔):

Ubicado en la esquina inferior derecha, este botón permite guardar toda la información ingresada en la base de datos.

Proceso de validación:

Antes de realizar el guardado, el sistema verifica si los vuelos con los códigos introducidos ya existen en la base de datos:

- **Si el vuelo ya existe:** Se asocia directamente con la nueva reserva del usuario, sin necesidad de nuevas consultas externas.

- **Si el vuelo no existe:** Se realiza una consulta a la API de **AviationStack** para obtener la información completa y actualizada del vuelo.
  Una vez validado, el vuelo se guarda en la base de datos tanto para este usuario como para posibles futuras reservas de otros usuarios.

Confirmación:

Si la reserva se guarda correctamente, el sistema redirige automáticamente al usuario a la pantalla **ReservationScreen**, donde la nueva reserva aparecerá en primer lugar en la lista, indicando su reciente creación.

Relación con la aplicación:

La **AddReservationScreen** cumple un rol central en el proceso de planificación de viajes dentro de la aplicación.
Permite a los usuarios agregar reservas personalizadas, con uno o varios vuelos, gestionando información clave como el número de pasajeros, el coste y el equipaje asociado.

La **posibilidad de mostrar u ocultar las opciones de equipaje**, así como la de **eliminar vuelos individualmente antes del guardado**, mejora significativamente la usabilidad y adaptabilidad de la interfaz.

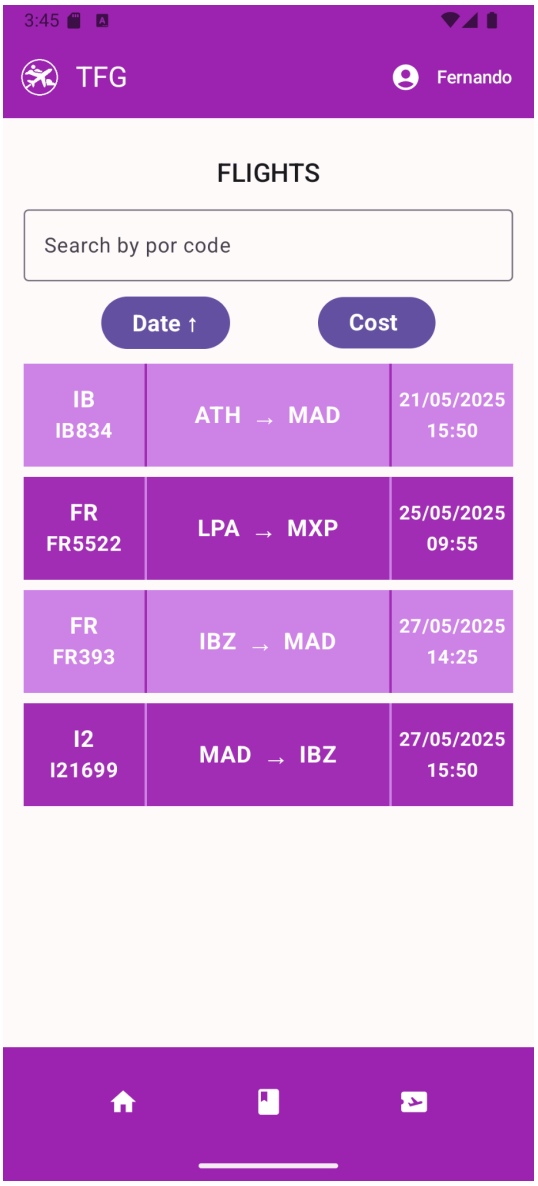
El uso de la API de **AviationStack** para validar vuelos no registrados previamente asegura que los datos almacenados sean precisos y actualizados, reforzando la fiabilidad del sistema.

Finalmente, la coherencia en el diseño y el flujo de navegación, gracias al Scaffold compartido con otras pantallas, garantiza una **experiencia de usuario consistente, intuitiva y profesional**.

Pantalla de Vuelos (FlightsScreen)

La **pantalla de vuelos (FlightsScreen)** permite a los usuarios visualizar de forma clara y ordenada todos los vuelos que han sido registrados en la aplicación, ya sea como parte de una reserva o añadidos individualmente. Su diseño se basa en el uso del **Scaffold estándar**, asegurando una navegación coherente y consistente con el resto de la aplicación.

Componentes de la pantalla:



Tarjetas de vuelo (Flight Cards):

Cada vuelo se presenta dentro de una **tarjeta individual**, lo que facilita la consulta visual de los datos más relevantes. Estas tarjetas están organizadas dentro de un **LazyColumn**, una estructura eficiente para listas extensas que permite un desplazamiento fluido incluso con un gran volumen de vuelos.

Cada tarjeta muestra la siguiente información clave:

- **Código de aerolínea:** Identificador de la compañía aérea correspondiente.
- **Código de vuelo:** Número único asignado al vuelo.
- **Código del aeropuerto de salida:** Abreviatura IATA del aeropuerto de origen.
- **Código del aeropuerto de llegada:** Abreviatura IATA del aeropuerto de destino.
- **Hora de salida:** Hora programada de salida, obtenida a partir de los datos internos o de la API de validación externa.

Mejora visual:

Para facilitar la diferenciación rápida entre tarjetas, se ha implementado un esquema de **variación de colores entre tarjetas alternas**. Esto mejora la legibilidad general, especialmente cuando hay múltiples registros en pantalla.

Sistema de filtros y ordenación:

Con el objetivo de mejorar la experiencia del usuario y facilitar la gestión de la información, se han añadido funcionalidades de **búsqueda y ordenación**:

- **Campo de texto (Search):** Permite buscar vuelos introduciendo directamente su código. El listado se filtra en tiempo real según los caracteres escritos.
- **Botones de ordenación:**
  - **Date:** Ordena los vuelos por su fecha de salida.
  - **Cost:** Ordena por el coste total asociado al vuelo (si está disponible).

Estos botones **alternan el orden ascendente y descendente** cada vez que se presionan, e incluyen un **ícono de flecha** que indica la dirección actual del ordenamiento. Esta lógica es similar a la implementada en la pantalla ReservationScreen, manteniendo la coherencia funcional en toda la aplicación.

Navegación a pantalla de detalles:

Al pulsar sobre cualquiera de las tarjetas de vuelo, el usuario es redirigido a la pantalla **InfoFlightScreen**, donde podrá consultar de manera ampliada y detallada toda la información relacionada con ese vuelo. Esta separación entre vista general y vista detallada evita sobrecargar la interfaz principal y facilita una gestión más enfocada.

Relación con la aplicación:

La **FlightsScreen** representa un componente esencial dentro de la arquitectura funcional de la aplicación, ya que centraliza toda la información relativa a los vuelos que el usuario ha registrado.

Gracias a la disposición en tarjetas, el uso de LazyColumn, la diferenciación cromática, y las herramientas de búsqueda y ordenación, esta pantalla proporciona una experiencia de usuario ágil, visualmente agradable y altamente funcional.

Pantalla de Información de Reserva (InfoReservationScreen)

La **pantalla InfoReservationScreen** está diseñada para mostrar de forma detallada toda la información correspondiente a una reserva específica. Esta pantalla es accesible desde dos puntos principales de la aplicación: al pulsar sobre cualquier tarjeta de reserva en la pantalla **ReservationScreen**, o bien desde la pantalla **InfoFlightScreen**, si se accede a través de un vuelo vinculado a una reserva.

Componentes de la pantalla:

Encabezado superior:

En la parte superior de la interfaz, se presenta un **cuadro con fondo morado** que contiene:

- **Nombre de la reserva:** Mostrado de forma destacada para facilitar la identificación del registro.
- **Botón "Delete" (Eliminar):** De color rojo y ubicado junto al nombre de la reserva. Al pulsarlo, se procede a eliminar la reserva y todos los vuelos asociados **únicamente para el usuario actual**. Esto significa que los vuelos en sí **no se eliminan completamente de la base de datos**, ya que pueden estar asociados a otros usuarios o utilizarse en futuras reservas. Esta decisión de diseño mantiene la integridad de los datos compartidos.

Resumen de la reserva:

Inmediatamente debajo del encabezado, se encuentra un **cuadro rosa claro** que ofrece una vista rápida y visual de dos datos clave:

- **Cantidad de vuelos asociados:** Muestra el número total de vuelos incluidos dentro de la reserva.

- **Coste total de la reserva:** Indica el importe total sumado de todos los vuelos que conforman la reserva.

Este diseño visual es **coherente con las tarjetas de reserva** vistas en la pantalla principal y en la HomeScreen, lo que refuerza la uniformidad de la interfaz y facilita la familiarización del usuario.

Botón “Add Flight to Reservation”:

A continuación, se incluye un botón con el texto “Add Flight to Reservation”, que permite al usuario añadir más vuelos a la reserva actual. Al pulsarlo, el usuario es redirigido a la pantalla **AddFlighToReservationScreen**, que ahora también está adaptada para **añadir vuelos a reservas ya existentes**, no solo para crear nuevas.

Listado de vuelos de la reserva:

En la parte inferior de la pantalla, se muestran **todas las tarjetas de vuelo** correspondientes a los vuelos que componen la reserva actual. Estas tarjetas mantienen el mismo estilo visual utilizado en la pantalla **FlightsScreen**, presentando la información clave de cada vuelo (código, aerolínea, aeropuerto de salida/llegada, hora de salida, etc.).

- Al pulsar sobre cualquiera de estas tarjetas, el usuario será redirigido a la pantalla **InfoFlightScreen**, donde podrá consultar en profundidad todos los detalles del vuelo seleccionado.

Sistema de filtros:

Sobre el listado de vuelos se encuentra un botón con la etiqueta **“Mostrar filtros”**, el cual despliega las opciones de filtrado y ordenación. Estas son:

- **Botón "Date":** Ordena los vuelos por la fecha de salida.

- **Botón "Cost":** Ordena los vuelos por su precio.

Ambos botones **alternan el orden ascendente o descendente** cada vez que se presionan, e incluyen un ícono de flecha para indicar la dirección actual del orden. Este sistema replica el funcionamiento presente en la pantalla **FlightsScreen**, a excepción del campo de búsqueda por código de vuelo, que no está incluido aquí debido a que ya se está trabajando sobre una única reserva.

Relación con la aplicación:

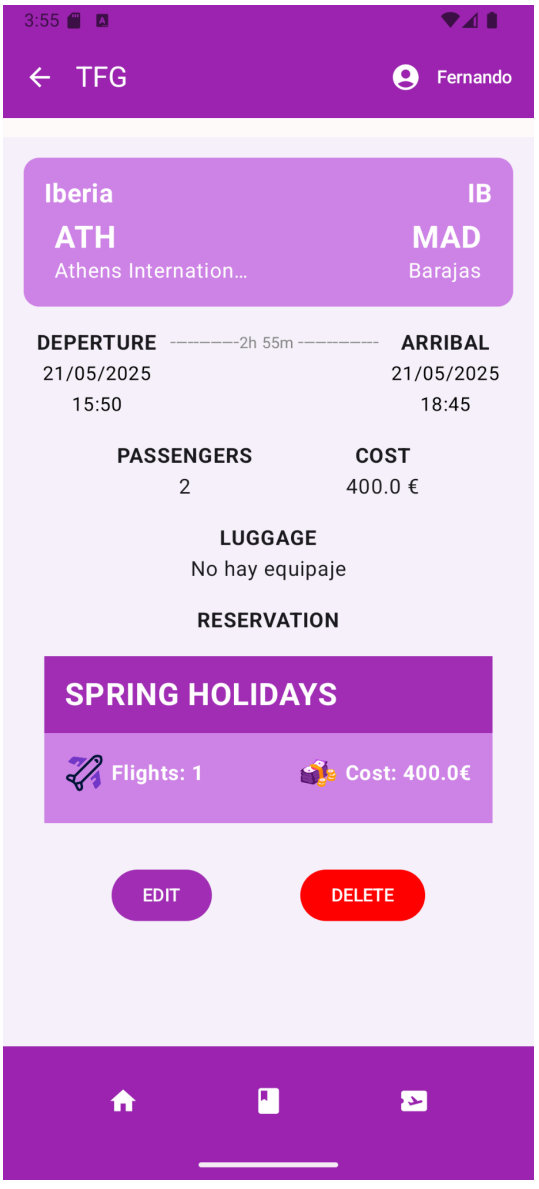
La **InfoReservationScreen** cumple un papel esencial en la gestión detallada de las reservas. Permite al usuario:

- Visualizar claramente la estructura y contenido de cada reserva.
- Añadir nuevos vuelos a una reserva existente.
- Acceder a los detalles de cada vuelo individual.
- Eliminar reservas sin comprometer la base de datos global compartida entre múltiples usuarios.

Todo esto se realiza manteniendo el diseño unificado y reutilizando el **Scaffold estándar**, lo que refuerza una experiencia de usuario intuitiva, fluida y visualmente coherente.

Pantalla de Información de Vuelo (InfoFlightScreen)

La **pantalla InfoFlightScreen** permite al usuario consultar y gestionar los detalles completos de un vuelo previamente registrado en la aplicación. Este vuelo puede formar parte de una o varias reservas y contener información personalizada vinculada al usuario. La pantalla utiliza el diseño **Scaffold estándar**, garantizando una experiencia visual uniforme con el resto de la interfaz.



Se accede a esta pantalla desde **FlightsScreen** (listado general de vuelos del usuario) o desde otras vistas como **InfoReservationScreen**, cuando el vuelo está asociado a una reserva específica.

Componentes de la pantalla:

1. Tarjeta informativa del vuelo

En la parte superior de la pantalla se muestra una **card visual** que presenta los **datos principales del vuelo seleccionado**, tales como:

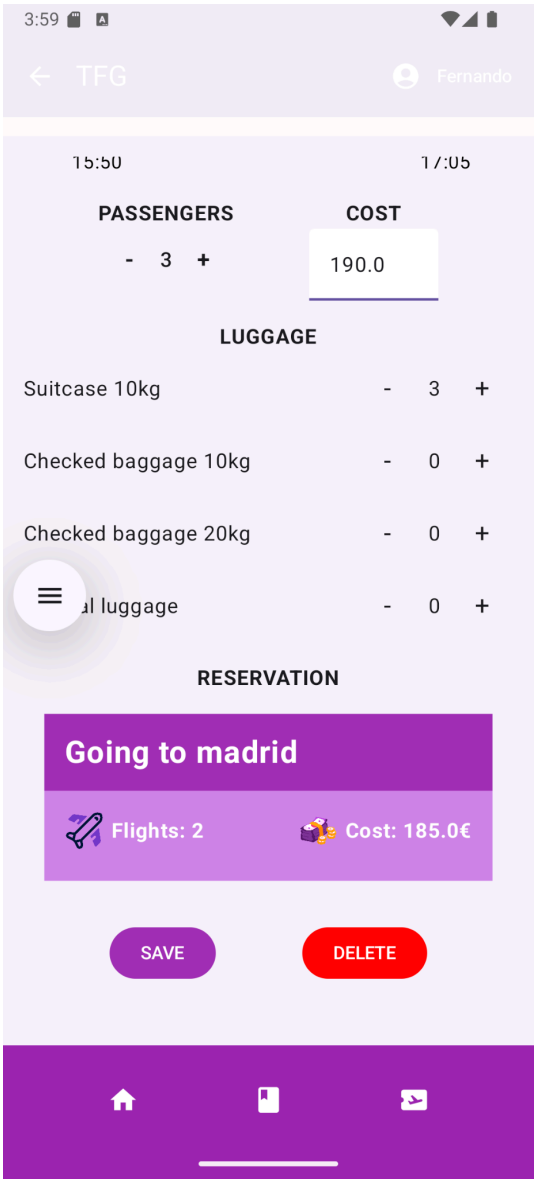
- **Compañía aérea** y **código de vuelo.**
- **Aeropuertos de salida y llegada**, tanto por nombre como por código IATA.
- **Fechas y horas** exactas de salida y llegada.
- **Duración del vuelo**, calculada automáticamente en función del horario registrado.

Esta visualización ofrece un resumen claro y estructurado de la información básica del vuelo, destacando los datos más relevantes para el usuario.

2. Información editable del vuelo

Debajo de la tarjeta principal, se presenta una sección de información adicional vinculada al usuario actual. Esta incluye campos que pueden ser **editados tras pulsar un botón específico**:

- **Número de personas (Passengers):** Indica cuántos pasajeros fueron registrados por el usuario para este vuelo.
- **Coste del vuelo (Cost):** Representa el precio total registrado para ese vuelo.
- **Equipaje (Luggage):** Muestra los tipos y cantidades de equipaje añadidos al vuelo. El equipaje puede modificarse dinámicamente utilizando botones para incrementar o disminuir la cantidad de cada tipo.



Por defecto, estos campos se muestran en **modo lectura**. Para modificarlos, el usuario debe pulsar el botón **"Edit"**, que habilita la edición y transforma su etiqueta en **"Save"**. Al pulsar "Save", los cambios se guardan en la base de datos del usuario, actualizando inmediatamente la interfaz con la nueva información.

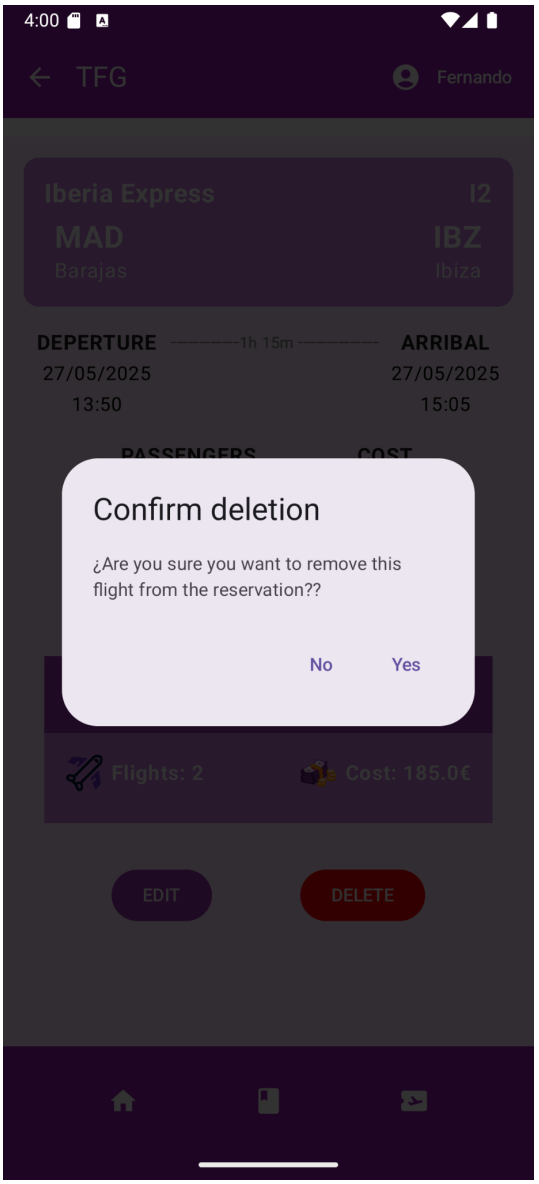
3. Tarjeta de la reserva asociada

En caso de que el vuelo esté vinculado a una reserva, se muestra una **tarjeta resumen** que identifica dicha reserva (nombre, número de vuelos y coste total). Al pulsar sobre esta tarjeta, el usuario es redirigido a la pantalla **InfoReservationScreen**, lo que permite una navegación contextual entre los vuelos y sus respectivas reservas.

Funcionalidades adicionales:

Botón “Edit / Save”:

- **Modo “Edit”:** Al pulsar, convierte los campos de personas, coste y equipaje en editables.
- **Modo “Save”:** Guarda los cambios realizados y restaura la interfaz a modo lectura.



Esta funcionalidad mejora la usabilidad al permitir modificar solo la información necesaria, sin saturar visualmente la pantalla.

Botón “Delete”:

Permite al usuario eliminar el vuelo **del historial personal**, pero no de la base de datos global (por si otros usuarios tienen el mismo vuelo registrado).

- **Al pulsar este botón**, aparece un mensaje de **confirmación**.
- Si el vuelo pertenece a una reserva con **más de un vuelo**, se elimina correctamente y el usuario es redirigido automáticamente a la **pantalla anterior**.
- Si el vuelo es el **único asociado a su reserva**, se muestra un mensaje de advertencia indicando que **no se puede eliminar el último vuelo sin borrar primero la reserva completa**. En ese caso, el usuario deberá gestionar la eliminación desde **InfoReservationScreen**.

Relación con la aplicación:

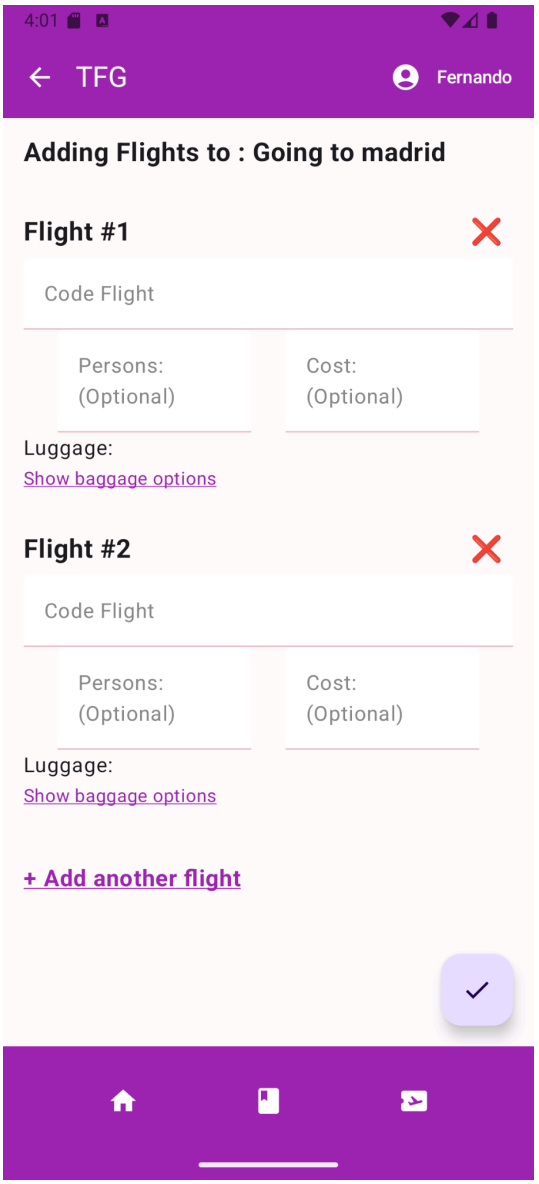
La pantalla **InfoFlightScreen** desempeña una función clave en la gestión individual de vuelos dentro del sistema. Permite:

- Revisar la información técnica y personal de un vuelo.
- Editar datos relevantes como el número de pasajeros, el coste o el equipaje.
- Conectarse directamente con la reserva asociada.
- Eliminar vuelos del historial personal de forma segura y validada.

Esta pantalla refuerza la lógica relacional de la aplicación, manteniendo la consistencia de navegación entre vuelos y reservas. Su diseño coherente, apoyado en el uso del **Scaffold estándar**, asegura una experiencia de usuario fluida, clara y eficiente.

Pantalla de Añadir Vuelo a Reserva (AddFlightToReservationScreen)

La pantalla **AddFlightToReservationScreen** permite a los usuarios añadir nuevos vuelos a una reserva ya existente. Se accede a través del botón **"Add flight to Reservation"** ubicado en la pantalla **InfoReservationScreen**, que muestra los detalles de una reserva previamente creada.



Esta pantalla reutiliza la interfaz y lógica base de la pantalla **AddReservationScreen**, manteniendo la coherencia visual y funcional del sistema, pero **con ajustes específicos al contexto**, ya que la reserva destino ya existe y no es necesario volver a definir su nombre.

Componentes de la pantalla:

Campos de entrada de vuelo:

La pantalla incluye un conjunto de campos para introducir la información de cada vuelo, exactamente igual que en la creación de reservas nuevas:

- **Código de vuelo (Flight Code):** Campo de texto donde el usuario introduce el identificador único del vuelo.
- **Número de personas (Persons):** Campo numérico para indicar la cantidad

de pasajeros.

- **Coste del vuelo (Cost):** Campo numérico donde se especifica el importe pagado por el vuelo.

- **Equipaje (Luggage):** Selector interactivo con botones "+" y "–" para registrar las cantidades de diferentes tipos de equipaje.

Botón "Add Flight+":

Al pulsar este botón, se genera una nueva sección de entrada idéntica a la actual, lo que permite añadir múltiples vuelos de forma consecutiva a la misma reserva. Cada vuelo nuevo tendrá un **botón de eliminación ("X")** en la parte derecha para descartarlo antes de guardar, en caso de haberlo añadido por error.

Opción de mostrar/ocultar equipaje:

Antes de los campos de equipaje, se muestra una línea de texto tipo **"Show baggage options" / "Hide baggage options"**, que permite al usuario decidir si desea gestionar o no la información de equipaje para cada vuelo, simplificando así la interfaz en casos donde no sea relevante.

Botón de guardar (✔):

Ubicado en la parte inferior derecha de la pantalla. Al pulsarlo, se realiza la validación y almacenamiento de todos los vuelos añadidos:

- Si el código de vuelo ya existe en la base de datos, se **asocia a la reserva existente del usuario**.

- Si no existe, la aplicación consulta la **API de AviationStack**, guarda la información del vuelo y luego lo vincula a la reserva del usuario.

Una vez finalizado el proceso de guardado exitosamente, la aplicación redirige automáticamente al usuario a la **pantalla ReservationScreen**, donde la reserva recientemente actualizada **aparecerá en primer lugar en la lista**, reflejando los cambios realizados.

Diferencias con AddReservationScreen:

La pantalla **AddFlightToReservationScreen** es funcionalmente equivalente a **AddReservationScreen**, con la diferencia de que **no incluye el campo**

“Reservation Name” al inicio, ya que el nombre de la reserva está definido y no puede ser modificado en este punto.

Esto asegura que los vuelos añadidos se integren correctamente a la reserva seleccionada, sin crear nuevas entradas ni permitir confusiones respecto a la identidad de la reserva original.

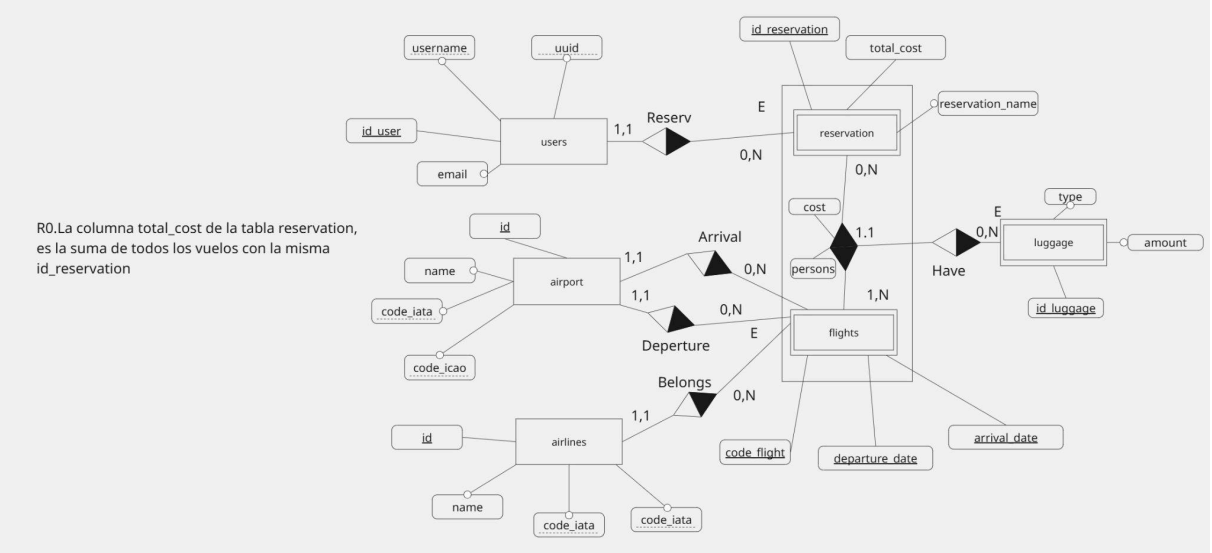
Relación con la aplicación:

Esta pantalla mejora la **flexibilidad** del sistema, permitiendo que una reserva no sea un conjunto cerrado al momento de su creación, sino que pueda **evolucionar en el tiempo** mediante la adición de nuevos vuelos. La interfaz optimizada y la reutilización del diseño de AddReservationScreen garantizan una experiencia de usuario coherente y eficiente.

Su integración directa con **InfoReservationScreen** refuerza la lógica relacional de la aplicación y facilita una gestión detallada y progresiva de los viajes por parte del usuario.

Diseños

Entidad Relación



Diseño Lógico

users (id_user:int, username: varchar, email:varchar, uuid)
CP:{id_user}
VNN:{username, email, uuid}
UNIC:{email, username, uuid}

airport(id_airport:int, name:varchar, code_iata:varchar, code_icao:varchar)
CP:{id_airport}
VNN:{name, code_iata,code_icao}
UNIC:{code_iata,code_icao}

airlines(id_airline: int, name:varchar,code_iata:varchar, code_icao:varchar)
CP:{id_airlines}
VNN:{name, code_iata, code_icao}
UNIC:{code_iata,code_icao}

reservation(id_reservation:int, total_cost:double, id_user:int, reservation_name)
CP:{id_reservation}

Fernando Vicent Torres Serra

CF:{id_user} → users(id_user)
VNN:{id_user, reservation_name}

flights(code_flight:int, departure_date:date, arrival_date:date,id_airline:int , departure_airport: varchar, arrival_airport:varchar)
CP:{code_flight, departure_date, arrival_date }
VNN:(departure_airport, arrival_airport, id_airline)
CF:{departure_airport} → airport(id_airport)
CF:{arrival_airport} → airport(id_airport)
CF:{id_airline} → airline(id_airline)

fligh_reservation(id_reservation:int, code_flight:int, departure_date:date, arrival_date:date, cost: double, persons:int)
CP:{id_reservation, code_flight, departure_date, arrival_date}
CF:{id_reservation} → reservation(id_reservation)
CF:{code_flight} → flights(code_flight)
CF:{departure_date} → flights(departure_date)
CF:{arrival_date} → flights(arrival_date)

luggage(id_luggage:int,type:varchar,amount:int, id_reservation:int, code_flight:int, departure_date:date, arrival_date:date)
CP:{id_luggage, id_reservation, code_flight}
VNN:{id_reservation, code_flight, departure_date, arrival_date,type, amount }
CF:{id_reservation} → reservation(id_reservation)
CF:{code_flight} → flights(code_flight)
CF:{departure_date} → flights(departure_date)
CF:{arrival_date} → flights(arrival_date)

Desarrollo

El desarrollo de la aplicación se ha realizado siguiendo un enfoque incremental, priorizando desde el inicio la conectividad con servicios en la nube y el uso de tecnologías modernas de desarrollo móvil. A continuación, se resumen los pasos clave llevados a cabo durante el proceso de implementación:

1. Configuración de Firebase y sistema de autenticación

El primer paso fue la integración de **Firebase Auth** en el proyecto de Android. Se configuró el sistema de autenticación de usuarios, permitiendo el acceso mediante correo electrónico y contraseña, y garantizando así una experiencia personalizada y segura.

Una vez autenticado, cada usuario queda identificado de forma única, lo cual permite una gestión aislada y protegida de su información personal (reservas, vuelos, equipaje, etc.).

2. Definición de la estructura de datos en PostgreSQL

Se diseñó la estructura de la base de datos en **PostgreSQL**, teniendo en cuenta las relaciones entre los distintos elementos del sistema:

- Usuarios
    - Contienen reservas
        - Cada reserva contiene uno o más vuelos
            - Cada vuelo puede tener información de equipaje

Se optó por un diseño relacional eficiente, que permitiera realizar consultas específicas y actualizaciones rápidas sin cargar datos innecesarios.

3. Diseño base de la aplicación y navegación

Se estableció la arquitectura de navegación utilizando **Jetpack Navigation** y el patrón **MVVM** (Model-View-ViewModel), separando claramente las capas de lógica, vista y datos.
La interfaz fue construida con **Jetpack Compose** y el componente **Scaffold** como base para todas las pantallas principales. Esto permitió un diseño consistente, adaptable y fácil de mantener.

4. Implementación de pantallas clave

La aplicación fue desarrollada progresivamente, comenzando por las pantallas más fundamentales:

- HomeScreen
- ReservationScreen
- FlightsScreen
- AddReservationScreen
- InfoReservationScreen
- InfoFlightScreen
- AddFlightToReservationScreen

Cada pantalla fue diseñada para ofrecer una experiencia clara, responsiva y centrada en el usuario, incluyendo filtros, tarjetas, botones flotantes y navegación fluida.

5. Conexión con la API externa de AviationStack

Para garantizar la validez de los vuelos añadidos, se integró la API de **AviationStack** mediante **Retrofit**.
Cuando un usuario introduce un nuevo código de vuelo no registrado previamente, la app consulta la API, obtiene los datos y los guarda automáticamente en la base de datos (PostgreSQL).

Esto permite mantener una base de datos compartida de vuelos, reutilizable entre usuarios y con datos reales y actualizados.

6. Gestión y validación de datos

Todas las operaciones de creación, edición o eliminación de datos se realizan a través de peticiones HTTP gestionadas con **Retrofit** contra la API que comunica con la base de datos **PostgreSQL**.

También se implementaron validaciones en cada acción:

- Verificar que no se repita un vuelo en la misma reserva
- No permitir eliminar el último vuelo de una reserva sin eliminar primero la reserva
- Validar campos como coste, número de personas o equipaje antes de guardar

7. Interacciones avanzadas y filtros

Una vez funcional la base del sistema, se añadieron mejoras interactivas para el usuario:

- Filtros y ordenación dinámica por nombre, coste o fecha

- Tarjetas con diseño alternativo para mejorar la lectura
- Botones "Editar" que se convierten en "Guardar" en campos editables
- Mensajes de confirmación al eliminar elementos

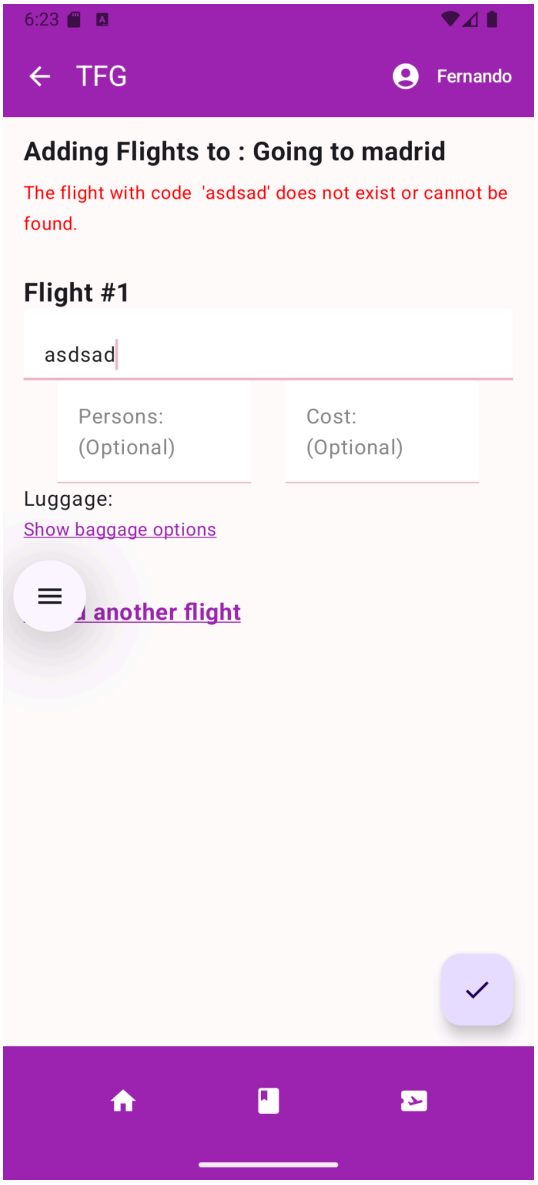
8. Pruebas funcionales continuas

Durante todo el desarrollo se realizaron pruebas manuales, simulando diferentes flujos de uso: añadir reservas con múltiples vuelos, editar campos, navegar entre pantallas o usar filtros.

Esto permitió detectar errores tempranos y refinar tanto el funcionamiento como la interfaz visual de la aplicación.

Pruebas

Debido a la proximidad de la entrega del trabajo y las limitaciones de tiempo**, no se ha podido realizar una campaña de pruebas automatizadas ni capturar evidencias gráficas** (pantallazos) de cada funcionalidad. No obstante, durante el desarrollo de la aplicación se han llevado a cabo **pruebas manuales constantes**, asegurando que las funciones críticas operan correctamente.



Verificaciones realizadas:

- **Inicio de sesión**: Se ha verificado que los usuarios pueden autenticarse correctamente mediante Firebase Auth y que los datos incorrectos (como email o contraseña erróneos) muestran un mensaje en rojo indicando el error.
- **Visualización del nombre del usuario:** Una vez autenticado, el nombre se muestra en la parte superior derecha de la aplicación.
- **Despliegue del menú:** Al hacer clic sobre el nombre del usuario, se despliega una opción para cerrar sesión (Sign out), que funciona correctamente.
- **Creación de reservas**: Se ha probado que al crear una reserva, esta se almacena en la base de datos y se muestra en la pantalla principal.
- **Añadir vuelos a reservas:**
    - Si se introduce un **codeFlight que no existe**, se muestra un **mensaje de error en rojo** que informa claramente al usuario.
    - Si los campos están incompletos o mal formateados, también se muestra un mensaje en rojo, impidiendo continuar hasta corregirlo.
- **Gestión de errores:** Toda la aplicación está diseñada para mostrar errores **visiblemente en color rojo**, lo cual ha sido esencial para detectar fallos durante las pruebas manuales.

AMPLIACIONES FUTURAS

Durante el desarrollo de la aplicación se identificaron diversas posibilidades de mejora y expansión que podrían implementarse en versiones futuras, tanto para enriquecer la experiencia del usuario como para dotar a la plataforma de un enfoque más completo, social e interactivo. A continuación, se detallan algunas de las funcionalidades más destacadas propuestas como ampliaciones:

1. Mapa interactivo de vuelos

Implementar un mapa visual e interactivo que muestre los lugares del mundo donde el usuario ha viajado. Cada vuelo registrado marcaría una línea entre el origen y el destino, permitiendo visualizar rutas, países visitados y destinos frecuentes.

2. Sistema de logros y medallas

Incorporar una funcionalidad de logros desbloqueables, que premie al usuario por cumplir ciertos hitos, como:

- Volar varias veces al mismo destino.
- Registrar vuelos con diferentes compañías.
- Visitar distintos continentes.
- Superar cierto número de horas en vuelo.
- Completar una cantidad específica de vuelos con una misma persona.

Este sistema gamificaría la experiencia, fomentando el uso continuo de la aplicación.

3. Sistema de amigos e interacción social

Añadir una sección de amigos, donde los usuarios puedan conectarse entre sí, visualizar vuelos compartidos o retos colaborativos, como:

- Vuela dos veces con el mismo amigo.
- Crea una reserva grupal.
- Visita un destino popular entre tus contactos.

Esto convertiría la aplicación en una experiencia más comunitaria y colaborativa.

Fernando Vicent Torres Serra

4. Estadísticas avanzadas del usuario

Ampliar las métricas registradas, permitiendo consultar:

- Horas totales viajadas.
- Número de países visitados.
- Cantidad de vuelos por compañía o aeropuerto.
- Vuelos con destinos repetidos.
- Trayectos más frecuentes.

Estas estadísticas darían al usuario una visión más rica de sus hábitos de viaje.

5. Etiquetas personalizadas y sellos

Ofrecer la opción de clasificar vuelos o reservas con stickers o etiquetas, como:

- “Trabajo”
- “Vacaciones”
- “Evento especial”
- “Viaje en grupo”

Estas etiquetas permitirían una mejor organización y segmentación del historial de viajes.

6. Expansión hacia servicios comerciales

En una fase más avanzada, se podría integrar un comparador de precios de vuelos, permitiendo buscar y comparar billetes de avión directamente desde la app. Esto añadiría valor añadido y convertiría la aplicación en una herramienta de planificación completa.

También se podrían implementar reseñas y puntuaciones para trayectos específicos, donde los usuarios puedan valorar sus experiencias y consultar opiniones de otros viajeros sobre:

- Comodidad del vuelo.
- Puntualidad.
- Servicio de la aerolínea.
- Experiencia general del trayecto.

7. Contenido comunitario sobre destinos

Añadir un módulo de documentación generado por la comunidad, donde los usuarios puedan:

- Ver recomendaciones de destinos.
- Leer guías rápidas (“Qué ver en Roma”, “Top 5 de París”...).
- Comentar sobre sus experiencias en ciudades o aeropuertos.

Esto enriquecería la utilidad de la app para quienes no solo desean registrar vuelos, sino también planificar sus viajes.

8. Registro de otros medios de transporte

Ampliar el tipo de trayectos que se pueden registrar en la app, incluyendo:

- Trenes
- Barcos
- Autobuses
- Viajes en AVE u otros sistemas ferroviarios de alta velocidad

Esto permitiría al usuario tener un registro de viajes más global y no limitado únicamente a vuelos.

9. Optimización de la API de vuelos

Actualmente, la aplicación utiliza la API de AviationStack, que presenta ciertas limitaciones (máximo de 100 peticiones al mes, restricciones en los datos disponibles, etc.). En el futuro se valorará:

- Migrar a otra API con mayor flexibilidad y capacidad.
- Contratar un plan premium de la API actual para ampliar el límite.
- Implementar mecanismos de cacheo y reutilización de datos ya consultados.

CONCLUSIÓN

A lo largo del desarrollo de este proyecto, se han presentado diversos retos técnicos y funcionales, algunos de los cuales han sido solucionados con éxito, mientras que otros han requerido adaptaciones o han quedado como limitaciones conocidas debido a restricciones externas.

Principales dificultades y soluciones

Una de las principales dificultades fue encontrar una forma viable de obtener datos reales de vuelos. Tras evaluar distintas alternativas, se optó por utilizar la **API de AviationStack**, que permite acceder a vuelos reales, pero con restricciones importantes en su versión gratuita:

- Límite de **100 peticiones al mes**, lo que impide escalar el uso libremente.
- Solo permite consultar vuelos ocurridos en los **dos días anteriores o el mismo día actual**.
- No es posible buscar vuelos por una fecha específica.

Estas limitaciones no pudieron eliminarse completamente, por lo que se tuvo que rediseñar parte de la lógica de la aplicación para trabajar únicamente con vuelos reales **registrados el mismo día del vuelo**. Esto reduce la flexibilidad para el usuario, pero garantiza que los datos sean verídicos.

Otra dificultad relevante fue un problema con las **zonas horarias al manejar fechas y horas** en la base de datos. Inicialmente, los horarios se almacenaban en formato español, sin considerar que los devolvía en otro huso horario. Este desfase generó problemas al actualizar y mostrar los datos correctamente. Aunque esta situación causó confusión durante un tiempo, se logró identificar la causa al comparar directamente las horas mostradas por la aplicación y las almacenadas en la base de datos. Finalmente, se ajustó el sistema para garantizar la correcta sincronización de horarios.

Aprendizajes alcanzados

A pesar de las limitaciones mencionadas, este proyecto ha supuesto un aprendizaje profundo en distintas áreas del desarrollo de software, destacando especialmente:

- **Integración con APIs externas** y adaptación a sus limitaciones.
- Uso de **Firebase** para la autenticación de usuarios y la gestión de una base de datos en tiempo real.
- Manejo de relaciones entre múltiples entidades (usuarios, reservas, vuelos).
- Diseño de una interfaz fluida y coherente basada en componentes reutilizables y el patrón Scaffold.
- Comprensión de problemas relacionados con **formato horario, sincronización y representación temporal**, especialmente relevantes en aplicaciones que manejan información internacional.

En resumen, aunque no todas las dificultades pudieron resolverse completamente debido a limitaciones externas (como la API gratuita), se logró construir una aplicación funcional, realista y con un alto grado de coherencia técnica. Las restricciones encontradas se han convertido en oportunidades de mejora para futuras versiones y han permitido consolidar conocimientos valiosos tanto técnicos como de diseño de producto.

BIBLIOGRAFÍA Y WEBGRAFÍA

AristiDevs. (s.f.). *Canal de YouTube*. YouTube. https://www.youtube.com/@AristiDevs/featured (consultado entre marzo y mayo de 2025)

AristiDevs. (2022, diciembre 4). *Curso Firebase + Jetpack Compose: Autenticación, Google Auth y Firestore* [Video]. YouTube. https://www.youtube.com/watch?v=LxABxtwhrDE (consultado entre marzo y abril de 2025)

AristiDevs. (2022, diciembre 10). *Jetpack Compose Navigation: Aprende a navegar entre pantallas fácilmente* [Video]. YouTube. https://www.youtube.com/watch?v=1OxiEaEWEe4 (consultado entre marzo y mayo de 2025)

ChatGPT. (2024). *Modelo de lenguaje asistido por inteligencia artificial para el aprendizaje y solución de problemas técnicos*. OpenAI. https://chat.openai.com/ (consultado entre marzo y mayo de 2025)

Firebase. (s.f.). *Documentación de Firebase*. Google. https://firebase.google.com/docs?hl=es-419 (consultado entre marzo y abril de 2025)

Firebase. (s.f.). *Autenticación con Firebase*. Google. https://firebase.google.com/docs/auth?hl=es-419 (consultado entre marzo y abril de 2025)

Google. (s.f.). *Desarrolladores Android – Documentación oficial*. https://developer.android.com/?hl=es-419 (consultado entre marzo y mayo de 2025)

Layer, M. (s.f.). *AviationStack API Documentation*. AviationStack. https://aviationstack.com/documentation (consultado entre marzo y abril de 2025)

Postman. (s.f.). *Documentación oficial de Postman*. https://learning.postman.com/docs/ (consultado entre marzo y mayo de 2025)

Square. (s.f.). *Retrofit – Type-safe HTTP client for Android and Java*. Square Open Source. https://square.github.io/retrofit/ (consultado entre marzo y abril de 2025)

Stack Overflow. (s.f.). *Stack Overflow – Comunidad de programadores para resolver dudas técnicas*. https://stackoverflow.com/ (consultado entre marzo y abril de 2025)

Wikipedia. (s.f.). *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Wikipedia:Portada (consultado entre marzo y abril de 2025)

Documentación académica (Aules)

- IES El Grao. (2024). *Programació multimèdia i dispositius mòbils* ,curso 2024/2025. Aules (consultado entre marzo y abril de 2025).
- IES El Grao. (2024). *Programació de servicis i processos* ,curso 2024/2025. Aules (consultado entre marzo y abril de 2025).
- IES El Grao. (2024). *Accés a dades* ,curso 2024/2025. Aules (consultado marzo y abril de 2025).
- IES El Grao. (2023). *Bases de dades*, curso 2023/2024. Aules (consultado entre marzo y abril de 2025).

ANEXO I

MANUAL DE INSTALACIÓN

1. Entornos de desarrollo necesarios

Para poder ejecutar, modificar o desarrollar el proyecto correctamente, es necesario contar con los siguientes entornos de desarrollo instalados en el sistema:

Visual Studio Code

Editor de código ligero pero potente, utilizado para desarrollar el servidor backend en Node.js. Se recomienda instalar la extensión de soporte para JavaScript/TypeScript y Node.js para una mejor experiencia de desarrollo.

Descarga: https://code.visualstudio.com/

pgAdmin

Herramienta de administración visual para bases de datos PostgreSQL. Se emplea para gestionar tablas, realizar consultas SQL y supervisar la base de datos utilizada por la aplicación.

Descarga: https://www.pgadmin.org/

Android Studio

IDE oficial de Android, utilizado para el desarrollo de la aplicación móvil con Kotlin y Jetpack Compose. Incluye un emulador para probar la app en un entorno virtual simulado.

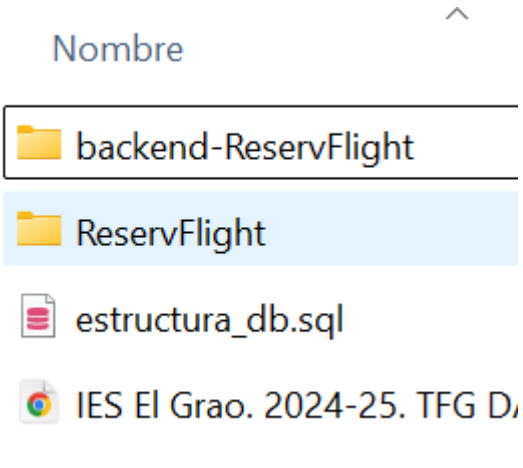
Descarga: https://developer.android.com/studio

2. Pasos de instalación

Paso 1: Archivos del proyecto

Dentro del dispositivo de almacenamiento proporcionado se incluyen los siguientes elementos esenciales para la instalación del sistema:

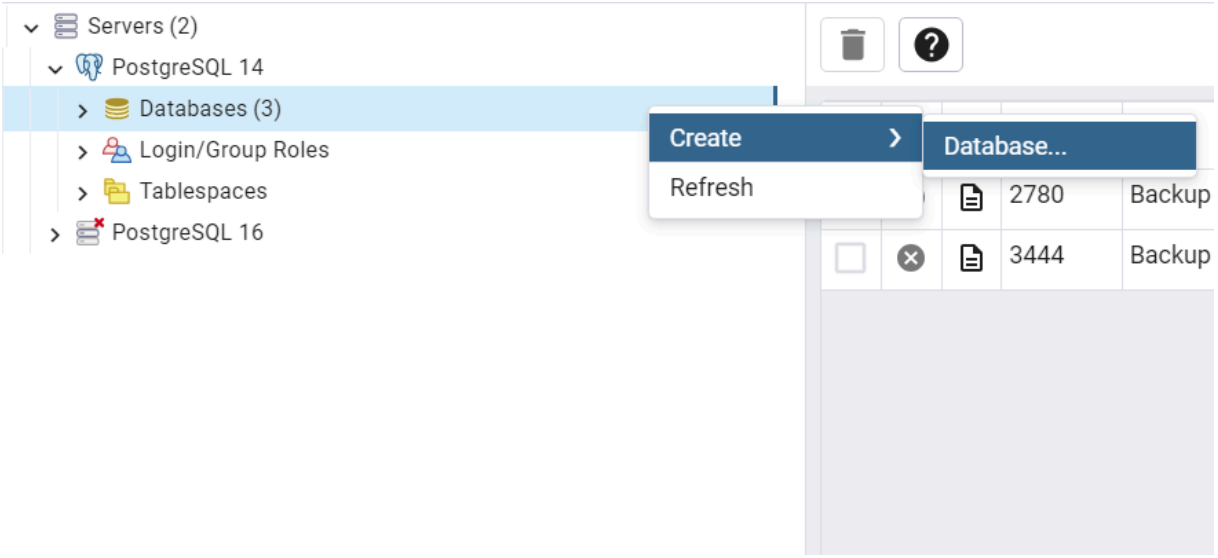
- 📁 **backend-ReservFlight/**: Contiene el código fuente del backend desarrollado con Node.js. Este directorio debe abrirse utilizando el entorno de desarrollo **Visual Studio Code**.
- 📁 **ReservFlight//**: Incluye el proyecto de la aplicación móvil implementada en Kotlin mediante **Android Studio**. Esta carpeta debe abrirse con dicho entorno de desarrollo.
- 📄 **estructura_db.sql**: Archivo SQL que define la estructura de las tablas necesarias en la base de datos PostgreSQL. Este fichero debe ser importado mediante **pgAdmin** para establecer correctamente el esquema de datos.



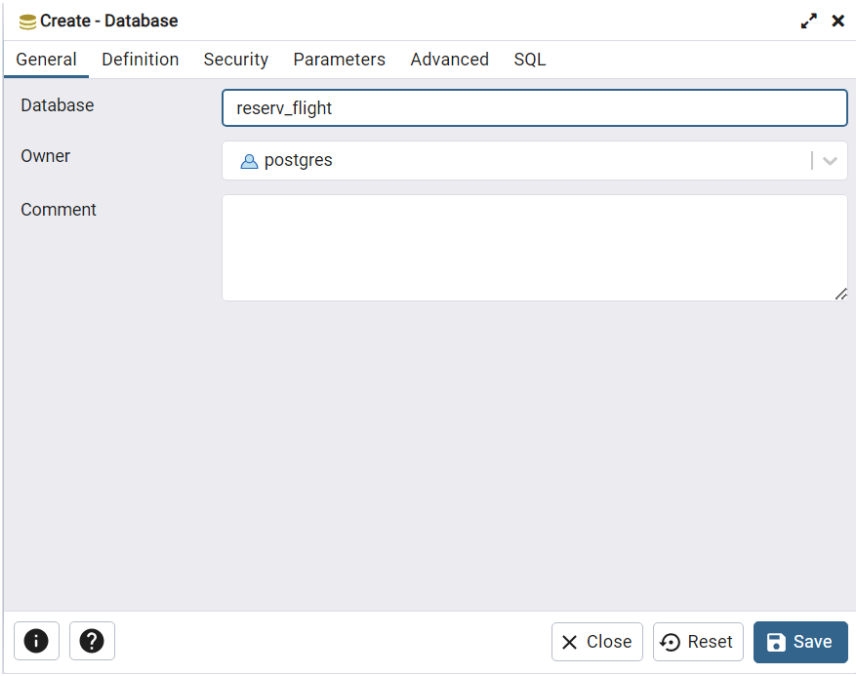
Paso 2: Crear la base de datos en pgAdmin

Abrir la herramienta **pgAdmin** e iniciar sesión con las credenciales correspondientes. A continuación, se debe realizar la creación de una nueva base de datos siguiendo estos pasos:

1. En el panel lateral izquierdo, hacer clic derecho sobre el servidor correspondiente (una vez conectado) y seleccionar la opción **"Create"** → **"Database..."**.



2. Asignar un nombre representativo a la base de datos, por ejemplo:reserv_flight.



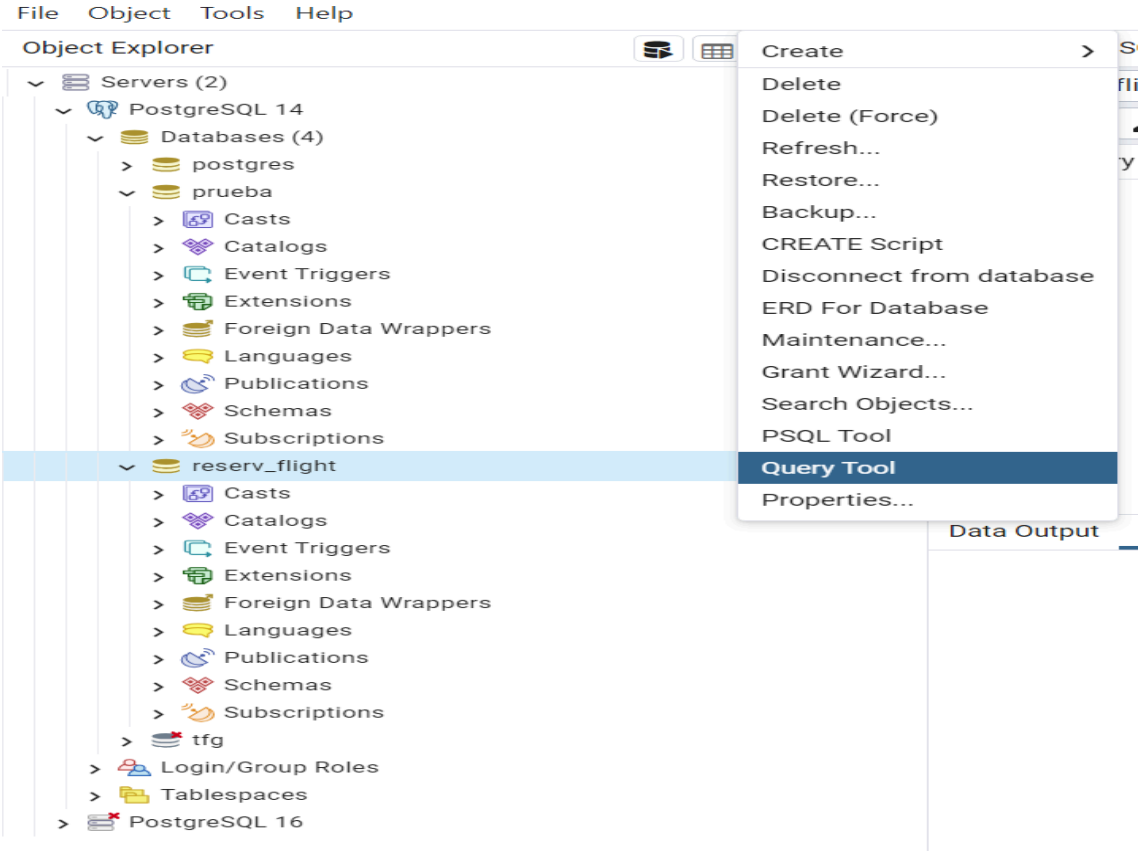
3. Confirmar la creación pulsando **Save**.

Esta base de datos será utilizada posteriormente para importar la estructura de tablas y establecer la conexión con el backend.

Paso 3: Importar la estructura de la base de datos

Una vez creada la base de datos, se debe importar su estructura mediante el archivo estructura_db.sql incluido en el pendrive. Para ello, seguir los siguientes pasos:

1. Hacer clic derecho sobre la base de datos recién creada en el panel izquierdo de **pgAdmin** y seleccionar **"Query Tool"**.



2. Se abrirá una ventana de consultas. En la barra superior, pulsar el icono de **abrir archivo** y seleccionar el archivo estructura_db.sql.

3. Una vez cargado el contenido del archivo en el editor, pulsar el botón **"Execute/Play"** para ejecutar el script.

Este proceso generará automáticamente todas las tablas necesarias para el funcionamiento de la aplicación.

Paso 4: Abrir el proyecto del backend en Visual Studio Code

El siguiente paso consiste en abrir el proyecto del backend para poder ejecutarlo. Para ello:

1. Abrir **Visual Studio Code**.

2. Seleccionar **"Archivo" > "Abrir carpeta..."** y buscar la carpeta llamada backend-ReservFlight incluida en el pendrive.

3. Una vez abierta la carpeta, se recomienda instalar las extensiones necesarias si Visual Studio Code lo solicita (por ejemplo, para Node.js o soporte de entorno).

Este proyecto contiene el servidor desarrollado con Node.js que actúa como intermediario entre la base de datos PostgreSQL, la API externa y la aplicación móvil.

Paso 5: Obtener la clave de API y configurarla en el servidor

Dentro del proyecto backend-ReservFlight, se debe localizar y abrir el archivo index.js. En este archivo se encuentra la línea:

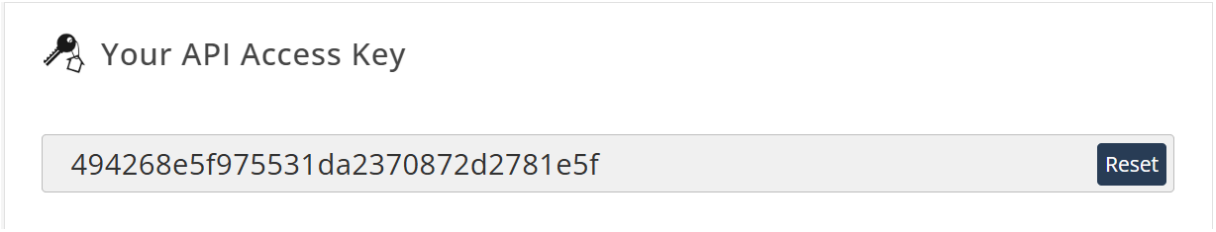
```
const API_KEY = "TU_API_KEY_AQUÍ";
```

```
const express = require("express");
const cors = require("cors");
const { Pool } = require("pg");
const axios = require("axios");

const app = express();
const port = 3000;
const { DateTime } = require("luxon");

// Middleware
app.use(cors());
app.use(express.json());

//Key de la api de vuelos
const API_KEY = "TU_API_KEY_AQUÍ";
// Configuración de PostgreSQL
const pool = new Pool({
  user: "postgres",
  host: "localhost",
  database: "reserv_flight",
  password: "postgres",
  port: 5433,
});
```

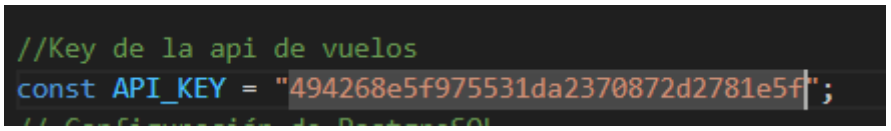
Esta clave (API_KEY) es necesaria para realizar peticiones a la API externa de información de vuelos.

Para obtener esta clave:

1. Acceder al sitio web oficial de Aviationstack: https://aviationstack.com/signup/free.
2. Registrarse con una cuenta gratuita.
3. Una vez completado el registro, la página principal mostrará una **API Key** personal.



4. Copiar dicha clave y pegarla entre comillas en la línea mencionada:
   `const API_KEY = "TU_API_KEY_AQUÍ";`



Esta clave permitirá que el servidor pueda comunicarse con la API y recuperar información en tiempo real sobre vuelos.

Paso 6: Configurar los datos de conexión a PostgreSQL

En el mismo archivo index.js, dentro del directorio backend-ReservFlight, se encuentra el bloque de configuración para la conexión con la base de datos PostgreSQL:

```
// Configuración de PostgreSQL

const pool = new Pool({

  user: "postgres",

  host: "localhost",

  database: "reserv_flight",

  password: "postgres",

  port: 5433,

});
```

Es necesario asegurarse de que los datos que aparecen en esta sección coincidan con los parámetros de la base de datos creada previamente en **pgAdmin**:

- user: nombre del usuario de PostgreSQL (por defecto suele ser postgres).
- host: normalmente se mantiene como localhost.
- database: nombre exacto de la base de datos creada anteriormente (por ejemplo, reserv_flight).
- password: contraseña del usuario PostgreSQL.
- port: el puerto en el que se ejecuta el servidor de PostgreSQL, comúnmente 5432 o 5433.

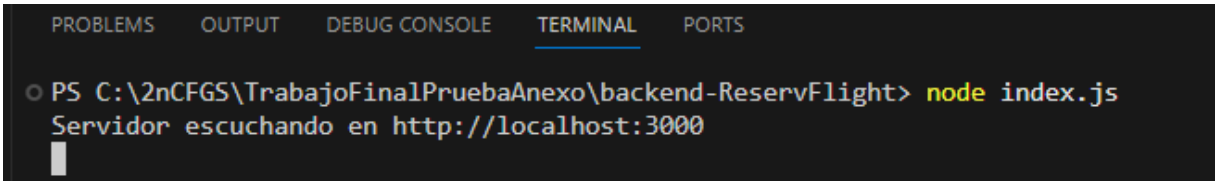
Una vez actualizados estos valores correctamente, el backend podrá conectarse a la base de datos sin problemas.

Paso 7: Iniciar el backend

Para iniciar el servidor backend, se debe abrir una terminal dentro del directorio backend-ReservFlight en Visual Studio Code y ejecutar el siguiente comando:

```
node index.js
```

Si la configuración es correcta, en la terminal aparecerá un mensaje indicando que el servidor está escuchando, lo cual confirma que el backend se ha iniciado correctamente y está listo para recibir peticiones.



Paso 8: Ejecutar la aplicación móvil

El siguiente paso consiste en abrir el directorio ReservFlightApp utilizando Android Studio. Una vez cargado el proyecto, se debe esperar a que finalice la sincronización de Gradle.

Posteriormente, para iniciar la aplicación, basta con pulsar el botón de ejecución (icono de ▶ verde) situado en la barra superior. Se recomienda tener configurado un emulador de Android para que la aplicación se despliegue correctamente.

Verificación del funcionamiento y uso posterior

Si la instalación se ha realizado correctamente, al ejecutar la aplicación en Android Studio se mostrará la pantalla de inicio). Esta pantalla confirma que la app se ha conectado correctamente al backend y que todo el sistema está operativo.

> **Nota importante**: Cada vez que se quiera utilizar la aplicación, será necesario iniciar manualmente el backend. Para ello, se debe abrir una terminal dentro de Visual Studio Code, posicionarse en el directorio del backend (backend-ReservFlight) y ejecutar el siguiente comando:
>
> ```
> node index.js
> ```

Si en la terminal aparece el mensaje *"Servidor escuchando"*, significa que el backend está activo y listo para recibir peticiones.

FERNANDO VICENT TORRES SERRA

Firmado digitalmente por FERNANDO VICENT TORRES SERRA
Fecha: 2025.05.30 10:15:52 +02'00'

Fernando Vicent Torres Serra