Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Base de Datos

Grupo 1

Tarea 3

Mauricio Luna Acuña – 2022035682

Mauro Andrés Navarro Obando - 2018170753

Luis Felipe Jiménez Ulate - 2022211166

Andrés Guzmán Rojas

Daniel Duarte Cordero - 2022012866

Prof. Marco Rivera Meneses

I semestre 2024

Descripción de los métodos implementados:

API:

Admin:

ObtenerPedidosActivos(): Función que retorna los objetos de pedidos que tengan como estado “true”.

mostrarTopPlatos(): Función que retorna los 10 objetos de platos con mayor cantidad de ventas.

mostrarTopGanancias(): Función que retorna los 10 objetos de platos con mayor ganancia, esta calculada por el precio del plato multiplicado por su cantidad de ventas.

mostrarTopClientes(): Función que retorna los 10 objetos de clientes con mayor cantidad de pedidos realizados.

mostrarTopCalificacion(): Función que retorna los 10 objetos de platos con mayor cantidad de estrellas recibidas.

Chef:

ObtenerPedidosActivosChef(string correo): Función que recibe el correo de un chef y retorna los objetos de pedidos cuyo atributo “chef” tenga el indicado.

ObtenerPedidosDesasignado(): Función que retorna los objetos de pedidos cuyo atributo “chef” sea null.

agarrarPedido(int id, string correo): Función que recibe el id de un pedido y el correo de un chef, busca el objeto de pedido que tenga ese id y lo asigna al correo del chef indicado.

mostrarTodosPedidos(): Función que muestran todos los objetos de pedidos.

terminarPedido(int id): Función que recibe el id de un pedido y pasa su atributo “estado” de “true” a “false”.

ObtenerPedidosActivosOtrosChef(string correo): Función que recibe el correo de un chef y retorna los objetos de pedidos activos que estás asignados a otros chefs.

Cliente:

encontrarPorCedula(int cedula): Función que recibe una cedula retorna el objeto de un cliente asignado a esa cedula.

encontrarCorreoPasswd(string correo, string password): Función que recibe un correo y contraseña y retorna el objeto de un cliente asignado a ambos atributos.

agregarCliente([FromBody] ClienteRequest nuevoClienteRequest): Función que recibe un objeto de tipo ClienteRequest y crear un nuevo objeto Cliente para la base de datos.

modificarContraseña(int cedula, string nuevaContraseña): Función que recibe una cedula y una nueva contraseña, busca al cliente con esa cedula y modifica la contraseña con la nueva.

eliminarCliente(int cedula): Función que recibe una cedula y borra el cliente con esa cedula de la base de datos.

Usuario:

encontrarCorreoPasswd(string correo, string password): Función que recibe un correo y contraseña e indica si el usuario es chef o administrador

Aplicación Web:

login(): Esta es la función principal del componente Login. Como su nombre lo indica este componente es el encargado de determinar la condición del usuario que ingresa a la aplicación, ya sea administrador o chef.

GetActiveOrders() [admin]: Esta es una de las funciones principales del componente Menu. Esta toma platillos seleccionados previamente por el administrador. Una versión similar de esta misma vista será mostrada a los clientes.

NewDishT(): Esta es una de las funciones principales del componente Admin. Esta función da al administrador la potestad de crear nuevos platillos que se podrán mostrar en el menú a los clientes.

GetActiveOrders() [chef]: Esta es una de las funciones principales del componente chef, que es la vista principal del chef. Esta mostrará los platos en los que el chef está trabajando, así como el tiempo restante para completar el pedido.

GetPendingOrder(): Este es de las funciones principales del componente queue, esta es la vista en la que los chefs pueden tomar ordenes que estén esperando a ser atendidas, al tomar la orden esta función llama a la función selectOrder().

StartTimer(): Este es de la función principal del componente Timer. Este componente es parate de cada una de las ordenes una vez son presentadas al chef. Esta función iniciará una cuenta atrás en los pedidos para referencia del chef.

App Móvil

goBack()

moveTo()

Este tipo de funciones son usadas para cambiar entre las diferentes páginas de la app. Esto se logra haciendo que el router inyectado navegue según paths agregrados al app.routes.ts. goBack() se usa para devolverse a la página anterior mientras que las de tipo moveTo() a cualquier otra ruta.

Funciones para peticiones a la API

register():Registra usuarios

checkInfo():Verifica datos para iniciar sesión

goPedido(): Agrega pedidos

deleteAccount(): Elimina cuentas

calificar(): Realiza la calificación de los pedidos

(Algunos casos del ngOnInit)

Todas estas funciones se encargan de realizar pedidos mediante el Http Client. Se emplearon peticiones de tipo put, post, get y delete. El formato típico para la petición es el siguiente _http.peticion('parametros').subscribe()

En el caso de funciones con get bastaba con un único parámetro, siendo este el url de la petición, para luego obtener un objeto con diferentes atributos. El URl es conformado por la ip de la API, la entidad a la que se quiere acceder, así como los parámetros y sus respectivos argumentos.

De la misma manera el delete únicamente requiere de un parámetro el cual es la URL de la petición. En esta petición no retorna nada.

Finalmente, tanto para el put como para el post se establecen dos parámetros. Una Url, conformada por la ip de la API, la entidad con la que se desea trabajar y el nombre de la función en la API, el segundo parámetro es para recibir un objeto. Ninguna de estas dos peticiones retorna nada.

Descripción de las estructuras de datos desarrolladas:

API:

En la implementación de la API se requirió un gran uso de objetos y clases para representar a sus respectivas entidades y atributos de la base de datos, como por ejemplo:

Cliente

Plato

Pedido

Chef

Administrador

También se requirió el uso de de arrays tanto de tipo "string" como de tipo "int" debido a que se definió algunos atributos como atributos multi-evaluados como en los siguientes casos:

Cliente: List<int> teléfonos, List<int> pedidos

Factura: List<int> pedidos

Menu: List<int> platos

Pedido: List<int> platos

Plato: List<string> ingredientes

Aplicación Web:

Para la aplicación web no fue necesario utilizar estructuras de datos complejas, realmente la mayor parte de la información se manejó con simples listas, ya que también esto era lo más sencillo sobre tobo para acoplar los archivos de HTML con TypeScript.

Con el uso de listas, se pudo realizar la implementación del menú que presenta sus opciones como una lista, las ordenes que pudieran tomar los chefs se manejan como una lista y de igual manera se manejan las ordenes una vez que han sido asignadas a un chef.

App Móvil
En el caso de la app móvil se requirió únicamente el uso de arrays.Tanto de tipo number e incluso para objetos de interfaces como IDish y IClient .El arreglo de objetos fue necesario ya que se almacenaba en una página mediante servicios, y luego se hacía uso de ellos en otra página. Para los arreglos de tipo number se usaron para almacenar los id de los platos en la página de carritos y también para los números de teléfono en el sing-up.

Descripción detallada de los algoritmos desarrollos:

API:

No se utilizaron algoritmos para la implementación de la API ya que para su búsqueda en el archivo .json de la base de datos se utilizó la librería Newtonsoft.Json la cual ya tiene métodos de serialización y deserialización de archivos .json.

Aplicación Web:

Para el desarrollo de la app web, no se desarrolló ningún algoritmo realmente complejo. La creación de los timers fue lo único que requirió un poco más de lógica ya que estos son un sub-componente del componente “ordenes” y cuando el timer llega a cero, la orden queda inactiva.

Problemas conocidos:

API:

En la implementación de la API no quedaron problemas encontrados sin resolver.

Aplicación Web:

La aplicación web presentó el problema de que no se logró incorporar la vista en la que un chef puede tomar pedidos de otros chefs. Esto por una situación de tiempo, ya que la implementación tiene requisitos similares a la ventana para tomar ordenes disponibles, pero un sistema de confirmación en el que un chef confirma a otro que puede tomar una de sus órdenes es una característica que no se pudo implementar

App Móvil:

El problema para la aplicación móvil surgió a la hora de generar la apk para correr el código en dispositivos móviles. Ya que si bien, esta fue posible de generar, no se logró establecer la conexión con la API a diferencia de cuando se corría en el navegador. Investigando se pudo encontrar que para solucionar este problema era necesario utilizar servicios en la nube o bien acceder a una IP pública, por lo que claramente no era posible realizar la conexión desde una apk trabajando la API en localhost.

Problemas encontrados:

API:

Cliente:

Descripción: A la hora de crear un cliente se necesita recibir un parámetro de una lista de enteros de teléfonos, pero este parámetro no aparecía en la url con el endpoint de dicha función, por lo que no podía recibir dicho parámetro desde el frontend y por ende no puede crear el usuario.

Intentos de solución: Se intentó cambiar la forma la cual la app web envía la lista de los teléfonos.

Se intentó cambiar el atributo de teléfonos por una lista de strings.

Se cambió la forma la cual la función recibe parámetros a recibir un objeto, pero este tenía que recibir todos los atributos incluso los que no son necesarios para crear dicho cliente por lo que el frontend no podía enviar todos los atributos.

Solución: Se cambió la forma la cual la función de agregar cliente recibe los parámetros a recibir un objeto y aparte se creó otra clase basada en cliente, pero solamente con los atributos necesarios para crearlo.

Recomendaciones: Al trabajar con una API Rest Service, es mejor que las funciones http request tengan como parámetro un body en vez de parámetros separados, ya que no permiten algunos valores como listas.

Conclusiones: Se logró crear un función para añadir un cliente a la base de datos al recibir un objeto en forma de body.

Aplicación Web:

1) **Interconectar componentes en Angular:** Este fue uno de los principales problemas que se tuvo con la aplicación web ya que se tenía conocimiento de desarrollo web básico (html, css, javaScript) pero hubo que acoplarse a la herramienta, Angular en este caso. Poder resolver esto era clave para las etapas posteriores, ya que para trabajar eficientemente se necesitaba comprender al menos las funcionalidades básicas de Angular.
- **Intentos de solución sin éxito:** Realmente acá no se mencionaría tanto algo como casos sin éxito ya que fue principalmente un proceso necesario de aprendizaje.
- **Soluciones encontradas: Conforme se fueron creando e incorporando nuevos componentes se fue mejorando la interacción con el framework.**
- **Recomendaciones:** Para un siguiente caso, una de las principales recomendaciones es la de buscar cuales son los conceptos y características más básicas de la nueva tecnología, con base en eso se puede ir profundizando en los conceptos y ellos mismos son una guía sobre lo próximo por aprender.
- **Conclusiones:** El aprendizaje de Angular fue retador al inicio, pero hay mucha información al respecto de la tecnología y por medio de la práctica se logró tener un manejo aceptable de la misma.

- **Bibliografía:** Fireship. (2018, June 20). Angular Components Beginner's Guide [Video]. YouTube. https://www.youtube.com/watch?v=23o0evRtrFI

2) **Timer :** Este fue uno de los principales problemas que se tuvo con la aplicación web ya que se tenía conocimiento de desarrollo web básico (html, css, javaScript) pero hubo que acoplarse a la herramienta, Angular en este caso. Poder resolver esto era clave para las etapas posteriores, ya que para trabajar eficientemente se necesitaba comprender al menos las funcionalidades básicas de Angular.

- **Intentos de solución sin éxito:** Realmente acá no se mencionaría tanto algo como casos sin éxito ya que fue principalmente un proceso necesario de aprendizaje.
- **Soluciones encontradas: Conforme se fueron creando e incorporando nuevos componentes se fue mejorando la interacción con el framework.**
- **Recomendaciones:** Para un siguiente caso, una de las principales recomendaciones es la de buscar cuales son los conceptos y características más básicas de la nueva tecnología, con base en eso se puede ir profundizando en los conceptos y ellos mismos son una guía sobre lo próximo por aprender.
- **Conclusiones:** El aprendizaje de Angular fue retador al inicio, pero hay mucha información al respecto de la tecnología y por medio de la práctica se logró tener un manejo aceptable de la misma.
- **Bibliografía:** Fireship. (2018, June 20). Angular Components Beginner's Guide [Video]. YouTube. https://www.youtube.com/watch?v=23o0evRtrFI

Aplicación Movil:

1) **Parser datos AppMovil-API:** Al realizar el parser de datos entre la aplicación móvil y la API se obtenían variables indefinidas a la hora de acceder a los atributos.

- **Intentos de solución sin éxito:** No se cuenta con una cantidad exacta de intentos, ya que fueron bastantes, pero no tomaban de mucho tiempo, un aproximado puede ser unos 20 intentos. En donde se realizaron console.log() para encontrar el punto en el código donde se producía in indefinición.
- **Soluciones encontradas:** Se comunicó la sección del equipo encargada de la API para poder ver qué problema podría estarse presentando, estos mencionaron que había un error de sintaxis en una palabra, al corregir esto ya se logró solucionar este error.
- **Recomendaciones:** Tener más cuidado a la hora de manejar distintas variables se tenga un estándar para que no sucedan casos como estos
- **Conclusiones:** El trabajo en equipo consta de la comunicación y de decisiones grupales para tener un código entendible y que funcione.

Evidencias:

Plan de trabajo		
Actividad Planeada	**Responsable**	**Fecha de entrega**
Generación de proyecto y enrutamiento	Daniel Duarte	5/03
Funciones de datos de admin	Mauricio Luna	11/03
Funciones de datos de chef	Mauricio Luna	12/03
Páginas autorización	Daniel Duarte	7/03
Páginas carrito y home	Daniel Duarte	12/03
Funciones de datos de cliente	Mauricio Luna	15/03
Estética a la aplicación, y funciones varias	Felipe Jiménez	18/03
Funciones de pedidos	Mauricio Luna	21/03
Funciones para conexión API	Daniel Duarte	21/03
Funciones de reportes	Mauricio Luna	22/03
Documentación externa	Daniel Duarte	22/03

Minutas de sesión		
Fecha	**Integrantes**	**Discusión**
2/3	Mauricio Luna Felipe Jiménez Mauro Navarro Andrés Guzmán Daniel Duarte	Se discute sobre los roles de cada integrante, así como una fecha estimada de futuras reuniones
7/3	Mauricio Luna Felipe Jiménez Mauro Navarro	Se reunió el equipo para determinar con qué programas se trabajaría el proyecto.
15/3	Mauricio Luna Felipe Jiménez Mauro Navarro Andrés Guzmán Daniel Duarte	Se reunió el equipo para revisar los avances, determinar cambios y definir futuras funciones
17/3	Mauricio Luna Andrés Guzmán	Reunión para explicar cómo se conecta la API con la app web
19/3	Mauricio Luna Daniel Duarte	Reunión para explicar cómo se conecta la API con la app móvil
22/03	Mauricio Luna Felipe Jiménez Mauro Navarro Andrés Guzmán	Reunión para juntar las partes de cada integrante y revisar su funcionalidad

	Daniel Duarte	

Bitácora Mauricio Luna	
Fecha	**Tarea**
3/03	Se crea las clases de los modelos en la API de las entidades. Se trabaja con las funciones de autenticación del administrador.
5/03	Se termina la función de autenticación del administrador y se comienza con las del chef
7/03	Se comienza con las funciones de creación de platos con el retorno de sus datos
8/03	Se realiza funciones de gestión de pedidos en conjunto con los platos
11/03	Se terminan las funciones de admin para la gestión de platos
12/03	Se termina las funciones de chef para la gestión de pedidos y se comienza con las funciones del cliente
14/03	Se crean funciones de agregar, eliminar, modificar usuario.
15/03	Se actualizan dichas funciones de manera que la app móvil y web les sea más conveniente
21/03	Se finaliza las funciones de pedidos que van conectadas a otras entidades como cliente y chef
22/03	Se finaliza las funciones de generación de reportes para el admin.

Bitácora Felipe Jiménez	
Fecha	**Tarea**
7/03	Se logra la instalación de angular e ionic para lograr modificar la aplicación móvil
15/03	Se consigue modificar las primeras ventanas de la aplicación móvil para que se vea amigable al usuario
16/03	Se arreglan errores que se presentaron el día anterior
20/03	Se realiza la función para enviar pedido a la API
21/03	Se crea la página de factura con las funciones adecuadas
22/03	Se realiza la página de feedback y se termina la aplicación

Bitácora Daniel Duarte

1 Activitidades

1.1 1 de Marzo

Reunión distribución tareas
Tiempo estimado: 1 hora

1.2 4 de Marzo

Creación de proyecto, páginas iniciales
Tiempo estimado: 1 hora

1.3 12 de Marzo

Creación de páginas autenticación
Tiempo estimado: 2h

1.4 20 de Marzo

Conexión API Tiempo estimado: 7 horas

1.5 22 de Marzo

Últimos cambios y documentación
Tiempo estimado: 5 horas

Evidencia de algunas reuniones por un servidor de Discord.