Documentación Taco Paco

Santi Martínez

November 15, 2025

Contents

1 Introducción **3**

2 Arquitectura de la app **4**
2.1 Diagrama de flujo 4
2.2 Diagrama de casos de uso 5

3 APP Móvil **6**

4 APP Escritorio **8**
4.1 Inicio app 8
4.2 Control de mesas. CASO 1: Todas las mesas libres 9
4.3 Control de mesas. CASO 2: Mesas ocupadas 10

5 API Rest **11**
5.1 Flujo de la API con las APPs 11
5.2 Componentes 12
5.2.1 models/ 13
5.2.2 node_modules/ 13
5.2.3 server (Endpoints) 13

6 Problemas encontrados **14**
6.1 Conexión entre las APPs y la API 14
6.2 Calls 14
6.3 Organización inicial 14
6.4 Desarrollo de frontend 14

7 Conclusión **15**

8 Bibliografía **16**

1 Introducción

Taco Paco es una aplicación de comida rápida en la que el cliente puede reservar su mesa, y a partir de ahí seleccionar sus productos favoritos.

Historia de Taco Paco

La historia de Taco Paco comienza en 1998, cuando dos amigos decidieron abrir un pequeño restaurante de comida rápida en el centro de la ciudad. Con el tiempo, el restaurante fue ganando popularidad gracias a su deliciosa comida y su ambiente acogedor. Hoy en día, Taco Paco es un referente en la comida rápida de la ciudad, y su aplicación móvil es una herramienta esencial para sus clientes.

Figure 1: Loco Taco Paco

El proyecto consta de dos aplicaciones principales: una app móvil para el cliente y una app de escritorio para el personal del establecimiento.

- La app de escritorio forma parte únicamente del establecimiento, ya que su funcionalidad radica en cobrar pedidos y liberar las mesas. Gracias a ello podrá ver también las mesas libres y las ocupadas.
- La app móvil está diseñada para el cliente, el cual podrá seleccionar su mesa y realizar sus pedidos a través de la carta.

Ambas se conectan entre sí a través de una **API Rest** elaborada con express y que se conecta a la base de datos con mongoose, para guardar y editar los cambios en Mongo Atlas.

A continuación se explicará todo lo relacionado con la arquitectura de la app, las funcionalidades de ambas aplicaciones y la API Rest, así como los problemas encontrados durante el desarrollo del proyecto.

2 Arquitectura de la app

2.1 Diagrama de flujo

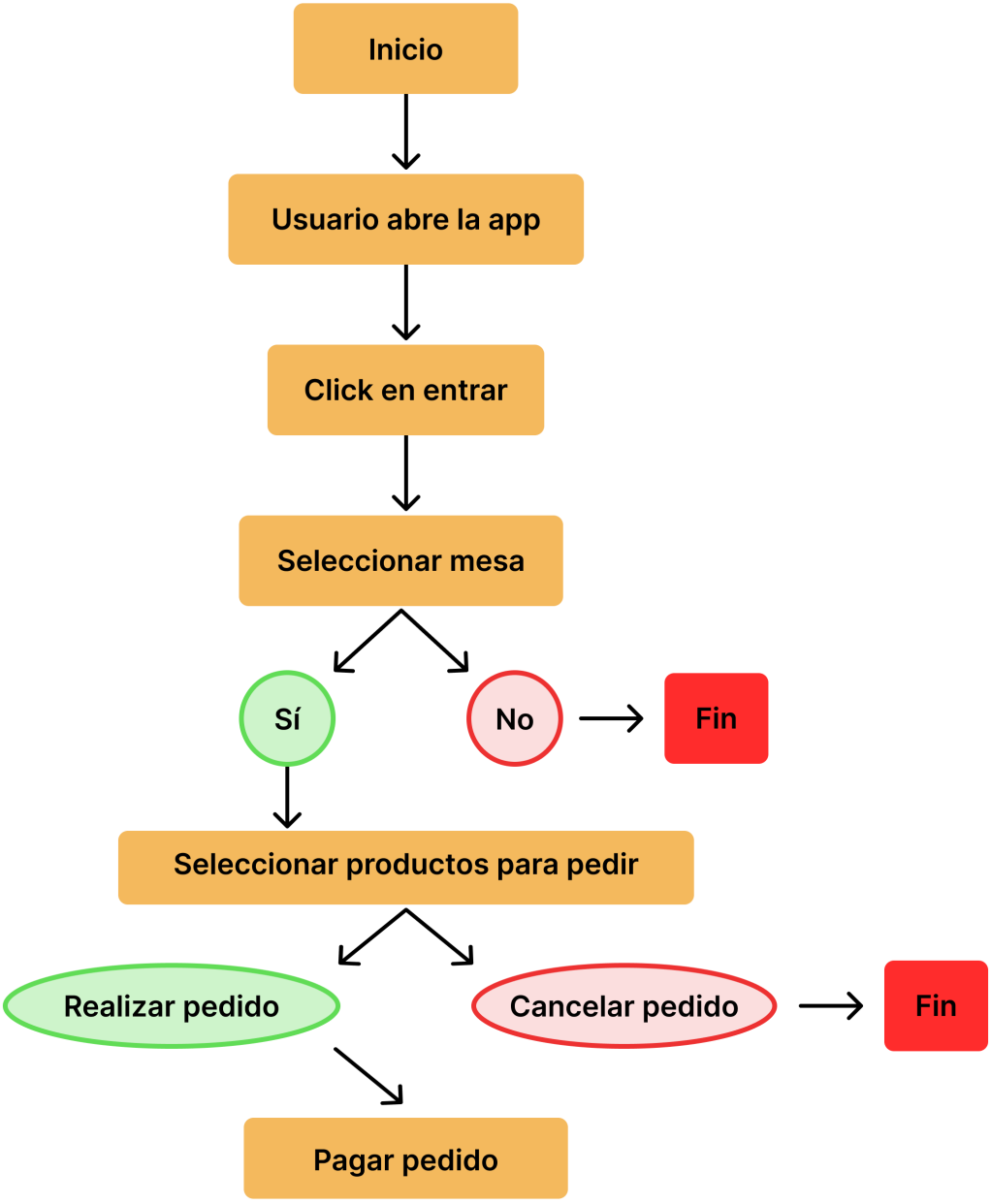


Figure 2: Diagrama de flujo

2.2 Diagrama de casos de uso

El **Diagrama de casos de uso** muestra las posibles utilizaciones de la aplicación por todos los actores*.

*Actor: Es el cliente y el servidor, los entes que pueden darle usos a la app.

En este caso el diagrama contiene dos **includes** por parte del cliente, que vienen a partir de la selección de productos dentro de la carta.

- Realizar Pedido
- Cancelar Pedido

Mientras que **extends** contiene uno solo por parte del Servidor:

- Librar mesas

Includes y extends

Include. Permite reutilizar funcionalidad común entre casos de uso, insertando obligatoriamente un caso dentro de otro para evitar duplicación.
Extend. Agrega comportamiento opcional o condicional a un caso de uso principal, ejecutándose solo bajo ciertas condiciones o escenarios específicos.

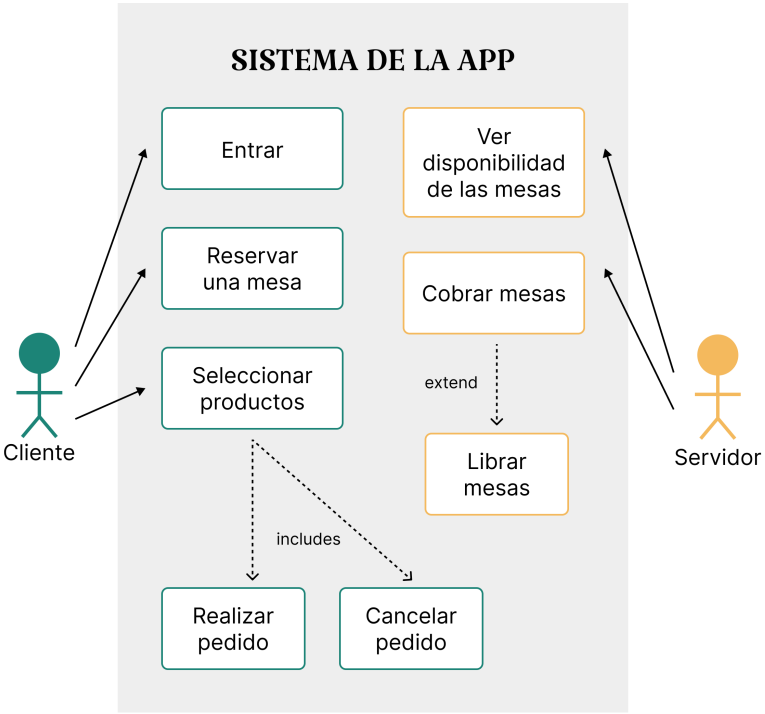


Figure 3: Diagrama de casos de uso

3 APP Móvil

Imágenes de la app

Pantalla principal

Primera pantalla que se encuentra el usuario al iniciarl la app, la única funcionalidad que tiene es la que le da el botón **Entrar**, el cual como su nombre indica, sirve para entrar en el siguiente menú de la app.

En la pantalla se encuentra el nombre de la empresa junto a su año en la que se fundó. También el botón de entrar que es el encargado de cambiar entre activities.

CASO 1: Todas las mesas libres

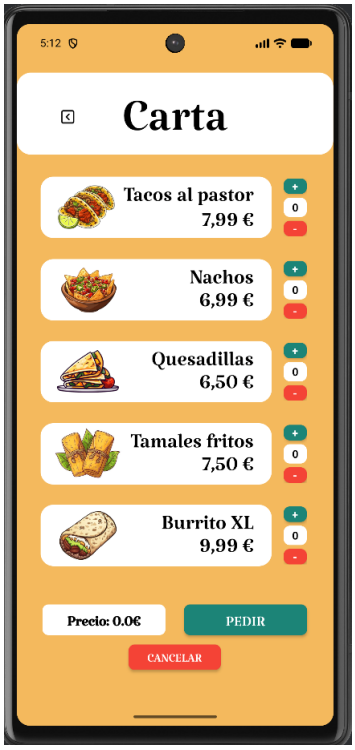
Esta es la pantalla en la que se muestran las mesas disponibles del establecimiento. Se trata de un menú con 5 mesas:

- 3 interiores
- 2 terraza

Esta diferenciación se encuentra en la parte baja de la pantalla, la cual se destaca con dos colores diferentes.
En pantalla se encuentran los botones que llevan a los activities de sus respectivas mesas y un botón para volver al activity anterior.

CASO 2: Mesas ocupadas

Mismo funcionamiento que en el caso anterior, pero sin embargo en esta pantalla se pueden apreciar que tanto la **Mesa 3** como la **Mesa 4**, se encuentran ocupadas.
La app no va a permitir su ocupación ni utilización de las mismas, es decir, quedan bloqueadas hasta que desde la APP-Escritorio se libreren cobrando el pedido a los clientes.



Carta

Esta es la pantalla en la que se muestran las mesas disponibles del establecimiento. Se trata de un menú con 5 mesas:

- 3 interiores
- 2 terraza

Esta diferenciación se encuentra en la parte baja de la pantalla, la cual se destaca con dos colores diferentes.
En pantalla se encuentran los botones que llevan a los activities de sus respectivas mesas y un botón para volver al activity anterior.

4 APP Escritorio

La app escritorio está diseñada para ser utilizada por el personal del establecimiento Taco Paco. Su función principal es gestionar las mesas del restaurante, permitiendo al personal ver qué mesas están ocupadas y cuáles están libres, así como tambíen tienen la facultad de limpiar las mesas y liberarlas una vez que los clientes han pagado su pedido.

4.1 Inicio app

Esta es la pantalla inicial de la app escritorio, en la que al pulsar **Entrar**, se abre el siguiente activity.

Cabe decir que es una app únicamente para el establecimiento de Taco Paco, ya que sus posibles usos son:

- Establecer un control de qué mesas están o no disponibles
- En caso de que no estén disponibles, se les puede cobrar su pedido desde esta app, lo que libera a continuación la mesa

Figure 4: Pantalla de inicio de la app escritorio

4.2 Control de mesas. CASO 1: Todas las mesas libres

Esta pantalla es la principal de la app. Como se puede observar contiene como contenido prioritario las 5 mesas des establecimiento. En este caso no hay ninguna mesa con un pedido pendiente.

Además, en la parte superior derecha se encuentra el botón de **Actualizar**, que sirve para refrescar el estado de las mesas en caso de que haya habido algún cambio desde la app móvil.

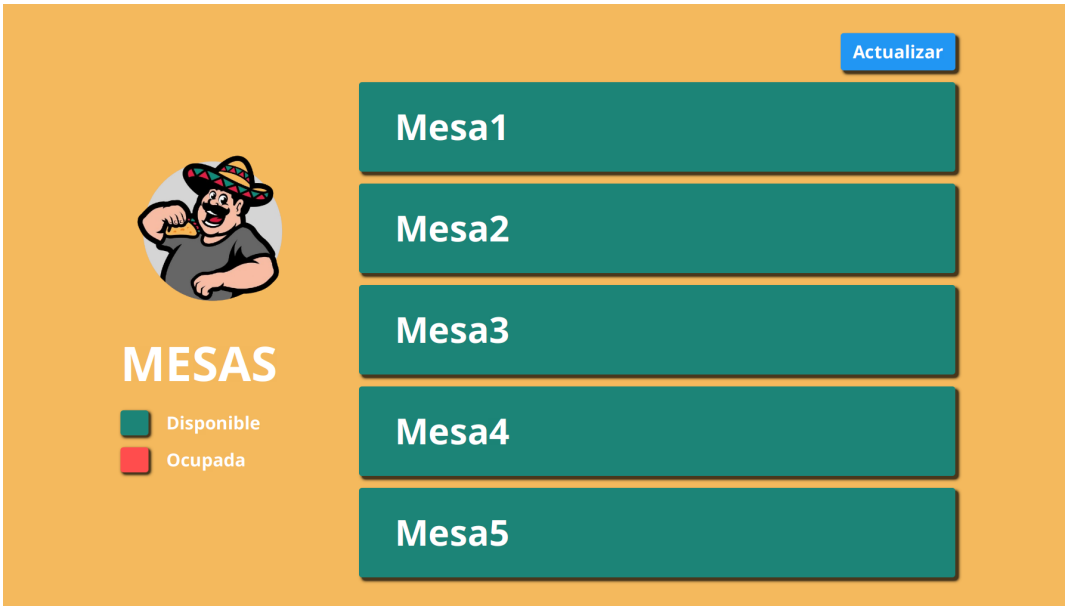


Figure 5: Todas las mesas disponibles

4.3 Control de mesas. CASO 2: Mesas ocupadas

Las mesas ocupadas se pueden apreciar por su fondo en rojo y tienen dos estados:

- **Mesa con pedido pendiente**: La mesa aparece de color rojo (ocupada) pero como el cliente todavía tiene la opción de seguir pidiendo y no ha pagado, no se puede liberar la mesa.
- **Mesa ocupada con pedido ya pagado**: En este caso, el cliente ya ha pagado su pedido desde la app móvil, pero el camarero no ha liberado la mesa. Al clicar en el botón **Limpiar mesa**, se libera la mesa y vuelve a estar disponible para nuevos clientes.

Una vez limpia la mesa, esta vuelve automáticamente a estar disponible en la app móvil, sin necesidad de actualizar nada.

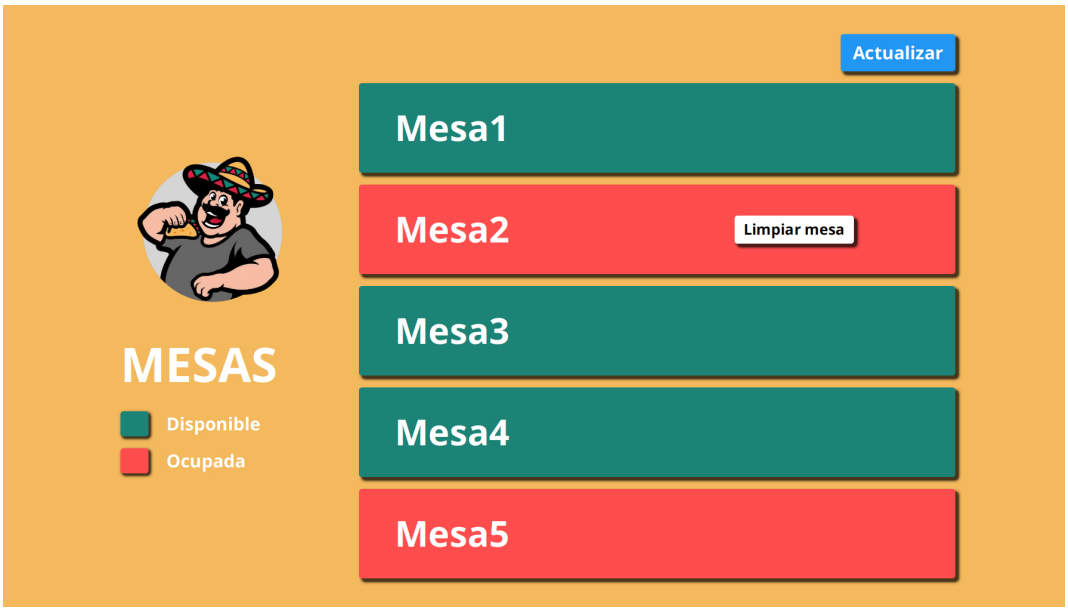


Figure 6: Mesas ocupadas en rojo

5 API Rest

5.1 Flujo de la API con las APPs

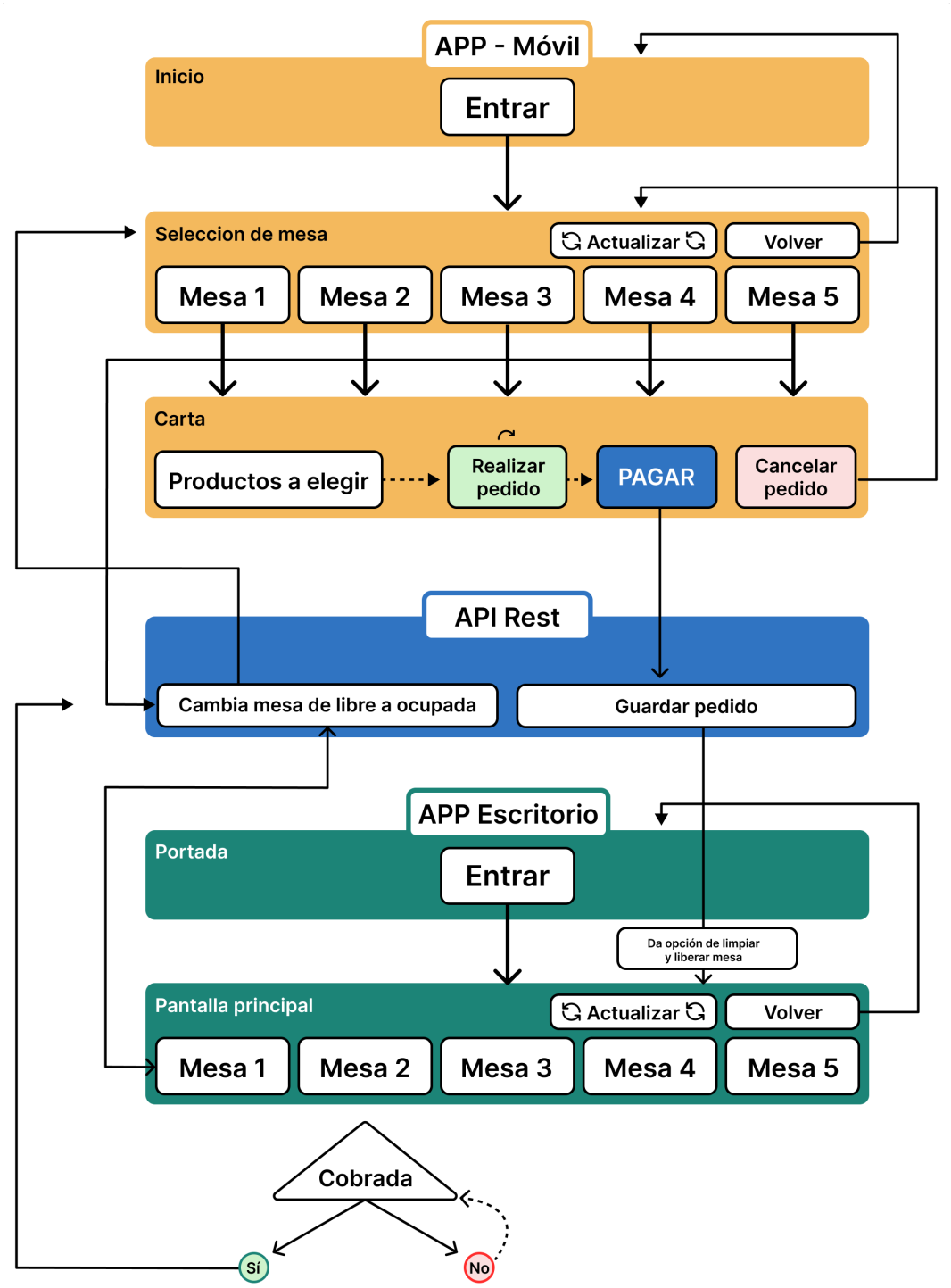


Figure 7: Diagrama de grafo

5.2 Componentes

Figure 8: Estructura de la API

5.2.1 models/

Explicación de lo que son los models

Endpoints

```
const mongoose = require('mongoose')
const mesaSchema = new mongoose.Schema(
    {
        nombre: { type: String, required: true, unique: true },
        ocupada: { type: Boolean, default: false }
    },
    { versionKey: false }
);
const Mesa = mongoose.model("Mesas", mesaSchema);
```

Endpoints

```
const mongoose = require('mongoose')
const pedidoSchema = new mongoose.Schema(
    {
        precioTotal: { type: Number, required: true }
    },
    { versionKey: false }
);
const Pedido = mongoose.model("Pedidos", pedidoSchema);
```

5.2.2 node_modules/

Conjunto de bibliotecas de node. Cada una de ellas cumplen una función en la API:

- **pnpm**: Paquete instalador de node elegido
- **dotenv**: Para la utilización de las variables de entorno guardadas en el .env
- **express**: Módulo con el que se construye la API
- **mongoose**: Módulo de Mongo que sirve para conectarse de forma sencilla a la base de datos de Mongo Atlas. A traves de los modelos creados anteriormente, se pueden guardar, actualizar y eliminar datos en la base de datos a través de los endpoints.

5.2.3 server (Endpoints)

6 Problemas encontrados

6.1 Conexión entre las APPs y la API

Uno de los principales problemas encontrados fue la conexión entre las aplicaciones y la API Rest. Al principio, iba más o menos todo bien, el problema llegó a raíz de un mal planeamiento inicial de la arquitectura de la app.

Al tener que modificar los modelos de **mesa** y **pedido** varias veces, se generaron conflictos entre las aplicaciones y la API, ya que por ejemplo, al principio la clase **mesa** no tenía el atributo **a_pagar**, lo que provocaba que el botón de **Limpiar mesa** no funcionara correctamente.

6.2 Calls

Otro problema fue la gestión de las peticiones asíncronas (los Calls). Tuve que pararme a entender bien el cómo trabajaban por detrás y darme cuenta de que como dije antes, son peticiones asíncronas; partiendo de eso y tras pararme con ello (no negaré cierta ayuda virtual...), conseguí que enlazara todo correctamente.

6.3 Organización inicial

Pero creo que el principal problema fue mi falta de organicación a la hora de crear la estructura de las apps; fue en un principio organizado, pero fui haciendo cambios a medida avanzaba el proyecto, y eso ocasionó el error anteriormente mencionado.

6.4 Desarrollo de frontend

Pese a que a mi parecer he mejorado un poco a nivel interfaz gráfica, sigue siengo algo que se me atasca un poco. Creo que también es cuestión de investigar un poco más.

En este proyecto empecé a inspirarme en imágenes de apps en pinterest, y a partir de ahí fui cogiendo pequeñas ideas. Este paso fue algo bueno para mi, ya que me dió una perspectiva de casos reales y el cómo tendría que ser este tipo de aplicación.

7 Conclusión

Este proyecto me fue un poco montaña rusa a nivel de apetencia de trabajo. Hubo días en los que me gustaba avanzar y romperme un poco la cabeza y otros en los que no me apetecía ni verlo, pero supongo que es algo común en proyectos "largos".

A estas alturas, respecto al anterior proyecto, me siento un poco más cómodo con las tecnologías utilizadas, ya sea por la acumulación de práctica en las mismas o por que está entrando **muy poco a poco** en mi zona de confort.

- **Android Studio**: Es una herramienta que va de la mano con lo que dije al principio de la conclusión, hay días que me siento muy cómodo y otros que pienso solamente en el logo de Android Studio y me apetece borrarlo del pc.
- **Javafx**: Lo mismo que la anterior, pero al ser, bajo mi opinión, mas sencilla a nivel cantidad de archivos, puede que no me de en ocasiones tanta pereza en x ocasiones.

Aunque parezca de momento que ambas las tiraría a la basura, creo que a medida que ha avanzado el curso me han ido gustando un poco más; sería la analogía a comer una hamburguesa con catarro, bien pero no me sabe.

En cuanto a la API, es la primera vez que hago modular el tema de los modelos de mongo, y algún problema que otro me ha dado, el cual tuve que solicitar ayuda a la IA para averiguar ciertor errores que ni viendo los logs pude resolver.

Y ahora vamos por fin con el contenido del proyecto.

Me ha parecido un tema entretenido y con aplicaciones al mundo real, lo que fue parte del incentivo motivacional para llevarlo a cabo. Es la primera vez que hago dos apps para un mismo proyecto y la verdad es que ya se puede ver (bajo la perspectiva de mi nivel) ciertos resquicios de mundo real a nivel proyecto, obviamente con lmiitaciones.

Como conclusión final decir que lo que me ha ayudado bastante a ver con perspectiva el proyecto, fue el hablar con mis compañeros de clase de problemas que nos surgían, cosa que tendría que empezar a hacer más a menudo, ya que muchas veces me encierro en mi mundo, pongo cascos y tiro para adelante.

8 Bibliografía

- **Formatos de imagen de Latex**: https://manualdelatex.com/tutoriales/figuras
- **Imagen al lado de texto en Latex**: https://foro.rinconmatematico.com/index.php?topic=114810.0
- **Texto al rededor de figuras en Latex**: https://es.wikibooks.org/wiki/Manual_de_LaTeX/Inclusi%C3%B3n_de_gr%C3%A1ficos/Texto_alrededor_de_figuras
- **Insertar código en el pdf**: https://trspos.com/insertar-bloque-de-codigo-latex/
- **Dependencias retrofit** :https://square.github.io/retrofit/download/
- **Cuadros en latex**: https://www.youtube.com/watch?v=ZN2QC1pcjMI