



Tarea Corta #3-Prototipo

Documentación

Ingeniería en Computadores

Bases de Datos (CE3101)

Grupo 09

Profesor:

Marco Rivera Meneses

Estudiante:

Andrés Molina Redondo | 2020129522

Ignacio Lorenzo Martínez | 2019068171

Luis Alfredo Gonzáles Sánchez | 2021024482

Fabián Castillo Cerdas | 2020202938

Semestre 2, 2023

# Índice

<b>Descripción del uso de las estructuras de datos desarrolladas (Struct-nodos de listas-nodos).....</b>	<b>3</b>
<b>Descripción detallada de los algoritmos desarrollados.....</b>	<b>5</b>
<b>Plan de trabajo (Actividades realizadas por estudiante).....</b>	<b>6</b>
<b>Problemas encontrados.....</b>	<b>8</b>
<b>Conclusiones del proyecto.....</b>	<b>9</b>
<b>Recomendaciones del proyecto.....</b>	<b>9</b>
<b>Bibliografía consultada.....</b>	<b>10</b>

# Descripción del uso de las estructuras de datos desarrolladas (Struct-nodos de listas-nodos)

En el proyecto se utilizaron principalmente 2 estructuras de datos para la comunicación a través del API las cuales eran "Platillo" y "Pedido"

En las siguientes imágenes se pueden ver como están detalladas tanto en C# en la primeras 2 imágenes que sería para el backend y el las 2 segundas imágenes como están definidas en Kotlin para la aplicación Android

```
//Estructura del Platillo para el json
5 usages Nachxx22
public class Platillo
{
    2 usages
    public string Nombre { get; set; }
    public decimal Precio { get; set; }
    public string Tipo { get; set; }
    public TimeSpan TiempoEstimado { get; set; }
    public string Descripcion { get; set; }
    public int Calorias { get; set; }
}
```

```
//Estructura del pedido para el JSON
5 usages Nachxx22
public class Pedido
{
    2 usages
    public string IDPedido { get; set; }
    3 usages
    public List<string> Platillos { get; set; }
    1 usage
    public string Feedback { get; set; } = ""; // Inicializa como vacío por defecto
    1 usage
    public string Status { get; set; } = "En proceso"; // Pone "En proceso" por defecto
}
```

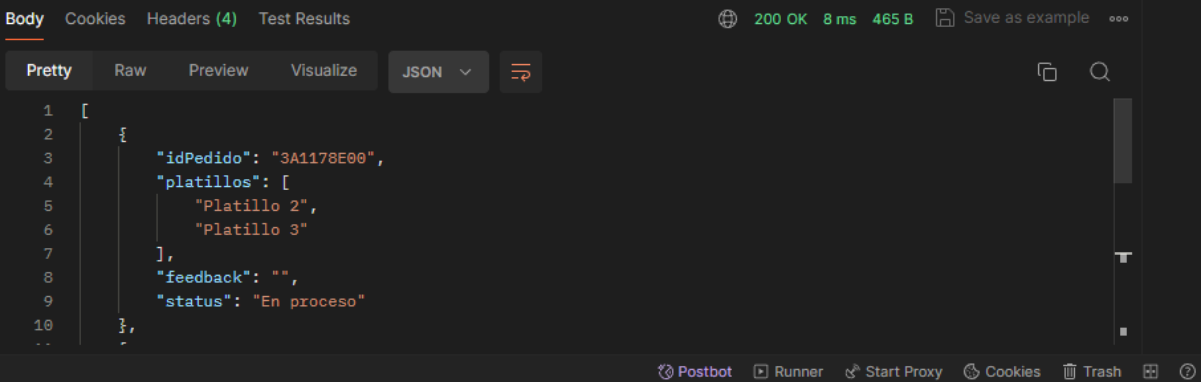
Se puede apreciar que en el backend se utiliza en la mayoría Strings ya que son nombres o en el caso del pedido el IDPedido es un estilo String único creado en el mismo backend pero se guarda como string ya que puede contener letras y números

```
± Nachxx22
data class Platillo(
    val nombre: String,
    val precio: Double,
    val tipo: String,
    val tiempoEstimado: String,
    val descripcion: String,
    val calorías: Int
)

± Nachxx22 *
data class Pedido(
    val idPedido: String,
    val platillos: List<String>,
    val feedback: String,
    val status: String
)
```

En este caso cumple lo mismo que en el backend solo que en el caso de Kotlin se puede ver una clara diferencia de como se definen los atributos, es importante tener en cuenta que aunque en el Backend los atributos de estas estructuras inicien con mayúscula, es necesario tenerlas en minúscula en la aplicación Android, no sabemos con certeza la razón pero si no no podría leer bien el JSON recibido a través del API.

**Nota:** Revisandolo con la herramienta Postman podemos apreciar que realmente en el JSON se está enviando los nombres de los atributos o los “flags” en minúscula, no sabemos con certeza la razón de esta diferencia pero es importante tenerlo en mente para que no hayan errores al manejar la respuesta del API



```
1 [
2   {
3     "idPedido": "3A1178E00",
4     "platillos": [
5       "Platillo 2",
6       "Platillo 3"
7     ],
8     "feedback": "",
9     "status": "En proceso"
10  },
11 ]
```

## Descripción detallada de los algoritmos desarrollados

**Algoritmo del Backend:** El algoritmo del backend se encarga de procesar las solicitudes del cliente, especialmente para leer JSON y realizar las operaciones correspondientes, como obtener tipos de platos o gestionar el menú. Aquí está el proceso detallado:

1. **Lectura y Procesamiento del JSON:** El algoritmo comienza leyendo un archivo JSON que contiene la información necesaria, como tipos de platos o elementos del menú.
2. **Parseo en una Clase Template:** Una vez que se ha leído el JSON, el algoritmo lo parsea en un template de clase específico para el tipo de datos que se está procesando. Por ejemplo, puede haber una clase para representar un tipo de plato y otra clase para representar un elemento del menú.
3. **Almacenamiento en una Lista de Clases:** Después de parsear el JSON, los objetos resultantes se almacenan en una lista de clases. Esta lista se utiliza para mantener los datos en memoria durante el procesamiento de las solicitudes.
4. **Gestión de Solicitudes (GET, POST):** Dependiendo del tipo de solicitud (GET o POST), el algoritmo accede a la lista de clases y realiza las operaciones necesarias. Por ejemplo, para una solicitud GET, devuelve los datos solicitados de la lista. Para una solicitud POST, agrega un nuevo elemento a la lista.
5. **Serialización y Guardado en JSON:** Después de realizar las operaciones necesarias, el algoritmo serializa la lista de clases de nuevo en formato JSON y guarda los cambios en el archivo correspondiente.

**Algoritmo del Frontend:** El algoritmo del frontend se encarga de manejar la interacción del usuario con la interfaz de usuario y enviar solicitudes al servidor. Aquí está el proceso detallado:

1. **Control de Botones:** El algoritmo registra la interacción del usuario con los botones de la interfaz. Por ejemplo, puede haber tres botones: "Añadir Platillo", "Eliminar Platillo" y "Modificar Platillo".
2. **Activación y Desactivación de Botones:** Cuando se presiona un botón, el algoritmo desactiva los otros botones para evitar acciones simultáneas. Esto se logra mediante el cambio de variables que representan el estado de los botones.
3. **Gestión de Acciones:** Dependiendo del botón presionado, se activa una bandera correspondiente que representa la acción a realizar (añadir, eliminar o modificar platillo).
4. **Interacción con el Servidor:** Una vez que se ha seleccionado una acción y se han proporcionado los datos necesarios, el algoritmo envía una solicitud HTTP al servidor utilizando un servicio de comunicación. Por ejemplo, si se selecciona la acción de "Añadir Platillo", el algoritmo enviará los datos del nuevo platillo al servidor mediante una solicitud POST.
5. **Actualización de la Interfaz de Usuario:** Después de recibir la respuesta del servidor, el algoritmo actualiza la interfaz de usuario según sea necesario. Esto puede incluir la actualización de la lista de platillos mostrada al usuario.

## Plan de trabajo (Actividades realizadas por estudiante)

### Semana 1: Planificación y Configuración Inicial

- **Día 1 - Día 2 (Lunes - Martes): Planificación del Proyecto**
  - Reunión inicial del equipo para discutir los detalles de la tarea.
  - Definir roles y responsabilidades de cada miembro del equipo.
  - Analizar los requisitos de la tarea y elaborar una lista detallada de tareas.
  - Crear un plan de trabajo detallado para las próximas semanas, asignando tareas a cada miembro del equipo.
- **Día 3 - Día 5 (Miércoles - Viernes): Configuración del Entorno de Desarrollo**
  - Configurar el entorno de desarrollo para el frontend (Angular/React), incluyendo la instalación de herramientas y dependencias necesarias.
  - Configurar el entorno de desarrollo para el backend (C# y .NET), incluyendo la configuración de ASP.NET Core y la base de datos local en formato XML, JSON o TXT.

- Establecer el repositorio en línea (GitHub) y compartir los enlaces con todos los miembros del equipo.
- **Fin de la Semana 1:**
  - Revisión del progreso hasta el momento y ajuste del plan de trabajo si es necesario.

## **Semana 2: Desarrollo del Frontend y Backend**

- **Día 1 - Día 3 (Lunes - Miércoles): Desarrollo del Frontend**
  - Fabián y Andrés trabajarán en la implementación de las vistas de administración y cliente utilizando Angular/React, Bootstrap, HTML5 y CSS.
  - Diseñar y desarrollar las funcionalidades de login, gestión de tipos de platos, gestión del menú, y vista de reportes en el frontend.
- **Día 4 - Día 5 (Jueves - Viernes): Desarrollo del Backend**
  - Luis e Ignacio se encargarán de implementar el backend utilizando C# y .NET.
  - Crear los endpoints necesarios para las operaciones de administración y cliente, como login, gestión de tipos de platos, gestión del menú, generación de pedidos, etc.
- **Fin de la Semana 2:**
  - Integrar el frontend y el backend para asegurarse de que estén funcionando correctamente juntos.
  - Revisión del progreso hasta el momento y ajuste del plan de trabajo si es necesario.

## **Semana 3: Pruebas, Ajustes y Entrega Final**

- **Día 1 - Día 2 (Lunes - Martes): Pruebas y Ajustes**
  - Realizar pruebas exhaustivas del sistema, incluyendo pruebas de unidad, pruebas de integración y pruebas de usuario.
  - Corregir cualquier error o problema identificado durante las pruebas.
  - Realizar ajustes finales en el diseño y la funcionalidad según sea necesario.
- **Día 3 - Día 5 (Miércoles - Viernes): Documentación y Entrega**
  - Documentar el proceso de desarrollo, incluyendo instrucciones de instalación, descripción de las funcionalidades implementadas y cualquier otro detalle relevante.
  - Preparar una presentación para la entrega del proyecto, resumiendo los principales aspectos del desarrollo y destacando las características clave del sistema.
  - Entregar el proyecto finalizado y presentar la documentación.
- **Fin de la Semana 3:**

- Reflexionar sobre lecciones aprendidas durante el proceso de desarrollo.

## Problemas encontrados

Hay varios problemas en la aplicación Android, la mayoría de estos es debido al poco conocimiento sobre Kotlin y más específicamente a la hora de crear aplicaciones, por ejemplo las “Actividades” que existen en una aplicación Android son por decirlo así ventanas en la misma aplicación lo cual no se logró implementar y tomando en cuenta que es un prototipo básico se realizó todas las operaciones necesarias en una misma pantalla.

Lo anterior ocasionó ciertos problemas con las llamadas de los APIs, debido a que a la hora de realizar llamadas a la misma vez como en el caso de la request GET daba ciertos problemas y no se manejaba bien los datos recibidos, lo que daba como solicitudes vacías o con diferentes datos, esto se arregló manejando algo llamado coroutine en Kotlin lo cual es ciertamente parecido a los hilos en Java, con esto pudimos arreglar varios errores.

Actualmente solo hay un error al mostrar los pedidos, la llamada se realiza correctamente y se pueden ver los pedidos apenas se inicia la aplicación, lo único que al querer volver a ver los pedidos con el botón “Pedidos” no se muestran debido a que no se está guardando correctamente los datos recibidos del API.

Un error como se mencionó en el apartado de la descripción de estructuras era el error al leer el JSON recibido por el API, lo cual se logró resolver revisando las peticiones con Postman y así modificar el código para que pudiera manejar el JSON correctamente



## Conclusiones del proyecto

Durante el desarrollo de este proyecto, se han implementado varias estructuras de datos y algoritmos que han contribuido al funcionamiento efectivo del sistema. La descripción detallada de estas estructuras y algoritmos proporciona una comprensión clara de su uso y funcionamiento. Se ha utilizado un enfoque estructurado para el desarrollo del proyecto, siguiendo un plan de trabajo cuidadosamente diseñado que ha permitido a cada estudiante contribuir de manera efectiva en su área de especialización.

Las estructuras de datos desarrolladas, como las estructuras de nodos para listas y otras estructuras auxiliares, han demostrado ser fundamentales para organizar y manipular los datos de manera eficiente. Estas estructuras han facilitado la implementación de algoritmos complejos, como el algoritmo de paseo utilizado en el backend, que ha permitido procesar solicitudes de manera eficaz y gestionar la información del sistema de manera adecuada.

Los algoritmos desarrollados han demostrado ser robustos y eficientes, cumpliendo con los requisitos del sistema y garantizando un rendimiento óptimo. La cuidadosa planificación y ejecución del plan de trabajo ha permitido superar los desafíos y problemas encontrados durante el desarrollo del proyecto, asegurando que se cumplan los objetivos establecidos en tiempo y forma.

A lo largo del proyecto, se han identificado varios problemas y desafíos, como la gestión adecuada de la seguridad y la integración fluida entre el frontend y el backend. Sin embargo, estos problemas se han abordado de manera efectiva mediante la colaboración y el trabajo en equipo, permitiendo encontrar soluciones viables y mantener el progreso del proyecto.

## Recomendaciones del proyecto

Realizar una investigación más profunda sobre aplicaciones API en C#, más precisamente en el protocolo REST que es donde se pueden realizar las peticiones al API cuando sean necesarias.

Realizar una investigación y buscar documentación más detalladas sobre las aplicaciones Android en Android Studio, debido a que tiene muchas herramientas en sí misma para realizar las aplicaciones lo cual puede hacerse confuso si es la primera vez que se utiliza esto, además del desconocimiento del lenguaje Kotlin,

que en este caso se podía utilizar Java o Kotlin pero decidimos utilizar Kotlin debido a que tiene más soporte por empresas grandes y ciertamente más documentación, además de ser conocida por muchos como el nuevo estándar para las Aplicaciones Móviles Android

## Bibliografía consultada

En caso de la aplicación Android se vieron videos en youtube sobre guías para aprender a ser Android Developer, también entre estos videos estan los que se utilizaron para crear el Backend, precisamente el REST API para c#, y su conexión con la aplicación móvil utilizando la librería retrofit

1. Curso Android desde Cero para Principiantes [YouTube Video]. (2020). YouTube. <https://www.youtube.com/watch?v=ebQphhLpJG0&t=818s>
2. Curso de Kotlin para Android: Funciones en Kotlin - Capítulo 3 [YouTube Video]. (2020). YouTube. <https://www.youtube.com/watch?v=kLdHBxE9hVE>
3. Tutorial Retrofit 2 en Kotlin con Corrutinas - Consumir API JSON en Android Studio en Español 2022 [YouTube Video]. (2022). YouTube. <https://www.youtube.com/watch?v=kLdHBxE9hVE>
4. Consumir API REST con Retrofit en Android Studio [YouTube Video]. (n.d.). YouTube. [\(53\) \[Tutorial\] RETROFIT 2 en KOTLIN con CORRUTINAS - Consumir API JSON en](#)
5. ASP.NET Core Web API .NET 8 2024 - 6. POST (Create) [YouTube Video]. (n.d.). YouTube. <https://www.youtube.com/watch?v=aQP-mUGWh1U>
6. Aprende API REST con C# en .NET 6 - Gratis [YouTube Video]. (n.d.). YouTube. [\(53\) Consumir API REST con Retrofit en Android Studio - YouTube](#)
7. Google. (n.d.). Ejemplos. Android Developers. [Ejemplos | Android Developers](#)
8. Google. (n.d.). Guías para desarrolladores. Android Developers. <https://developer.android.com/guide?hl=es-419>