Programación Multimedia y Dispositivos Móviles

Documentación Taco Paco

Aplicaciones de Móvil y Escritorio

Autor:

Santi Martínez

16 de noviembre de 2025

Índice

1 Introducción **2**

2 Arquitectura de la app **3**
2.1 Diagrama de flujo ... 3
2.2 Diagrama de casos de uso ... 4
2.3 Especificaciones Técnicas ... 5

3 APP Móvil **6**
3.1 Imágenes de la app ... 6

4 APP Escritorio **9**
4.1 Inicio app ... 9
4.2 Control de mesas. CASO 1: Todas las mesas libres ... 10
4.3 Control de mesas. CASO 2: Mesas ocupadas ... 11

5 API Rest **12**
5.1 Flujo de la API con las APPs ... 12
5.2 Componentes ... 13
5.2.1 models/ ... 13
5.2.2 node_modules/ ... 13
5.2.3 Endpoints ... 13
5.3 Base de Datos ... 14
5.3.1 Colecciones ... 14
5.3.2 Estructura de colecciones ... 14

6 Futuras mejoras **15**

7 Problemas encontrados **16**
7.1 Mesas ocupadas permanentes ... 16
7.2 Conexión entre las APPs y la API ... 16
7.3 Calls ... 16
7.4 Organización inicial ... 16
7.5 Desarrollo de frontend ... 16
7.6 Package babel Error: Unknown option 'spanish' ... 16

8 Conclusión **17**

9 Bibliografía **18**

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.

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.

Figura 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

El diagrama de flujo señala, como su propio nombre indica, el flujo estándar de la aplicación vista desde la perspectiva de un cliente, es decir, la finalidad que tiene el programa para el público.

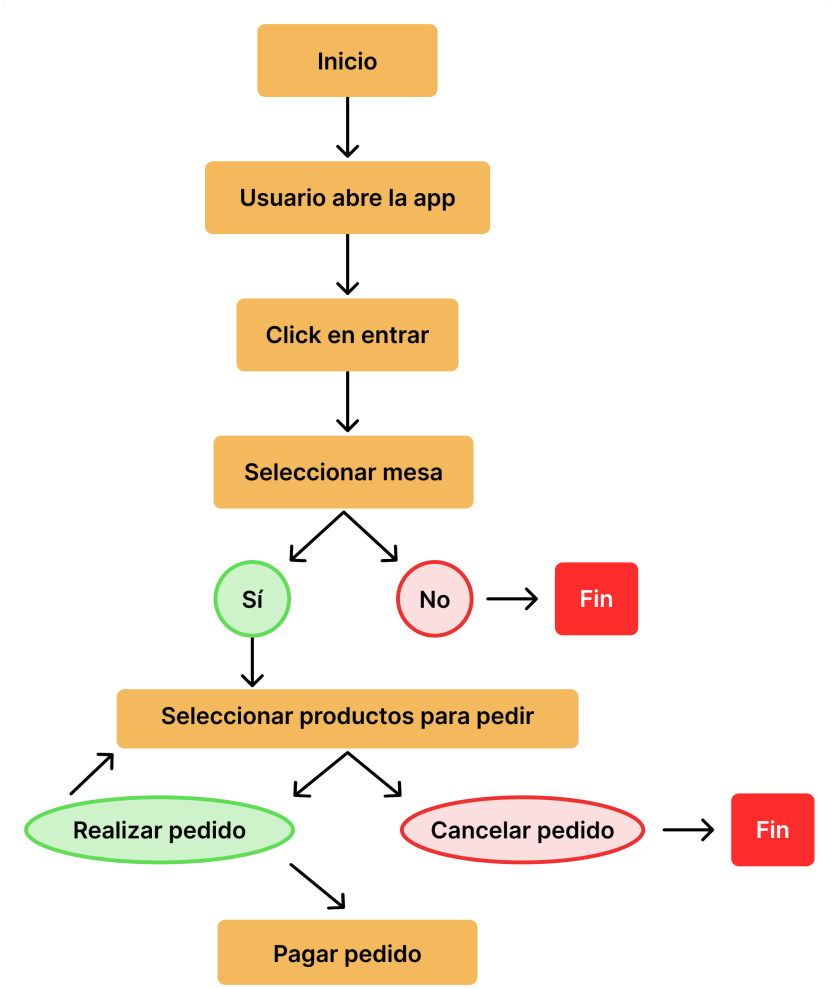


Figura 2: Diagrama de flujo

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

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.

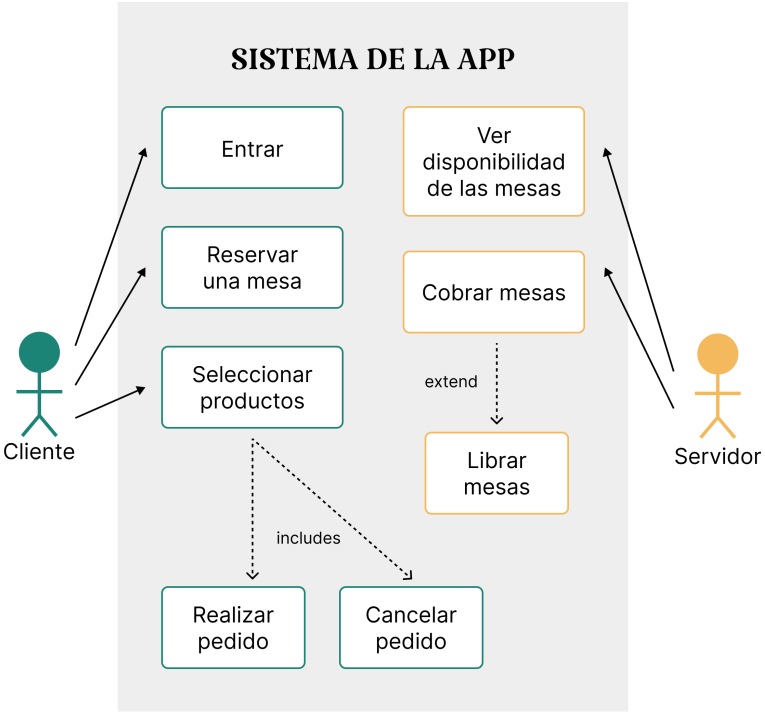


Figura 3: Diagrama de casos de uso

2.3. Especificaciones Técnicas

En esta sección se detallan las tecnologías, versiones y herramientas utilizadas en el desarrollo del proyecto Taco Paco, incluyendo frameworks, librerías y entornos de cada componente del sistema.

Aplicación Móvil

- **Lenguaje:** Java 17
- **SDK Android:** API 34 (Android 14)
- **Librerías principales:**
  - Retrofit - Consumo API REST
  - Gson - Procesamiento JSON

Aplicación Escritorio

- **Lenguaje:** Java 17
- **Framework:** JavaFX 21.0.1
- **Librerías principales:**
  - Retrofit - Consumo API REST
  - Gson - Procesamiento JSON

API REST

- **Runtime:** Node.js
- **Framework:** Express.js
- **Base de Datos:** MongoDB (Atlas Cloud)
- **ODM:** Mongoose
- **Puerto:** 3000
- **Instalador de paquetes:** pnpm

Herramientas de Desarrollo

- Android Studio
- IntelliJ IDEA 2023.2.5
- Visual Studio Code
- Git

3. APP Móvil

La aplicación de móvil está enfocada en el cliente de Taco Paco.

El cliente puede elegir en que mesa se va a sentar para comer, entre una lista de 5 mesas; tres de ellas interiores y otras dos de terraza.

Una vez elegida la mesa, el cliente puede elegir entre los platos de la carta y realizar el pedido o cancelarlo. En el caso de realizar un pedido, el cliente puede volver a hacer otro pedido en la misma mesa, y una vez terminado, paga y se marcha.

3.1. 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.

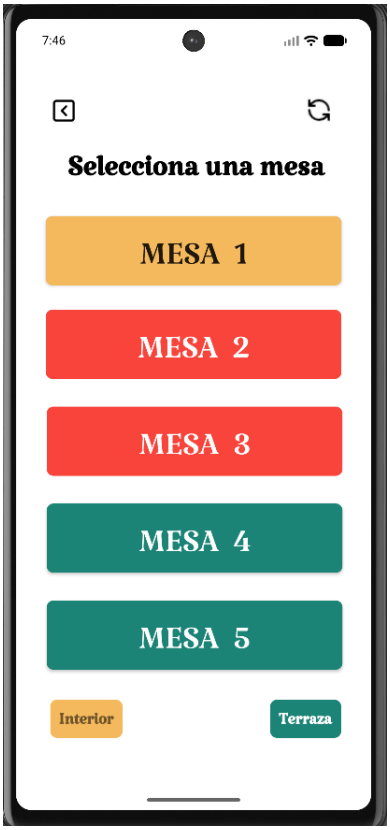
En la pantalla se encuentra el nombre de la empresa junto a su año en la que se fundó. También el botón de entrar que es el encargado de cambiar entre activities.

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:

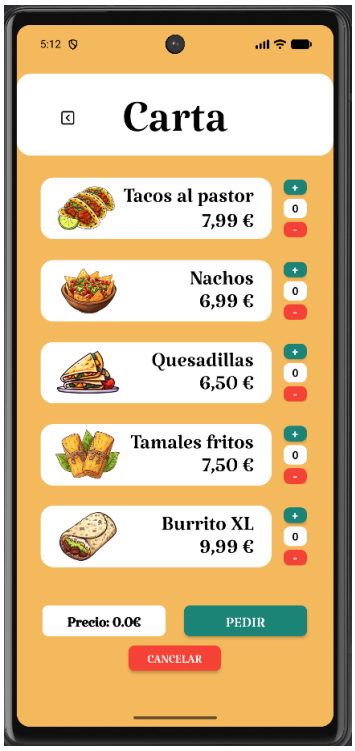
- 3 interiores
- 2 terraza

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.

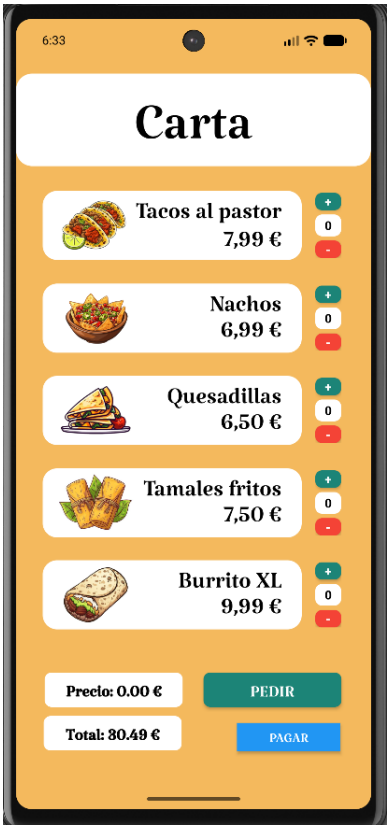


Carta

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.



Pagar pedido/s

En esta pantalla se puede observar que aparece un botón azul de **Pagar**, el cual se presiona al finalizar la comida

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

Figura 4: Pantalla de inicio de la app escritorio

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

La pantalla principal muestra las 5 mesas, indica su estado, permite actualizar con un botón y facilita controlar disponibilidad y pedidos, optimizando el flujo del establecimiento.

Esta pantalla es la principal de la app. Como se puede observar, contiene como contenido principal las 5 mesas del establecimiento. Cada mesa se muestra de manera clara y visual, permitiendo al trabajador identificar rápidamente su estado actual.

En este caso, no hay ninguna mesa con un pedido pendiente, lo que indica que todas las mesas están libres o que los pedidos anteriores ya han sido gestionados.

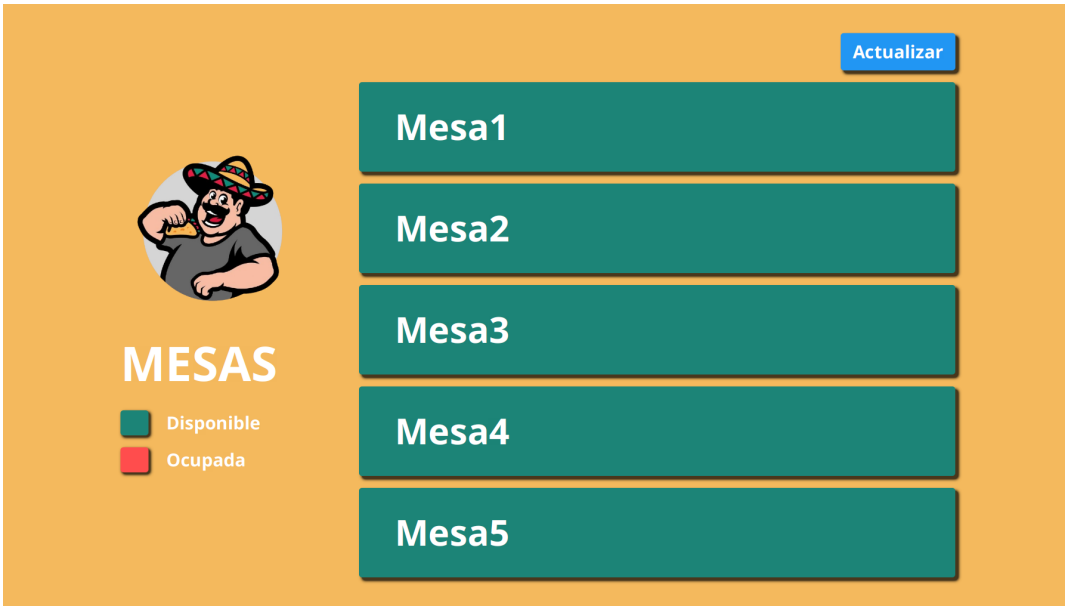


Figura 5: Todas las mesas disponibles

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. Esto garantiza que la información mostrada esté siempre actualizada.

De esta manera, los trabajadores pueden tener un control completo sobre la disponibilidad de las mesas y el progreso de los pedidos, optimizando así el flujo de trabajo dentro del establecimiento.

4.3. Control de mesas. CASO 2: Mesas ocupadas

Las **mesas rojas** indican ocupación: con pedido pendiente no se liberan; pagadas, se pueden limpiar y volver disponibles automáticamente.

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.

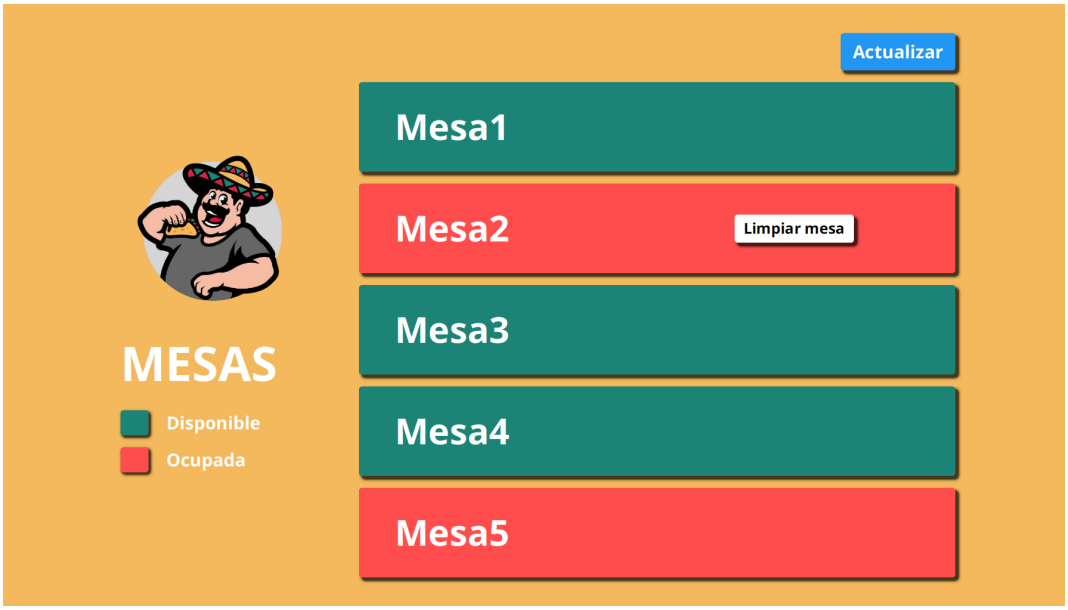


Figura 6: Mesas ocupadas en rojo

Una vez que la mesa ha sido limpiada, esta vuelve automáticamente a estar disponible tanto en la app de escritorio como en la app móvil. No es necesario realizar ninguna actualización manual, lo que garantiza que la información se mantenga sincronizada en tiempo real y facilita un flujo de trabajo más ágil para los trabajadores del establecimiento.

Esto permite que los clientes puedan ocuparla de inmediato, evitando conflictos y mejorando la eficiencia en la gestión entre aplicaciones.

5. API Rest

5.1. Flujo de la API con las APPs

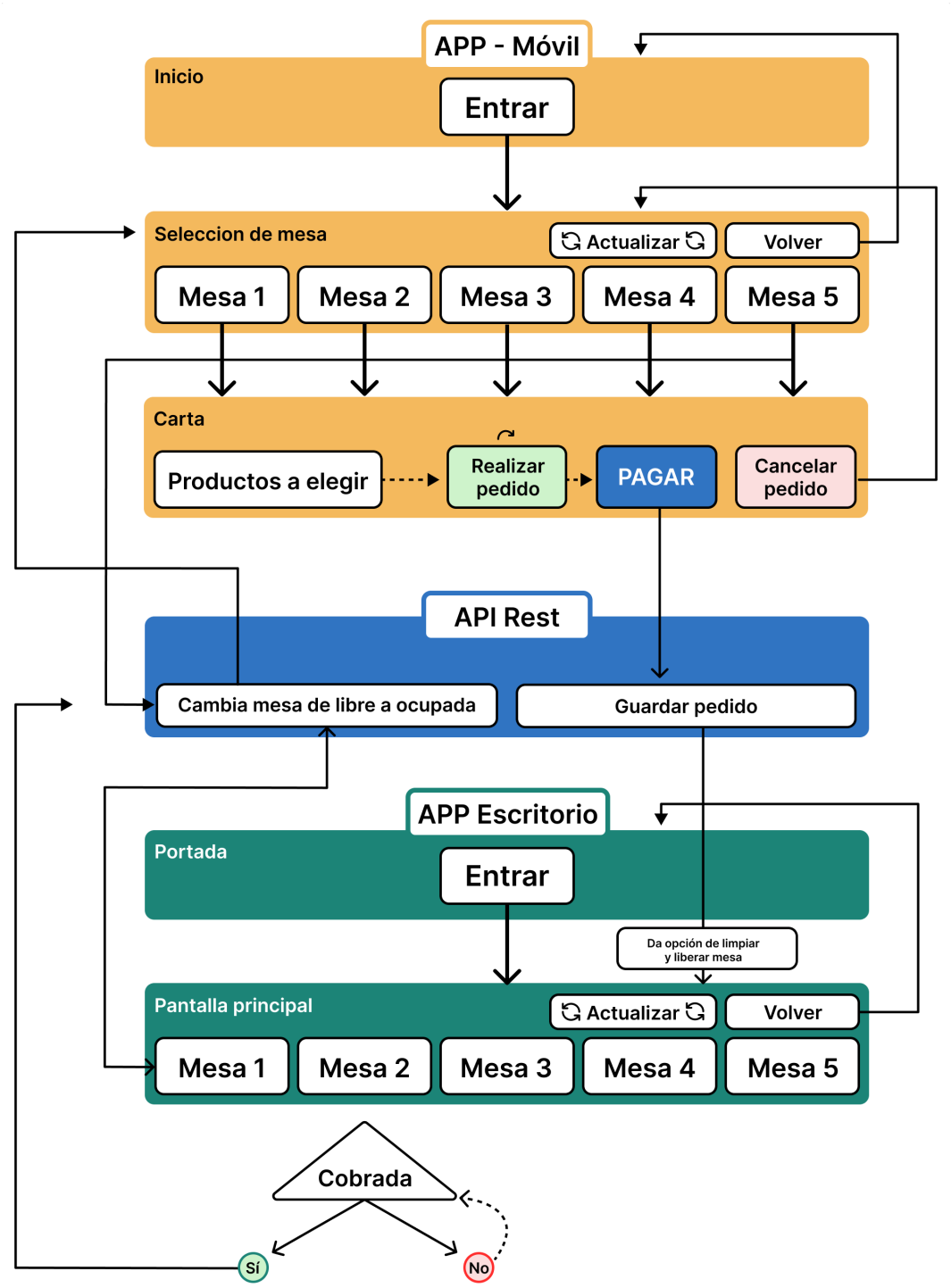


Figura 7: Diagrama de grafo

5.2. Componentes

Figura 8: Estructura de la API

5.2.1. models/

- **Mesa.js**: Modelo de mesa, contiene los atributos que tiene una mesa en la base de datos.
- **Pedido.js**: Modelo de pedido, contiene los atributos que tiene un pedido en la base de datos.

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. Endpoints

Son los puntos en los que la API, a través de una llamada desde la app, edita, lista, guarda o borra datos de la base de datos.

Endpoints de mesas

- GET: Para listar las mesas
- PUT: Para editar las mesas en la base de datos

Endpoints de pedidos

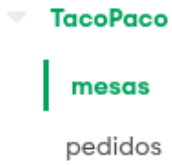
- GET: Para listar los pedidos
- POST: Para guardar los pedidos en la base de datos

5.3. Base de Datos

MongoDB Atlas

El sistema utiliza MongoDB Atlas como gestor de base de datos en la nube, lo que va a otorgar una sincronización de datos entre las aplicaciones móvil y de escritorio a través de la API REST.

5.3.1. Colecciones



5.3.2. Estructura de colecciones

Mesas Almacena el estado y la información de cada mesa de Taco Paco. Sus atributos son:

```
_id: ObjectId('69037eb7701776f90511d08c')
nombre : "Mesa1"
ocupada : false
a_pagar : false
```

- **_id**: Id que otorga Mongo
- **ocupada**: Indica si la mesa está siendo utilizada por un cliente
- **a_Pagar**: Señala si el cliente ya ha realizado el pago de su pedido

Pedidos Registra todos los pedidos realizados por los clientes. Sus atributos son:

```
_id: ObjectId('69186035557eab3e9b0518aa')
precioTotal : 19.98
```

- **_id**: Id que otorga Mongo
- **preciototal**: Precio total del pedido

6. Futuras mejoras

El proyecto Taco Paco cumple con los objetivos establecidos, pero hay varios aspectos en los que se puede escalar a algo mucho más completo.

Funcionalidad

- **Sistema de autenticación:** Login diferenciado para clientes, camareros y administradores
- **Gestión de menú:** Permitir añadir, editar y eliminar productos desde la app de escritorio

Aspectos técnicos

- **Evitar el estar tiempo inactivo con mesa ocupada:** Implementar timeout automático tras inactividad

Interfaz de usuario

- **Modo oscuro:** Implementar tema claro/oscuro
- **Responsive:** Adaptación a tablets y diferentes resoluciones
- **Varios idiomas:** Soporte para varios idiomas

Nuevas funcionalidades

- **Valoraciones:** Sistema de reseñas con valoraciones de la experiencia y comida
- **Descuentos:** Puntos y descuentos para clientes frecuentes
- **Gestión de inventario:** Control de stock e ingredientes

7. Problemas encontrados

7.1. Mesas ocupadas permanentes

Casi rematando el proyecto me dí cuenta de que si el usuario entraba en una mesa, y este cerraba la app, la mesa en la app de escritorio quedaba marcada como ocupada, ya que ni se llegó a cancelar ni se hizo pedido.

Para solucionar este "limbo."estuve viendo varias funciones que trabajan sobre los activities como onPause(), onStop(), onDestroy(),... pero me rompían el programa, ya que cada notificación saliente que aparece en el programa la interpretaba como una pausa o parón, y por como lo utilizaba me rompía la base de datos.

ACLARACIÓN: Existe un botón de Reiniciar mesas en la pantalla de escritorio, pero no formaría parte de la app final, ya que sirve para mejorar la exposición del proyecto, en el sentido de que si sales del simulador de Android, la mesa se queda colgada; y por esta razón agregué esta funionalidad, para poder restablecer las mesas (mediante confirmación se seguridad) y seguir con la presentación

7.2. 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.

7.3. 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.

7.4. 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.

7.5. 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.

7.6. Package babel Error: Unknown option 'spanish'

No me cargaba la documentación, y a través de los logs, me aparecía el mensaje del título del problema; esto me decía que mi Latex no tenía instalado el paquete de idioma español, el cual fue un paquete que implementé dentro del .tex durante el transcurso del proyecto.

8. 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 ciertos errores que ni viendo los logs pude resolver.
Y hablando de IA tambíen tuve que utilizarla en pequeñas partes que no sabía como resolver (Ejemplo: APP Móvil → EleccionMesa.java → Líneas 89 y 90, esto para formatear una variable y dividir el resultado de postformateo y preformateo para diferentes lados)

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.

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.

Extra Este párrafo lo estoy escribiendo 2 días antes de presentar la defensa, un sábado a las 23.25 de la noche. El trabajo está ya terminado. Puedo decir que una vez visto todo finalizado, pese a ser un trabajo de absoluto novato, se siente un ligero orgullo personal, sobre todo por el entendimiento de cosas que hace un mes me generaban muchas dudas.

9. Bibliografía

- **Formatos de imagen en LaTeX**: manualdelatex.com/tutoriales/figuras
- **Imagen al lado de texto en LaTeX**: foro.rinconmatematico.com
- **Texto alrededor de figuras en LaTeX**: es.wikibooks.org
- **Insertar código en PDF**: trspos.com
- **Claude.ai**: Claude
- **Formateo de decimales en Java**: es.stackoverflow.com
- **Dependencias Retrofit**: square.github.io/retrofit/download
- **Cuadros en LaTeX**: YouTube
- **Métodos `onPause()`, `onStop()`, `onDestroy()` en Android**: developer.android.com