Oishii Sushi (APP Móvil y Escritorio)

Documentación de Arquitectura de Software

Jacobo Pérez de Torres

Índice

1. Introducción

Oishii Sushi es un conjunto de aplicaciones (Móvil y Escritorio) desarrollada como práctica para las asignaturas de Acceso de Datos, Programación multimedia y móvil y Desarrollo de Interfaces. La aplicación móvil permite seleccionar una mesa del restuarante, hacer pedidos desde la carta y tramitar el pago de la cuenta. La aplicación de escritorio, recibe las comandas de los pedidos realizados desde la APP y tramitarlas. Ambas aplicaciones se comunican a traves de un backend con Express y MongoDB Atlas.

2. Arquitectura de la Aplicación

La aplicación se organiza siguiendo una arquitectura por capas que separa la interfaz de usuario, la lógica de negocio y el manejo de datos (Similar al modelo vista-controlador).

- **Capa de interfaz (UI):** Activities de Android Studio en XML y vistas de JavaFX FXML.
- **Capa de lógica:** Gestiona el envío de datos entre vistas y activities, la lógica del negocio (Platos, comandas, carrito, cuenta...) y coordina la base de datos.
- **Capa de datos:** Contiene el acceso a la base de datos y a la API.

3. Diseño de la base de datos (MongoDB)

En el backend se definen las siguientes colecciones de Items:

- Comandas:

```
numeroMesa: Number,
pedidoPlatos: Array,
atendidaComanda: Boolean
```

- Platos:

```
nombrePlato: String,
precioPlato: Number,
categoriaPlato: String,
unidadesPlato: Number
```

- Mesas:

```
numeroMesa: Number,
ocupadaMesa: Boolean,
carritoMesa: Array
```

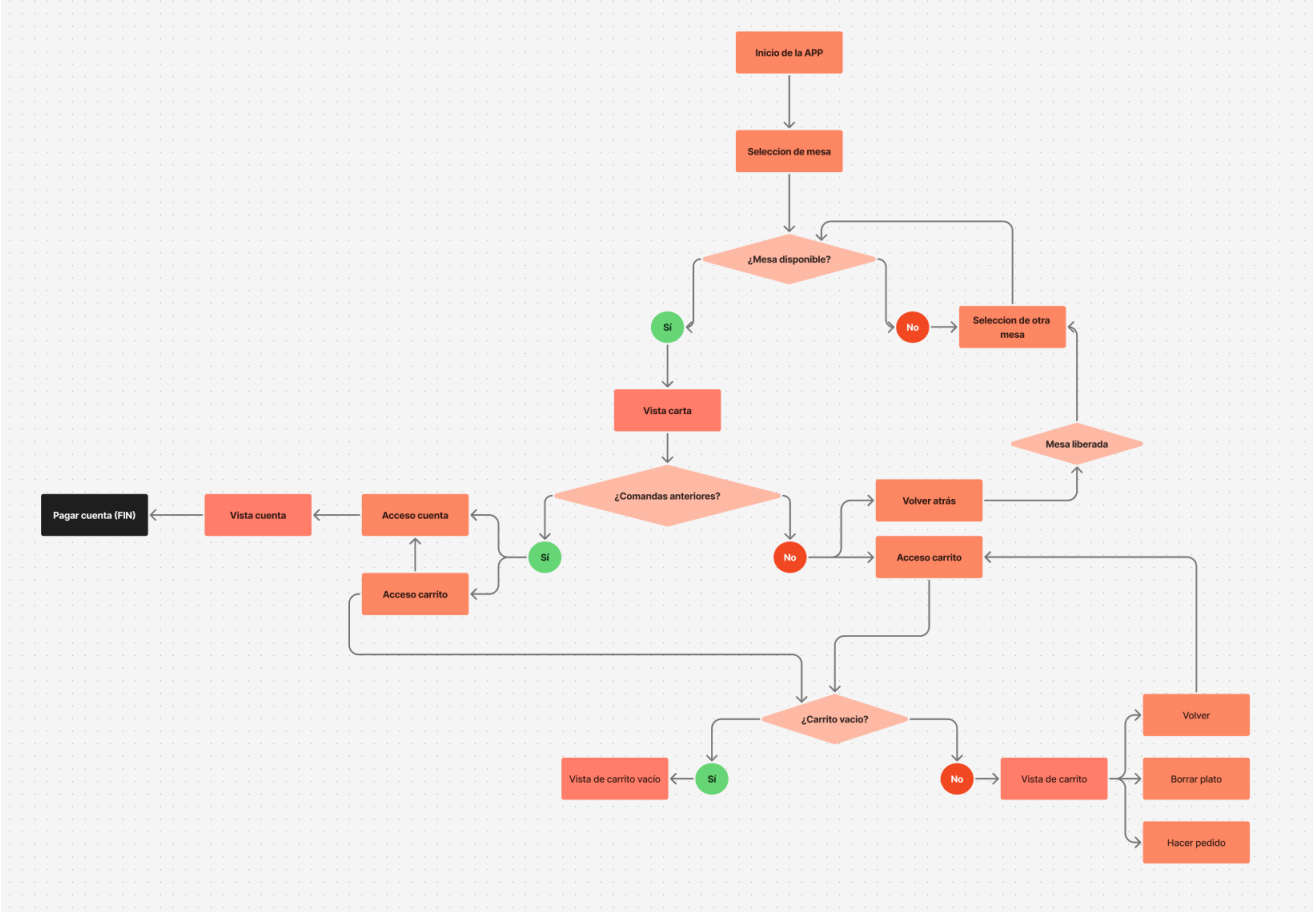
4. API

La API sigue un modelo RESTful. Con los siguientes endpoints:

Método	**Endpoint**	**Descripción**	**Parámetros**
GET	/platos	Recibe la lista de platos	N/A
GET	/comandas	Recibe la lista de comandas	N/A
GET	/mesas	Recibe la lista de mesas	N/A
POST	/platos	Añade un nuevo plato (Sin uso actual)	N/A
POST	/comandas	Añade una nueva comanda	N/A
POST	/mesas	Añade una nueva mesa	N/A
PUT	/platos	Actualiza un plato (Sin uso actual)	N/A
PUT	/comandas/:numeroMesa	Actualiza la información de una comanda	numeroMesa
PUT	/mesas/:numeroMesa	Acutaliza la información de una mesa	numeroMesa

5. Diagrama de flujo de la APP

A traves del siguiente diagrama se muestra el flujo de la aplicación móvil:



6. Tecnologías utilizadas

Para el desarrollo de la aplicación se han empleado las siguientes tecnologías:

6.1. Frontend App móvil (Android Studio)

- Lenguaje: Java.
- Interfaz: XML.
- Conexión con backend: Retrofit.

6.2. Frontend Desktop (JavaFX)

- Lenguaje: Java.
- Interfaz: FXML.
- Conexión con backend: Retrofit.

6.3. Servidor Backend

- Lenguaje: JavaScript (Node.js).
- Framework: Express.js.
- Base de datos: MongoDB.
- ODM: Mongoose.

7. Clases e Interfaces

En esta sección se muestra una lista de todas las clases existentes.

7.1. Clases principales (Móvil y Escritorio)

A continuación se describen las principales clases e interfaces que componen la aplicación:

- **Comandas** (entidad)
  - Contiene los atributos de una comanda: int numeroMesa, `List<Platos>` pedidoPlatos, boolean atendidoComanda.
  - Representa el modelo de datos de las comandas de la aplicación.
  - En la aplicación de escritorio esta clase varía añadiendo en atributos el id de mongo, para poder actualizar cada comanda.
- **Platos** (entidad)
  - Contiene los atributos de una comanda: String nombrePlato, double precioPlato, String categoriaPlato, int unidadesPlato.
  - Representa el modelo de datos de los platos de la aplicación.
- **Mesas** (entidad)
  - Contiene los atributos de una comanda: int numeroMesa, boolean ocupadaMesa, `List<Comandas>` comandasMesa, `List<Platos>` carritoMesa.
  - Representa el modelo de datos de las mesas de la aplicación.
- **Mesas** (entidad)
  - Contiene los atributos de una comanda: int numeroMesa, boolean ocupadaMesa, `List<Comandas>` comandasMesa, `List<Platos>` carritoMesa.
  - Representa el modelo de datos de las mesas de la aplicación.
- **APIService** (servicio)
  - Conecta con los endpoints del backend para establecer los metodos de tratado de datos.
- **ApiAdapter** (adaptador)
  - Se conecta con la URL del backend a traves de retrofit,

7.2. Clases exclusivas de Android Studio

- **MainActivity**
  - Controlador del layout `activity_main.xml`
  - Inicia la aplicación y permite acceder a selección mesas.
- **SeleccionMesa**
  - Controlador del layout `seleccion_mesa.xml`
  - Permite la selección de una mesa, comprueba cuáles están ocupadas, regula el acceso a mesas ocupadas y envía al usuario a Carrito.
- **Carta**
  - Controlador del layout `carta.xml`
  - Permite la selección de platos, retroceder para liberar la mesa y desplazarse al carrito.
- **ContenedorPlatosAdaptador**
  - Adaptador que controla el RecycleView de la clase Carta.
- **Carrito**
  - Controlador del layout `carrito.xml`
  - Permite visualizar el carrito con sus correspondientes platos y hacer un pedido.
- **ContenedorCarritoAdaptador**
  - Adaptador que controla el RecycleView de la clase Carrito.
- **CarritoVacio**
  - Controlador del layout `carrito_vacio.xml`
  - Permite visualizar el carrito vacio y regresar a la carta.
- **Cuenta**
  - Controlador del layout `ver_cuenta.xml`
  - Permite visualizar la cuenta total de pedidos, pagar y salir del restaurante.
- **ContenedorCuentaAdaptador**
  - Adaptador que controla el RecycleView de la clase Cuenta.

7.3. Clases exclusivas de JavaFX

- **HelloController**
  - Controlador del layout principal, contiene todos los métodos y funciones.

8. Problemas encontrados

Durante el desarrollo de las aplicaciones móvil y de escritorio se encontraron los siguientes problemas:

9. Problemas encontrados

Durante el desarrollo de las aplicaciones móvil y de escritorio surgieron varios problemas, algunos de ellos bastante curiosos y otros más técnicos, que fueron solucionados de diferentes maneras:

- **Actualización de datos en la API:** Al principio los cambios realizados en la aplicación no se reflejaban en el backend, lo que generaba confusión al probar los pedidos. *Solución:* Aprendí a usar correctamente los métodos PUT y PATCH con Retrofit para que los objetos se sincronizaran correctamente ([24, 2, 1]).
- **Carga de imágenes en RecyclerView:** Las imágenes de los platos a veces no se cargaban o se veían tarde, haciendo que la interfaz pareciera lenta. *Solución:* Implementé carga asíncrona en los ViewHolders para que la UI no se bloquease y todo se mostrara correctamente ([19]).
- **Eventos onClick en RecyclerView:** Al principio, los clics en los elementos personalizados no funcionaban bien. *Solución:* Moví los onClickListener dentro del adaptador y los pasé al Activity correspondiente para que respondieran correctamente ([19, 25]).
- **Compatibilidad de objetos en Android Studio:** Intentar pasar objetos entre Activities daba errores de tipo "no se puede castear a Parcelable". *Solución:* Implementé correctamente la interfaz `Parcelable` en los objetos que necesitaba transferir ([19]).
- **Conexión de JavaFX con NodeJS:** La aplicación de escritorio no lograba conectarse bien con la API, lo que hacía que los datos no se actualizaran. *Solución:* Usé Retrofit también en JavaFX y gestioné correctamente los hilos para evitar bloqueos ([31, 19]).
- **Visibilidad de elementos en Android Studio:** Algunos botones se veían bien en el editor gráfico pero no aparecían en el dispositivo al ejecutar la app. *Solución:* Comprobé las restricciones, la jerarquía de layouts y que los IDs coincidieran correctamente ([19, 26]).
- **Sincronización con MongoDB:** Me costó actualizar arrays de objetos y mantener los datos de pedidos y mesas sincronizados entre backend y frontend. *Solución:* Aprendí a usar correctamente los métodos de MongoDB Realm y probé varias veces PUT y POST hasta que los cambios se guardaban correctamente ([22]).

10. Bibliografía

Para llevar a cabo el desarrollo de esta documentación y siendo la primera vez que tenía que simular una API inexistente y documentar la arquitectura de un software, he

necesitado consultar diversas fuentes, desde StackOverflow hasta consultas sobre Latex a chatgpt. Todas las fuentes empleadas se citan a continuación:

Referencias

[1] Programacionymas. *Consumir una API usando retrofit*. Disponible en: `https://programacionymas.com/blog/consumir-una-api-usando-retrofit`.

[2] Geeksforgeeks. *Como actualizar datos en una API usando retrofit en Android Studio*. Disponible en: `https://www.geeksforgeeks.org/android/how-to-update-data-in-api-using-retrofit-in-android/`.

[3] StackOverflow. *Como fijar una imagen en un view holder de forma asíncrona*. Disponible en: `https://stackoverflow.com/questions/33575011/how-to-set-image-to-a-view-holder-asynchronously`.

[4] StackOverflow *Como definir una forma circular en Android Studio a traves de un archivo xml drawable*. Disponible en: `https://stackoverflow.com/questions/3185103/how-to-define-a-circle-shape-in-an-android-xml-drawable-file`.

[5] StackOverflow. *Como implementar un onClickListener en un viewholder customizado en mi RecyclerView*. Disponible en: `https://stackoverflow.com/questions/46640897/how-to-implement-onclicklistener-for-custom-viewholder-in-my-recycler-view`.

[6] StackOverflow. *Eliminar y refrescar en un RecyclerView*. Disponible en: `https://stackoverflow.com/questions/68780473/delete-and-refresh-in-recyclerview`.

[7] StackOverflow. *JavaFX Como hacer una imagen clickeable en Scene Builder*. Disponible en: `https://stackoverflow.com/questions/40495658/javafx-how-make-a-clickable-image-using-scenebuilder`.

[8] StackOverflow. *Como conectar un cliente de JavaFX a un servidor de NodeJS*. Disponible en: `https://stackoverflow.com/questions/58750418/connecte-javafx-client-to-a-nodejs-server`.

[9] StackOverflow. *Android Studio - Objeto no puede ser casteado a Parceable*. Disponible en: `https://stackoverflow.com/questions/41924068/object-cannot-be-cast-as-parcelable`.

[10] StackOverflow. *Android Studio - Como cambiar el color del borde de un SearchView*. Disponible en: `https://stackoverflow.com/questions/36888511/change-border-color-of-a-searchview`.

[11] StackOverflow. *Android Studio - Ejemplo simple de un RecyclerView*. Disponible en: `https://stackoverflow.com/questions/40584424/simple-android-recyclerview-example`.

[12] StackOverflow. *Métodos PUT y GET con router.put y router.delete no funcionan (express)*. Disponible en: `https://stackoverflow.com/questions/64842029/put-and-get-methods-with-router-put-and-router-delete-not-working-express`.

[13] StackOverflow. *Error "The requested module does not provide and export named default"*. Disponible en: https://stackoverflow.com/questions/71022803/the-requested-module-does-not-provide-an-export-named-default-error-but.

[14] StackOverflow. *Como crear un contenedor rectangular alrededor de los elementos de una activity de Android Studio*. Disponible en: https://stackoverflow.com/questions/40582555/how-to-draw-a-rectangle-box-around-elements-of-android-activity.

[15] StackOverflow. *Como crear un rectángulo con fondo transparente en Android Studio*. Disponible en: https://stackoverflow.com/questions/73084654/how-to-draw-rectangle-with-transparent-black-and-white-background-in-android-can.

[16] StackOverflow. *Como cambiar la imagen de un ImageView al pulsar un botón en Android Studio*. Disponible en: https://stackoverflow.com/questions/24755849/android-imagebutton-how-to-change-the-image-when-button-is-clicked.

[17] StackOverflow. *Como arreglar el error Çall requires API level 26 current min is 25.*^{en} *Android Studio*. Disponible en: https://stackoverflow.com/questions/56695997/how-to-fix-call-requires-api-level-26-current-min-is-25-error-in-android.

[18] StackOverflow. *Botón de Android Studio aparece en el editor gráfico pero no en el dispositivo en ejecución*. Disponible en: https://stackoverflow.com/questions/24116403/android-button-shows-in-graphical-layout-but-not-on-device.

[19] StackOverflow. *Simple Android Alert Dialog*. Disponible en: https://stackoverflow.com/questions/26097513/android-simple-alert-dialog.

[20] MongoDB. *Como empezar con el Driver de Java - Java Sync Driver - MondoDB Docs*. Disponible en: https://www.mongodb.com/docs/drivers/java/sync/current/get-started/#quick-start.

[21] MongoDB. *Java y MongoDB*. Disponible en: https://www.mongodb.com/resources/languages/java.

[22] MongoDB. *Uso de android studio y mongo DB realm - Cómo pushear un update a un Array en mi objeto*. Disponible en: https://www.mongodb.com/community/forums/t/using-android-studio-and-mongo-db-realm-how-to-push-an-update-to-an-array-in-my-o201919.

[23] Sentry Answers. *Redondear un decimal a N posiciones en Java*. Disponible en: https://sentry.io/answers/round-a-number-to-n-decimal-places-in-java/.

[24] Future Stud. *Retrofit2 como actualizar objetos en el servidor put vs patch*. Disponible en: https://futurestud.io/tutorials/retrofit-2-how-to-update-objects-on-the-server-put-vs-patch.

[25] GitHub. *Ejemplo de onClickListener*. Disponible en: https://gist.github.com/TheItachiUchiha/66322bc3a998bae23e56.

[26] Youtube. *SearchView con RecyclerView en Android Studio*. Disponible en: https://www.youtube.com/watch?v=tQ7V7iBg5zE.

[27] OpenAI. ChatGPT *Ayuda con retrofit en Android Studio* `https://chatgpt.com`

[28] OpenAI. ChatGPT *Consulta sobre el reciclado de clases de Retrofit y clases de Java de Android Studio a JavaFX* `https://chatgpt.com`

[29] OpenAI. ChatGPT *Ayuda con el paso de datos de una activity a otra en Android Studio* `https://chatgpt.com`

[30] OpenAI. ChatGPT *Ayuda con la documentación en latex* `https://chatgpt.com`

[31] OpenAI. ChatGPT *Como emplear Log.d para registrar logs en logcat* `https://chatgpt.com`