FOODTRACKER

Integrantes:
Paloma Osiris Báez Lara
Alejandro Sánchez Marín
Juan Reyes María Teresa

12 DE ENERO DE 2025

Contenido

Introducción

Esta documentación ofrece una explicación minuciosa de los usos para el sitio web de este proyecto, una plataforma de entrega de alimentos por internet que facilita a los usuarios la petición de comida desde varios establecimientos y obtenerla en su lugar de origen. La página web se ha diseñado para proporcionar una experiencia sencilla e intuitiva, facilitando al usuario acciones como crear y cambiar su cuenta, explorar menús, hacer pedidos, realizar pagos y revisar el estado de su carrito de compra.

Cada caso de uso especifica los procedimientos requeridos para finalizar cada acción, incluyendo flujos normales, alternos y excepciones para afrontar posibles fluctuaciones en la interacción del usuario. Este documento funcionará como guía para el equipo de desarrollo y para cualquier individuo que participe en la creación.

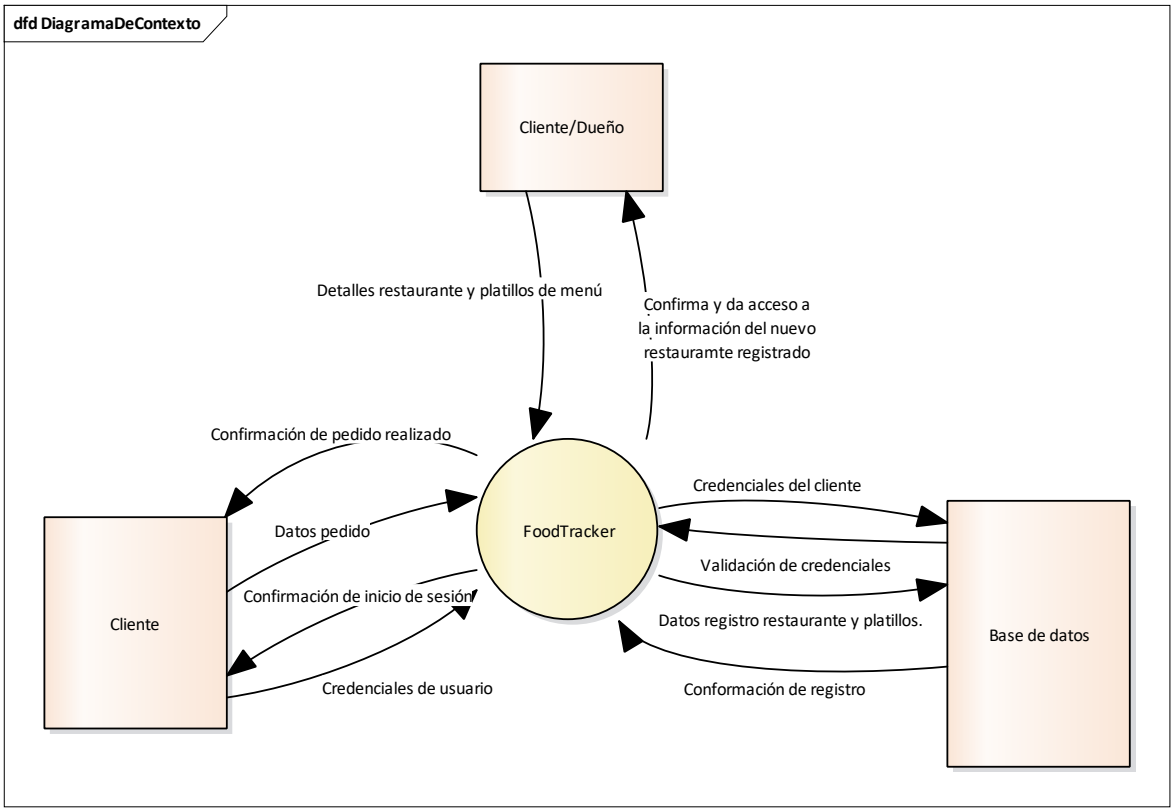
Requerimientos

Contexto

FoodTracker está diseñado para facilitar la interacción entre clientes y dueños de restaurantes en el ámbito de la entrega de alimentos a domicilio. Surge como respuesta a la creciente demanda de servicios de comida rápida y personalizada, adaptándose a un mercado donde la conveniencia y la eficiencia son esenciales.

Por un lado, los clientes desean explorar opciones gastronómicas, realizar pedidos desde la comodidad de su hogar y evaluar la calidad del servicio recibido. Por otro lado, los dueños de restaurantes requieren herramientas que les permitan administrar su negocio de manera integral, incluyendo la gestión de menús, pedidos y la percepción de su marca a través de calificaciones.

El proyecto FoodTracker conecta a ambos actores en un entorno digital intuitivo, seguro y accesible, promoviendo una experiencia positiva tanto para quienes consumen como para quienes ofrecen los servicios.



Clases de usuario

A continuación, en la Tabla 1, se presenta una descripcion detallada de las diferentes clases de usuario que interactúan con el sistema FoodTracker. Esta clasificación es importante para entender los niveles de acceso que cada tipo de usuario posee dentro del sistema.

Tabla 1. Clases de usuario

Clase de usuario	Tipo de usuario	Descripción
Cliente normal	Favorecido	El cliente es un tipo de usuario favorecido en el sistema de pedidos de comida a domicilio, diseñado para brindar una experiencia completa y personalizada. Este usuario puede crear una cuenta, modificarla según sus necesidades, y cambiar su contraseña para mantener la seguridad de su perfil. Además, tiene la posibilidad de consultar los platillos disponibles, realizar pedidos para recibirlos en su domicilio, y calificar el restaurante con un sistema de rating. Si decide convertirse en dueño, el cliente puede registrar un restaurante, lo que le habilita funcionalidades adicionales dentro del sistema.

Dueño	Favorecido	El dueño es un tipo de usuario favorecido que combina las funcionalidades del cliente con herramientas avanzadas para gestionar restaurantes. Puede realizar todas las acciones básicas, como crear y modificar su cuenta, cambiar su contraseña, consultar platillos, realizar pedidos, calificar restaurantes, y registrar nuevos restaurantes. Adicionalmente, el dueño tiene acceso exclusivo a gestionar los restaurantes que registre. Esto incluye la capacidad de modificar la información del restaurante, registrar menús, editar detalles de los platillos existentes, y eliminar platillos que ya no desee incluir en el menú. Estas funcionalidades avanzadas permiten al dueño optimizar la administración de su negocio directamente desde el sistema

Casos de uso

A nivel de análisis se lograron identificar los siguientes casos de uso, los cuales se dividieron por actor:

Cliente/Dueño

1.- Crear cuenta

2.- Modificar cuenta

3.- Realizar pedido

4.-Cambiar contraseña

5.- Consultar platillos

6.- Calificar restaurante (Rating)

7.- Registrar restaurante

Dueño

8.- Modificar restaurante

9.- Registrar menú

10.- Editar menú

11.- Eliminar platillo de menú

Prototipos de UI

Los prototipos realizados se muestran a continuación:

REGISTRARSE

1 2

Nombre completo:

Telefono::

Registrarme

Modificar cuenta

Nombre completo:

Correo electrónico:

Telefono:

Cambiar contraseña

Modificar

FOOD TRACKER

Home Restaurantes Paloma Báez ▼

Cambiar contraseña

Correo electrónico:

Continuar

Token:

Continuar

Nueva contraseña:

Guardar

FOOD TRACKER

Home Restaurantes Paloma Báez ▼

RESTAURANTES

Categorias

Pizzas Hamburguesas Mariscos Mexicana

HOJAS VERDES

Ver más

CREPAS DE PAULETTE

Ver más

DIVERGENTE

Ver más

FOOD TRACKER

Home Restaurantes Paloma Báez ▼

HOJAS VERDES

Mariscos

Telefono	Rating	Estatus
227731198	4.5/5.0	Abierto

MENÚ

Ensalada rusa
$80.00
+ Agregar

Ensalada rusa
$80.00
+ Agregar

Ensalada rusa
$80.00
+ Agregar

Ensalada rusa
$80.00
+ Agregar

FOOD TRACKER

Home Restaurantes Paloma Báez ▼

Estatus
Abierto

Mariscos

Descripción

Ensalada rusa

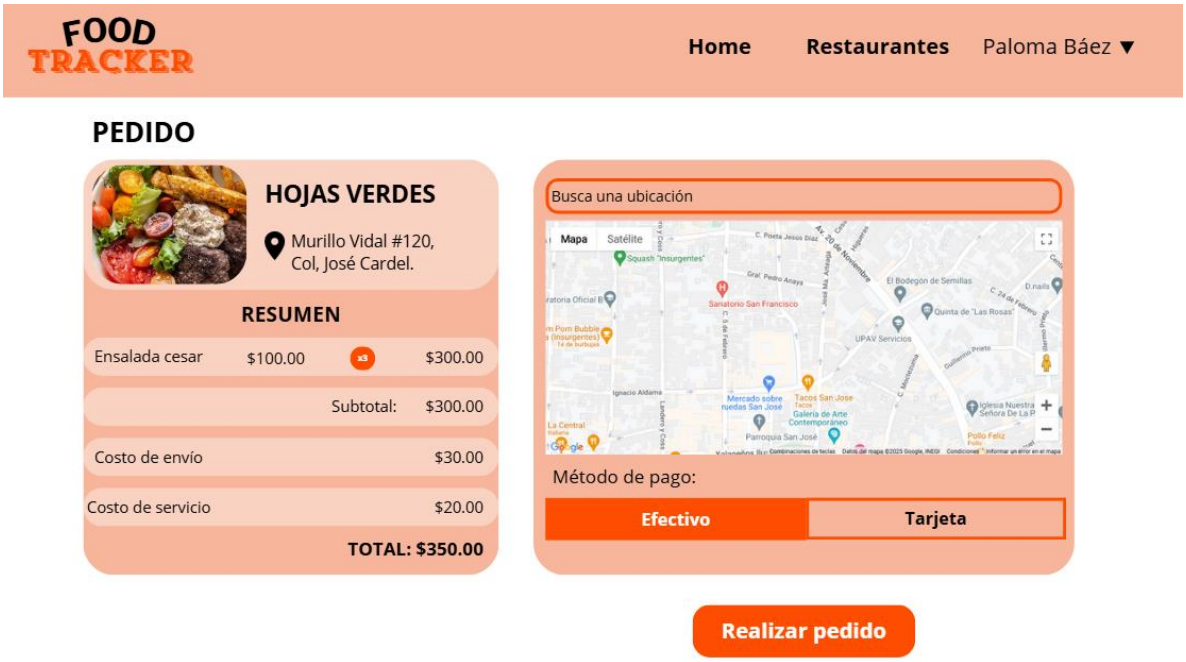
Esta ensalada de papas y verduras con un aderezo de mayonesa es muy conocida en Rusia, por toda Europa y también en América del Sur. Es un excelente acompañamiento para las fiestas, una cena con amigos o para una comida al aire libre.

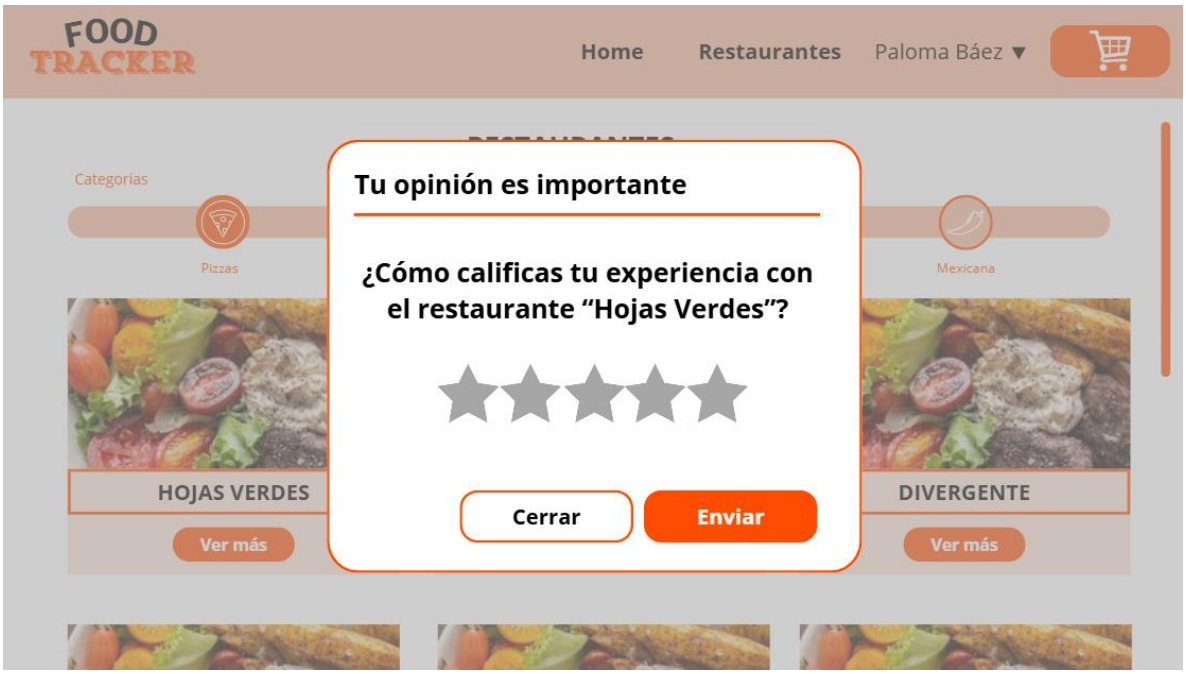
Cerrar

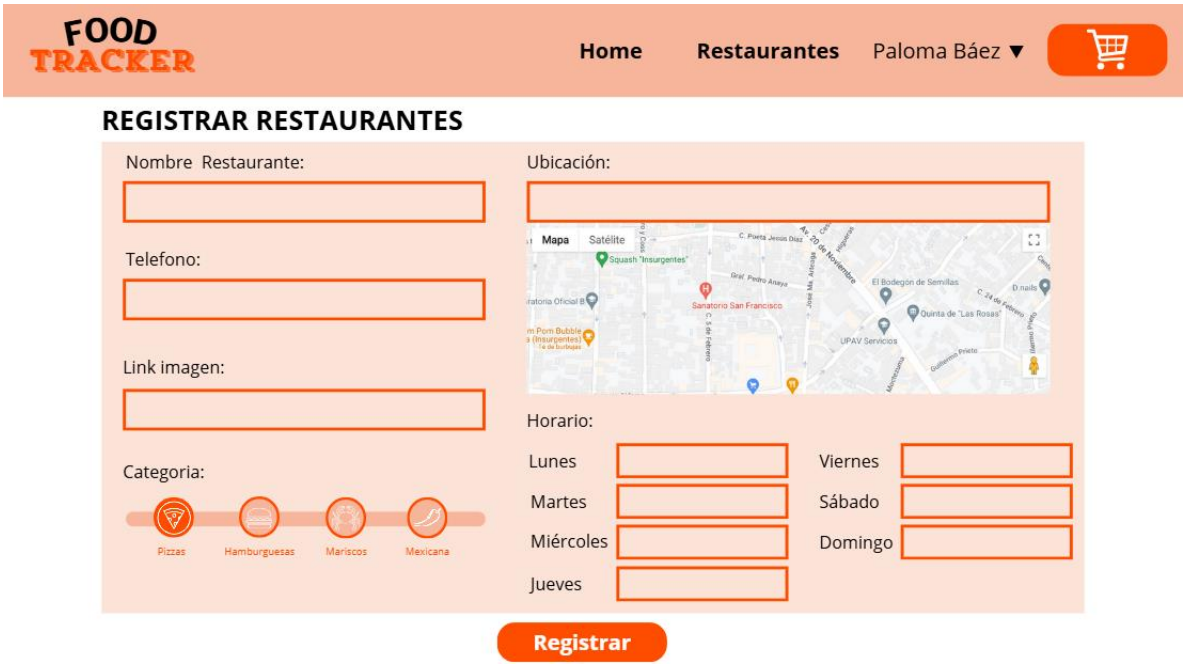
Ensalada rusa
$80.00
+ Agregar

Ensalada rusa
$80.00
+ Agregar


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
3
Carrito
Ensalada rusa
$80.00
- 1 +
Ensalada cesar
$100.00
- 3 +
Continuar
Estatus
Abierto
Mariscos
Ensalada rusa
$80.00
+ Agregar
+ Agregar
+ Agregar
Ensalada rusa
$80.00
+ Agregar


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
PEDIDO
HOJAS VERDES
Murillo Vidal #120,
Col, José Cardel.
RESUMEN
Ensalada cesar $100.00 x3 $300.00
Subtotal: $300.00
Costo de envío $30.00
Costo de servicio $20.00
TOTAL: $350.00
Busca una ubicación
Mapa
Satélite
Método de pago:
Efectivo
Tarjeta
Realizar pedido


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
Categorias
Pizzas
Mexicana
Tu opinión es importante
¿Cómo calificas tu experiencia con el restaurante "Hojas Verdes"?
Cerrar
Enviar
HOJAS VERDES
DIVERGENTE
Ver más
Ver más


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
REGISTRAR RESTAURANTES
Nombre Restaurante:
Telefono:
Link imagen:
Categoria:
Pizzas
Hamburguesas
Mariscos
Mexicana
Ubicación:
Mapa
Satélite
Horario:
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
Registrar

FOOD TRACKER

Home Restaurantes Paloma Báez ▼

MODIFICAR RESTAURANTE

Nombre Restaurante:

La Picrecha

Telefono:

2281204371

Link imagen:

jhttps://hips.hearstapps.com/hmg-rod/imajdk

Categoria:

Pizzas Hamburguesas Mariscos Mexicana

Ubicación:

Xalapeños Ilustres 201, Col. Centro 91100

Horario:

Lunes	2:00 - 19:00	Viernes	12:00 - 20:00
Martes	12:00 - 20:00	Sábado	12:00 - 20:00
Miércoles	12:00 - 20:00	Domingo	12:00 - 20:00
Jueves	12:00 - 20:00		

Modificar

FOOD TRACKER

Home Restaurantes Paloma Báez ▼

REGISTRAR MENÚ

Platillo:

Precio:

Link imagen:

Descripción:

Añadir

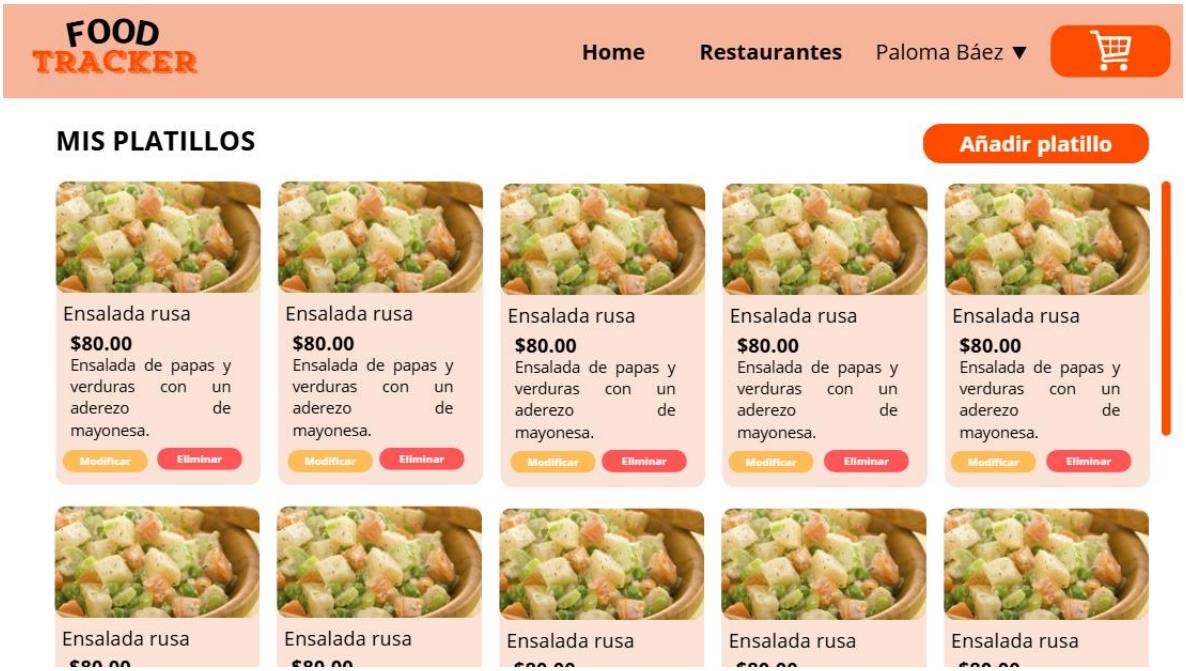
Platillos

Ensalada rusa
$80.00
Ensalada de papas y verduras con un aderezo de mayonesa.
Eliminar

Ensalada rusa
$80.00
Ensalada de papas y verduras con un aderezo de mayonesa.
Eliminar

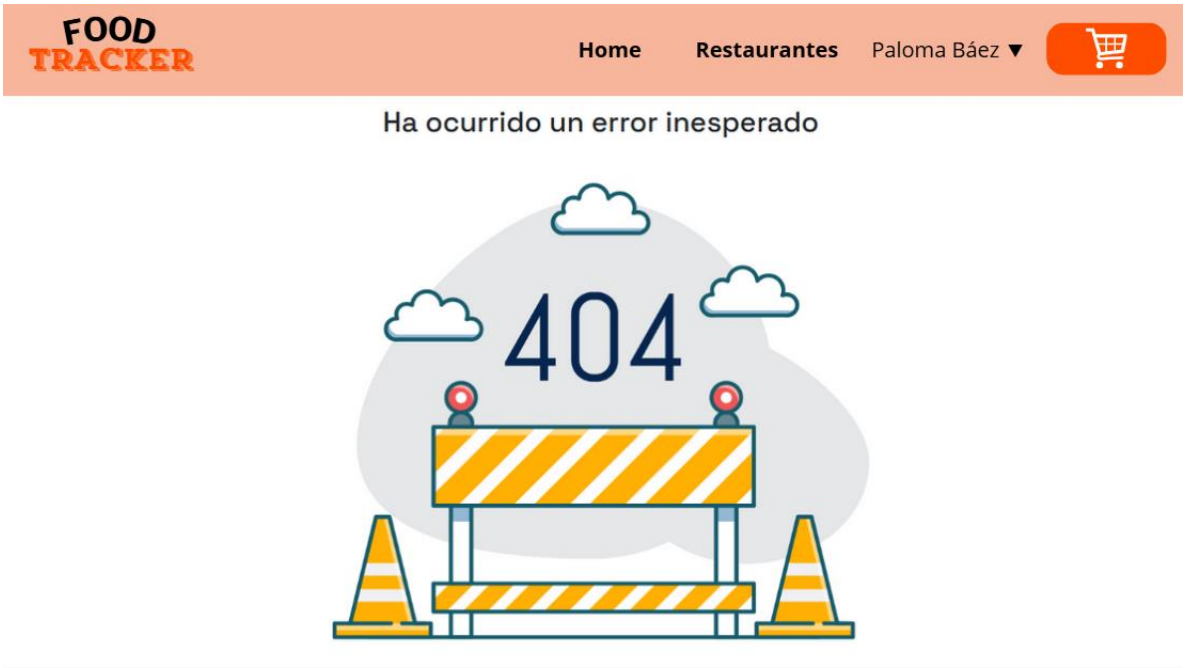
Ensalada rusa
$80.00
Ensalada de papas y verduras con un aderezo de mayonesa.
Eliminar

Registrar platillos


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
MIS PLATILLOS
Añadir platillo
Ensalada rusa
$80.00
Ensalada de papas y verduras con un aderezo de mayonesa.
Modificar
Eliminar


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
MIS PLATILLOS
Añadir platillo
Añadir platillo
Platillo:
Precio:
Link imagen:
Descripción:
Añadir
Ensalada rusa
$80.00
Ensalada de papas y verduras con un aderezo de mayonesa.
Modificar
Eliminar


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
MIS PLATILLOS
Añadir platillo
Modificar platillo
Platillo:
Ensalada rusa
Precio:
80.00
Link imagen:
jhttps://hips.hearstapps.com/hmg-rod/imajdk
Descripción:
Ensalada de papas y verduras con un aderezo de mayonesa.
Modificar
Ensalada rusa
$80.00
Ensalada de papas y verduras con un aderezo de mayonesa.
Modificar
Eliminar


FOOD TRACKER
Home
Restaurantes
Paloma Báez ▼
Ha ocurrido un error inesperado
404

Requerimientos funcionales

Estos requisitos detallan las funciones específicas que el sistema debe realizar para satisfacer las necesidades de los dos tipos de usuarios. A continuación, se presenta una lista de los requisitos funcionales clave para el sistema FoodTracker, los cuales abarcan desde la gestión de una cuenta hasta la manipulación de menús y platillos.

RF-01. El sistema debe permitir a los usuarios crear una cuenta proporcionando su nombre, correo electrónico y contraseña.

RF-02. El sistema debe permitir a los usuarios modificar los datos de su cuenta, como nombre, correo electrónico y contraseña.

RF-03. El sistema debe permitir a los usuarios cambiar su contraseña en cualquier momento.

RF-04. El sistema debe permitir a los clientes realizar pedidos de los platillos disponibles en los restaurantes registrados.

RF-05. El sistema debe permitir a los usuarios consultar la lista de platillos disponibles en los restaurantes registrados.

RF-06. El sistema debe permitir a los usuarios calificar los restaurantes mediante un sistema de rating.

RF-07. El sistema debe proporcionar un mapa que permita a los usuarios registrar ubicaciones de manera efectiva.

RF-08. El sistema debe permitir a los usuarios registrar un restaurante proporcionando información básica (nombre, dirección, horario de atención, etc.).

RF-09. El sistema debe permitir a los dueños modificar la información de sus restaurantes registrados.

RF-10. El sistema debe permitir a los dueños registrar un menú para sus restaurantes.

RF-11. El sistema debe permitir a los dueños editar los detalles de los platillos en el menú, como nombre, descripción, precio e imagen.

RF-12. El sistema debe permitir a los dueños eliminar platillos del menú según sea necesario.

Requerimientos no funcionales

Los requisitos no funcionales establecen los criterios de calidad y las condiciones en las que el sistema operará. La siguiente lista proporciona una visión clara de estos aspectos no funcionales que son esenciales para el éxito y la eficiencia del sistema FoodTracker.

Seguridad:

RNF – 01. El sistema debe implementar medidas de autenticación robustas, como la verificación en dos pasos, para garantizar que solo los usuarios autorizados puedan acceder a sus cuentas.

RNF – 02. Se deben establecer políticas de acceso y permisos adecuadas para garantizar que cada usuario solo pueda acceder y modificar datos de su rol correspondiente, evitando así el acceso no autorizado a los datos de otros usuarios.

Desempeño:

RNF – 03. Se deben implementar técnicas de optimización de rendimiento, como el almacenamiento en localStorage para mejorar la velocidad de carga de los archivos multimedia y la navegación por la interfaz del usuario.

RNF – 04. La arquitectura del sistema debe ser escalable, permitiendo aumentar la capacidad de procesamiento y almacenamiento de manera fácil y rápida en caso de un aumento repentino en la carga de trabajo. 15

Disponibilidad:

RNF – 05. El sistema debe tener una infraestructura redundante y tolerante a fallos para minimizar el tiempo de inactividad y garantizar la disponibilidad continua del servicio.

RNF – 06. El sistema debe utilizar técnicas de optimización de bases de datos, como la indexación adecuada y la optimización de consultas, para garantizar tiempos de respuesta rápidos en las operaciones de búsqueda y recuperación de datos.

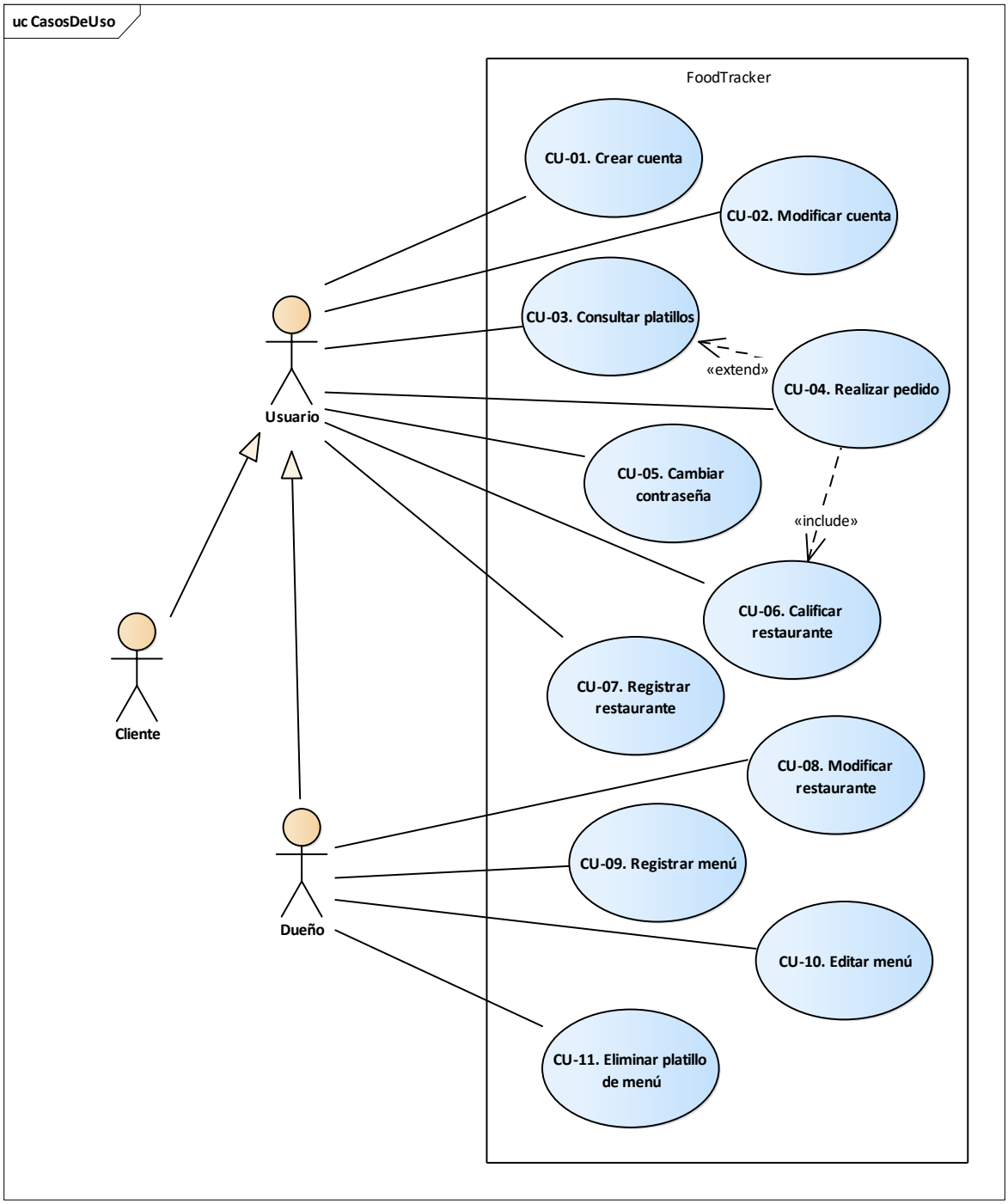
Diseño

Diseño Arquitectónico

Vista de casos de uso

El diagrama de casos de uso muestra las interacciones entre dos tipos de actores (Cliente y Dueño) y el sistema FoodTracker. Los clientes y dueños pueden crear y editar

cuentas, consultar platillos para realizar pedidos y asi calificar restaurantes. Si un cliente decide registrar un restaurante, pasa a ser dueño, donde podrá, modificar el mismo, registrar, editar y eliminar platillos del menú de su restaurante. Las relaciones de extensión indican la acción que se deriva de la principal, asi como las relaciones de inclusión, indican la acción que conlleva la principal.

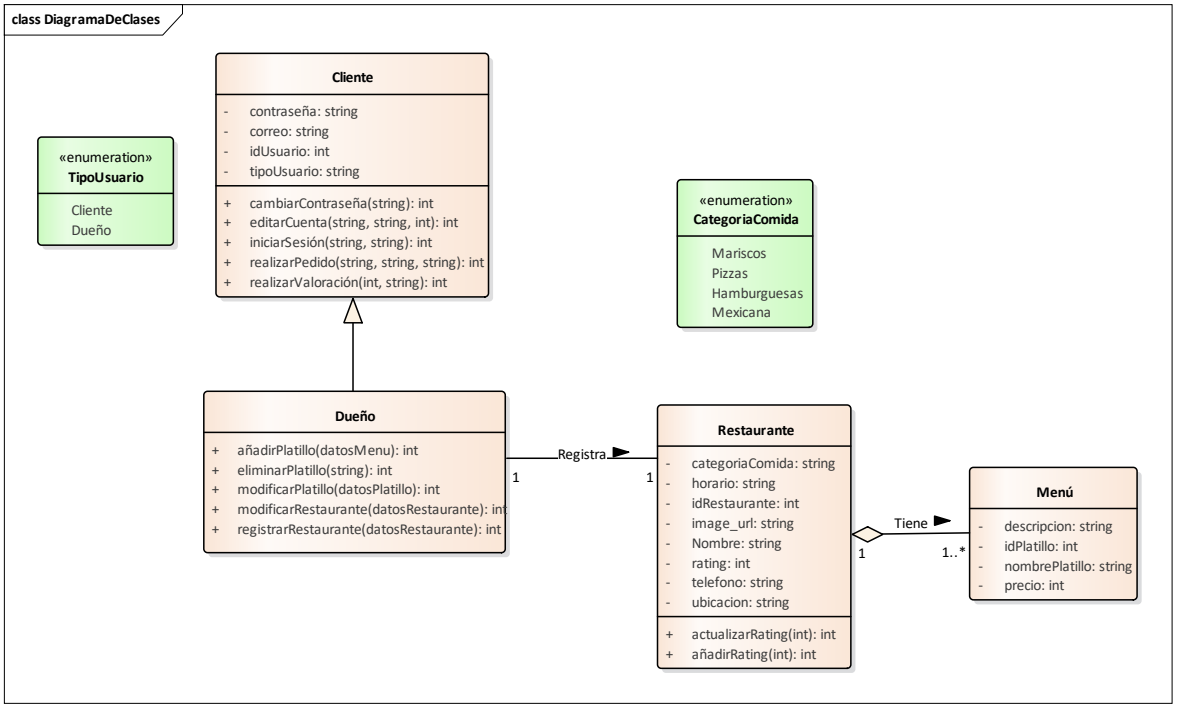


Vista lógica

El diagrama de clases de FoodTracker muestra la estructura y relaciones entre sus principales componentes. La clase principal, Cliente, incluye atributos como contraseña, correo, idUsuario y tipoUsuario, y métodos que permiten cambiar contraseñas, editar cuentas, iniciar sesión, realizar pedidos y registrar valoraciones. Por otro lado, la clase Dueño, que hereda de Cliente, añade funcionalidades específicas para administrar restaurantes, como añadir, eliminar y modificar platillos, además de registrar y actualizar información de los restaurantes.

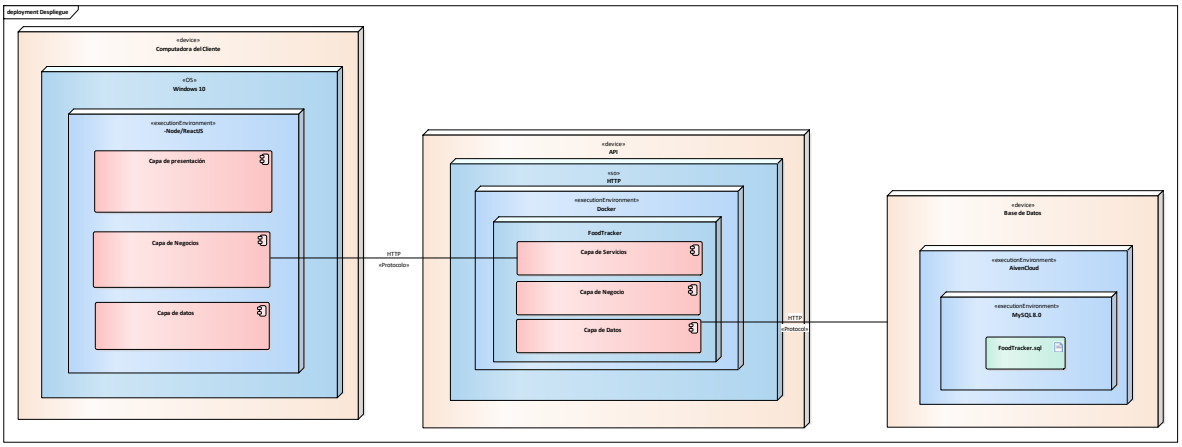
La clase Restaurante incluye atributos como categoriaComida, horario, idRestaurante, image_url, nombre, rating, teléfono y ubicación. También cuenta con métodos que permiten gestionar las valoraciones y actualizaciones de la calificación del restaurante. Cada restaurante tiene una relación de composición con la clase Menú, que describe los platillos disponibles mediante atributos como descripcion, idPlatillo, nombrePlatillo y precio. La relación entre un restaurante y sus platillos es de uno a muchos.

Además, el sistema utiliza enumeraciones para estandarizar datos clave: TipoUsuario, que define los roles de Cliente y CategoriaComida, que enumera las opciones de comida disponibles. También se refleja una relación de herencia entre las clases Cliente y Dueño, mientras que las relaciones de asociación y composición conectan a los dueños con los restaurantes y a los restaurantes con sus menús.



Vista de despliegue

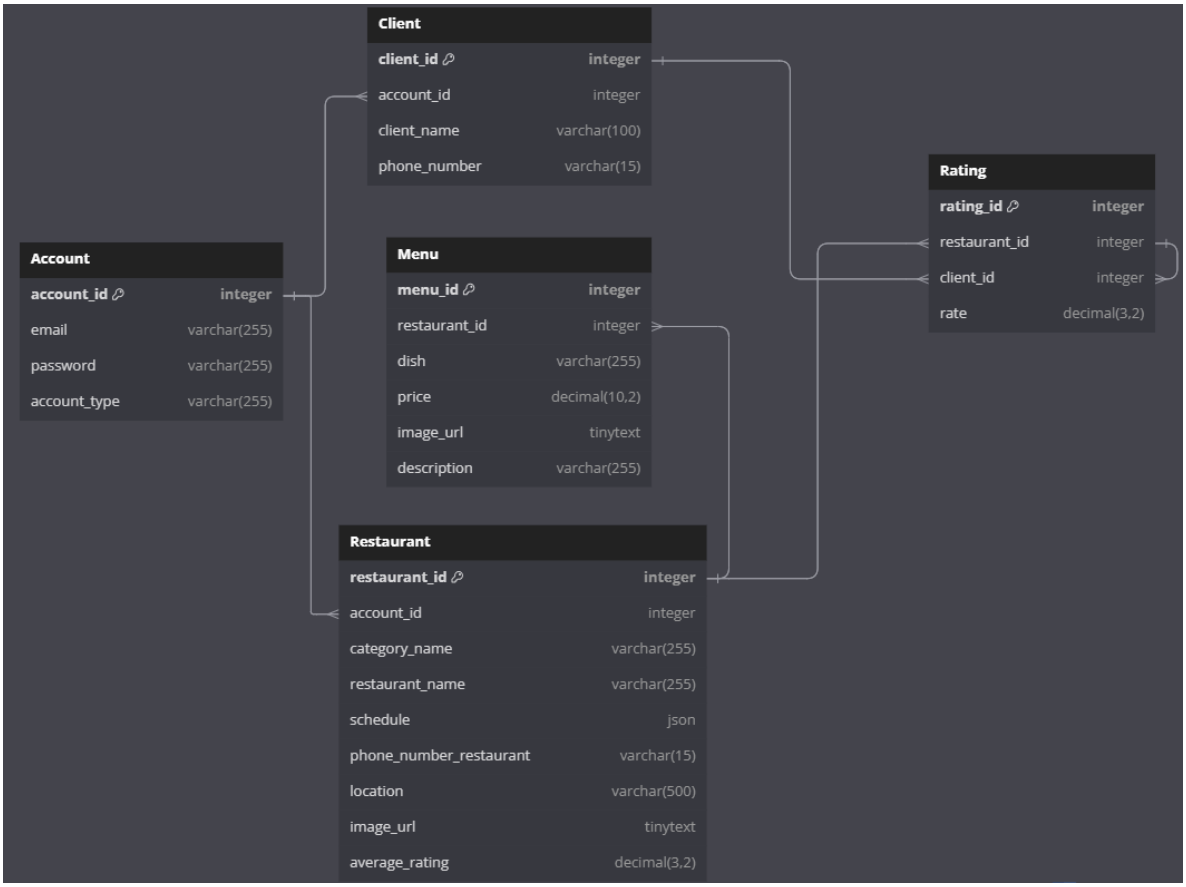
La arquitectura de la aplicación cliente-servidor se estructura en varios componentes distribuidos en entornos y tecnologías diferentes. En la computadora del cliente con sistema operativo Windows 10 y entorno de ejecución Node usando ReactJS, se encuentran las capas de Presentación, Negocios y Datos, responsables de la interfaz de usuario, la lógica de negocio y el acceso a los datos respectivamente, comunicándose con el servidor a través de HTTP. El servidor API, alojado en Docker, utiliza un microservicio implementado en Java Spring Boot, abarcando capas de Servicios, Negocios y Datos. La base de datos MySQL 8.0, que se encuentra en un hosting, almacena datos para el microservicio y se comunica mediante HTTP. Las interconexiones entre componentes se realizan principalmente mediante HTTP.



Modelo de datos

Este modelo representa el sistema de almacenamiento y gestión de restaurantes en FoodTracker. La tabla Account es la central en el modelo y almacena información de las cuentas de los usuarios, como el email, la contraseña y el tipo de cuenta. Esta tabla está vinculada a la tabla Client, que representa a los clientes y contiene información adicional como el nombre y número de teléfono del cliente, y está asociada a la cuenta mediante la clave foránea account_id. La tabla Restaurant almacena información sobre los restaurantes, incluyendo nombre, categoría, horario, ubicación y promedio de calificación, y está asociada a la tabla Account mediante la clave foránea account_id, lo que permite la vinculación con los usuarios que administran los restaurantes. Además, la tabla Menú contiene los platos disponibles en los restaurantes, con detalles como el nombre del plato, precio y descripción, y se

relaciona con la tabla Restaurant mediante la clave foránea restaurant_id. Por último, la tabla Rating permite registrar las calificaciones que los clientes otorgan a los restaurantes, con un campo rate que se encuentra limitado entre 1 y 5, y se vincula tanto a la tabla Client como a la tabla Restaurant mediante claves foráneas, permitiendo un sistema de evaluación para los restaurantes.



Descripciones de casos de uso

ID:	CU-01
Nombre del CU:	Crear cuenta
Descripción:	El usuario ingresa algunos datos personales que se registrarán en la base de datos del Sistema.
Actor(es):	Cliente (Primario)
Disparador:	El Cliente da clic en “Registrar” de la ventana **“Home”** sobre el Header**.**
Precondiciones:	PRE-01: El **usuario** a registrar no se encuentra previamente registrado en el sistema.

Flujo Normal:	1. FoodTracker muestra la ventana **"CreateAccount"** con un formulario. 2. El CLIENTE ingresa su *nombre, correo, contraseña, teléfono* y selecciona la opción "Registrar" de **"CreateAccount"**. 3. FoodTracker valida los datos ingresados del **usuario** en el formulario. (ver FA-01, ver FA-02, ver EX-01). 4. FoodTracker cierra la ventana **"CreateAccount".** 5. FoodTracker guarda los datos del **usuario** en la base de datos, llevando al usuario al "**Login**" (ver EX-01). 6. Fin del caso de uso.
Flujos Alternos:	FA-01: Datos Inválidos: 1. FoodTracker detecta campos inválidos y muestra sobre el campo el error. 2. El CLIENTE corrige lo indicado en el error. 3. Regresar al paso 2 del flujo normal. FA-02: Correo Duplicado: 1. FoodTracker detecta información duplicada y muestra sobre el campo de correo el error. 2. El CLIENTE cambia el correo por otro. 3. Regresar al paso 2 del flujo normal.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Un nuevo **usuario** es registrado en la base de datos del sistema.
Incluye:	N/A
Extiende:	N/A

ID:	CU-02
Nombre del CU:	Modificar cuenta
Descripción:	El cliente puede actualizar sus datos personales.
Actor(es):	Cliente
Disparador:	El cliente selecciona "Modificar cuenta" en el menú de perfil.
Precondiciones:	PRE-01: El **usuario** debe tener una cuenta activa.

Flujo Normal:	1. FoodTracker muestra la ventana **"ModifyAccount"** con un formulario y muestra los datos del cliente sobre los campos del formulario. (ver EX-01) 2. El CLIENTE modifica su *nombre, correo, contraseña, teléfono* y selecciona la opción "Modificar" de **"ModifyAccount"**. 3. FoodTracker valida los datos ingresados del **usuario** en el formulario. (ver FA-01, ver FA-02, ver EX-01). 4. FoodTracker cierra la ventana **"ModifyAccount".** 5. FoodTracker modifica los datos del **usuario** en la base de datos, llevando al usuario al "**Home**" (ver EX-01). 6. Fin del caso de uso.
Flujos Alternos:	FA-01: Datos Inválidos: 1. FoodTracker detecta campos inválidos y muestra sobre el campo el error. 2. El CLIENTE corrige lo indicado en el error. 3. Regresar al paso 2 del flujo normal. FA-02: Correo Duplicado: 1. FoodTracker detecta información duplicada y muestra sobre el campo de correo el error. 2. El CLIENTE cambia el correo por otro. 3. Regresar al paso 2 del flujo normal.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Un **usuario** es modificado en la base de datos del sistema.
Incluye:	N/A
Extiende:	N/A

ID:	CU-03
Nombre del CU:	Consultar platillos
Descripción:	El **cliente** puede consultar los platillos que tenga registrado un **restaurante** en la base de datos.
Actor(es):	Cliente
Disparador:	El **cliente** selecciona un restaurante.
Precondiciones:	PRE-01: El **cliente** debe estar logueado en el sistema.

Flujo Normal:	1. FoodTracker muestra la ventana **"MenuRestaurant"** 2. FoodTracker muestra los detalles del **restaurante** y muestra los platillos que tenga registrado dicho **restaurante** mostrando detalles como su *nombre, descripción, imagen y precio* del **platillo**. (ver EX-01). 3. El CLIENTE selecciona el botón "Agregar" de un **platillo**. 4. FoodTracker agrega el **platillo** al carrito de compras del **cliente.** (ver FA-01) 5. Fin del caso de uso.
Flujos Alternos:	FA-01: Caso de uso 4, realizar pedido
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	El **cliente** puede o no realizar un pedido.
Incluye:	N/A
Extiende:	CU 04: Realizar pedido

ID:	CU-04
Nombre del CU:	Realizar pedido
Descripción:	El usuario puede realizar realizar pedidos de los platillos que haya seleccionad.
Actor(es):	Cliente
Disparador:	El usuario seleccionar el botón Continuar de su carrito
Precondiciones:	PRE-01: Debe existir platillos en el carrito del cliente
Flujo Normal:	1. FoodTracker muestra la ventana "**MakeOrder"** con un listado de los **platillos** y un mapa con un campo de texto. 2. FoodTracker calcula el subtotal de venta de la orden. 3. El CLIENTE ingresa en el campo de texto su ubicación para envió del pedido. 4. FoodTracker habilita un switch para seleccionar la forma de pago, efectivo o por tarjeta. (ver FA-01). 5. El CLIENTE selecciona una opción y selecciona el botón "Aceptar". 6. FoodTracker muestra la GUI-PEDIDO-REALIZADO. 7. El CLIENTE selecciona la opción "Ok" de la GUI-PEDIDO-REALIZADO. 8. FoodTracker cierra la GUI-PEDIDO-REALIZADO y el flujo continua en el caso de uso Calificar Restaurante. 9. Fin del caso de uso

Flujos Alternos:	FA-01: Pago con tarjeta: 1. FoodTracker muestra un modal para realizar el pago por tarjeta. 2. El CLIENTE ingresa los campos correspondientes a su información bancaria y presiona la opción “Pagar”. 3. FoodTracker cierra el modal para realizar el pago por tarjeta. 4. El flujo continua en el paso normal 5.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción “Ok” de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se realiza un pedido de un cliente.
Incluye:	CU-06: Calificar restaurante
Extiende:	N/A

ID:	CU- 05
Nombre del CU:	Cambiar contraseña
Descripción:	Se desea cambiar la contraseña de un cliente dada la posibilidad de que este haya olvidado su contraseña.
Actor(es):	Cliente (Primario)
Disparador:	El Cliente selecciona la opción "Olvide mi contraseña" de la ventana **"Login"**.
Precondiciones:	PRE-01: El **usuario** tiene una cuenta registrada en el sistema.
Flujo Normal:	1. FoodTracker muestra un formulario para cambiar la contraseña de un **usuario** en la ventana **"ChancePassword"**. 2. El CLIENTE ingresa el correo y selecciona la opción "Validar". (ver FA-04) 3. FoodTracker valida que los datos ingresados sean correctos. (ver FA-01, ver EX-01) 4. FoodTracker envía un correo con un token y habilita el campo para ingresar el token. 5. FoodTracker ingresa el token y selecciona la opción "Validar". (ver FA-04) 6. FoodTracker valida que el token ingresado sea correcto. (ver FA-02, ver EX-01) 7. FoodTracker habilita el campo para ingresar la nueva contraseña. 8. FoodTracker ingresa su nueva contraseña y selecciona la opción "Aceptar". (ver FA-04) 9. FoodTracker valida que la contraseña ingresada sea válida. (ver FA-03) 10. FoodTracker guarda la nueva contraseña del **usuario** en la base de datos y muestra la GUI-CAMBIO-EXITOSO. (ver EX-01). 11. FoodTracker cierra la ventana **"ChancePassword"** y regresa al **usuario** a la ventana **"Login"**. 12. Fin del caso de uso.
Flujos Alternos:	FA-01: Correo inválido 1. FoodTracker detecta que el email ingresado es inválido y muestra el error sobre el campo. 2. El CLIENTE cambia el campo ingresado por uno valido. 3. Regresa al flujo normal 2. FA-02: Token invalido 1. FoodTracker detecta que el token ingresado no es el correcto y muestra el error sobre el campo. 2. El CLIENTE ingresa un token valido. 3. Regresa al flujo normal 5. FA-03: Contraseña invalida

	1. FoodTracker detecta que la contraseña ingresada no cumple con el formato y lo muestra sobre el campo 2. El CLIENTE corrige lo ingresado por un campo valido. 3. Regresa al flujo normal 8. FA-04: Salir del registro 1. El CLIENTE selecciona el icono de regreso de la ventana **"ChangePassword"**. 2. FoodTracker cierra la ventana **"ChangePassword"** y abre la ventana **"Login"**.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se cambia la antigua contraseña que tenía el **usuario** asignado y se guarda en la base de datos la nueva contraseña
Incluye:	N/A
Extiende:	N/A

ID:	CU-06
Nombre del CU:	Calificar restaurante
Descripción:	Se califica la experiencia del usuario al realizar un pedido en un restaurante.
Actor(es):	Cliente
Disparador:	Se presiona el botón "Aceptar" de la ventana "**MakeOrder**":
Precondiciones:	PRE-01: El **cliente** ha realizado un pedido.
Flujo Normal:	1. FoodTracker muestra un modal preguntando al usuario sobre su experiencia y los sabores del platillo al realizar el pedido en el restaurante con un puntaje del 0 al 5. 2. El CLIENTE selecciona un valor del puntaje. (ver FA-01) 3. FoodTracker guarda en la base de datos el puntaje realizado y muestra la GUI-AGRADECIMIENTOS. (ver EX-01) 4. El CLIENTE selecciona la opción "Aceptar". 5. FoodTracker cierra la GUI-AGRADECIMIENTOS y el modal del puntaje. 6. Fin del caso de uso.
Flujos Alternos:	FA-01: Salir de la calificación del restaurante. 1. El CLIENTE selecciona la opción cerrar del modal del puntaje de restaurante. 2. FoodTracker cierra el modal. 3. Fin del caso de uso.
Excepciones:	EX-01: No hay conexión con el servidor

	1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se guarda en la base de datos la asociación del puntaje del **restaurante** realizado por el **cliente.**
Incluye:	N/A
Extiende:	N/A

ID:	CU-07
Nombre del CU:	Registrar restaurante
Descripción:	El cliente ingresa los datos necesarios para llevar a cabo el registro de un nuevo restaurante.
Actor(es):	Cliente (Primario)
Disparador:	El Cliente da clic en "Registrar restaurante" de la ventana **"Home"** sobre el Header**.**
Precondiciones:	PRE-01: El **restaurante** a registrar no se encuentra previamente registrado en el sistema.
Flujo Normal:	1. FoodTracker muestra la ventana **"RegisterRestaurant"** con un formulario. 2. El CLIENTE ingresa su *nombre*, *horario*, *categoría*, *ubicación* e *imagen* y selecciona la opción "Registrar" de **"RegisterRestaurant".** 3. FoodTracker valida los datos ingresados del **usuario** en el formulario. (ver FA-01, ver EX-01). 4. FoodTracker cierra la ventana **"RegisterRestaurant".** 5. FoodTracker guarda los datos del **restaurabte** en la base de datos, llevando al usuario al "**Home**" (ver EX-01). 6. Fin del caso de uso.
Flujos Alternos:	FA-01: Datos Inválidos: 1. FoodTracker detecta campos inválidos y muestra sobre el campo el error. 2. El CLIENTE corrige lo indicado en el error. 3. Regresar al paso 2 del flujo normal.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El CLIENTE selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se registra un nuevo **restaurante** asociándolo al **cliente** quien lo creo.

Incluye:	N/A
Extiende:	N/A

ID:	CU-08
Nombre del CU:	Modificar restaurante
Descripción:	El dueño ingresa los datos necesarios para llevar a cabo la modificación de los datos de su restaurante.
Actor(es):	Dueño (Primario)
Disparador:	El Dueño da clic en "Modificar restaurante" de la ventana **"Home"** sobre el Header**.**
Precondiciones:	PRE-01: Existe un **restaurante** relacionado a la cuenta del **usuario**.
Flujo Normal:	1. FoodTracker muestra la ventana **"ModifyRestaurant"** con un formulario y recupera los datos del restaurante sobre el formulario. (ver EX-01). 2. El DUEÑO modifica su *horario, ubicación* e *imagen* y selecciona la opción "Modificar" de **"ModifyRestaurant".** 3. FoodTracker valida los datos ingresados del **dueño** en el formulario. (ver FA-01, ver EX-01). 4. FoodTracker cierra la ventana **"ModifyRestaurant".** 5. FoodTracker modifica los datos del **restaurabte** en la base de datos, llevando al dueño al "**Home**" (ver EX-01). 6. Fin del caso de uso.
Flujos Alternos:	FA-01: Datos Inválidos: 1. FoodTracker detecta campos inválidos y muestra sobre el campo el error. 2. El DUEÑO corrige lo indicado en el error. 3. Regresar al paso 2 del flujo normal.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El DUEÑO selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se modifica los datos del **restaurante** asociado del **dueño**.
Incluye:	N/A
Extiende:	N/A

ID:	CU-09
Nombre del CU:	Registrar menú
Descripción:	El dueño ingresa los datos necesarios para registrar platillos en su restaurante
Actor(es):	Dueño (Primario)
Disparador:	El Dueño da clic en "Registrar Menú" de la ventana **"Home"** sobre el Header**.**
Precondiciones:	PRE-01: Existe un **restaurante** relacionado a la cuenta del **usuario**.
Flujo Normal:	1. FoodTracker muestra la ventana **"RegisterMenu"** con un formulario. 2. El DUEÑO ingresa el *nombrePlatillo, imagen, precio, descripciónPlatillo* y selecciona la opción "Registrar" de **"RegisterMenu".** 3. FoodTracker valida los datos ingresados del **dueño** en el formulario y registra en la base de datos el platillo. (ver FA-01, ver EX-01). 4. FoodTracker muestra la GUI-REGISTRO-MENU-EXITOSO. 5. El DUEÑO selecciona la opción "Aceptar" de GUI-REGISTRO-MENU-EXITOSO. 6. FoodTracker cierra la GUI-REGISTRO-MENU-EXITOSO. 7. Fin del caso de uso.
Flujos Alternos:	FA-01: Datos Inválidos: 1. FoodTracker detecta campos inválidos y muestra sobre el campo el error. 2. El DUEÑO corrige lo indicado en el error. 3. Regresar al paso 2 del flujo normal.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El DUEÑO selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se registran los datos del **platillo** asociado al **restaurante**.
Incluye:	N/A
Extiende:	N/A

ID:	CU-10
Nombre del CU:	Editar menú
Descripción:	El dueño modifica los datos previos del platillo en su restaurante
Actor(es):	Dueño (Primario)
Disparador:	El Dueño da clic en "Modificar Menú" de la ventana **"Menu"** sobre el card de un platillo**.**
Precondiciones:	PRE-01: Existe al menos un platillo registrado en el **restaurante** del dueño.

Flujo Normal:	1. FoodTracker muestra un modal y recupera la información del platillo seleccionado. (ver EX-01) 2. El DUEÑO modifica el *nombrePlatillo, imagen, precio, descripciónPlatillo* y selecciona la opción "Modificar" del modal. 3. FoodTracker valida los datos ingresados del **dueño** en el formulario y modifica en la base de datos el platillo. (ver FA-01, ver EX-01). 4. FoodTracker muestra la GUI-CAMBIO-MENU-EXITOSO. 5. El DUEÑO selecciona la opción "Aceptar" de GUI-CAMBIO-MENU-EXITOSO. 6. FoodTracker cierra la GUI-CAMBIO-MENU-EXITOSO y el modal. 7. Fin del caso de uso.
Flujos Alternos:	FA-01: Datos Inválidos: 1. FoodTracker detecta campos inválidos y muestra sobre el campo el error. 2. El DUEÑO corrige lo indicado en el error. 3. Regresar al paso 2 del flujo normal.
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR. 2. El DUEÑO selecciona la opción "Ok" de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se modifican los datos del **platillo** asociado al **restaurante**.
Incluye:	N/A
Extiende:	N/A

ID:	CU-11
Nombre del CU:	Eliminar menú
Descripción:	El dueño elimina los datos del platillo en su restaurante
Actor(es):	Dueño (Primario)
Disparador:	El Dueño ha consultado los platillos que tiene registrados en su restaurante.
Precondiciones:	PRE-01: Existe al menos un platillo registrado en el **restaurante** del dueño.
Flujo Normal:	1. FoodTracker muestra los platillos que tenga un **dueño** registrado en su **restaurante.** (ver EX-01). 2. El DUEÑO selecciona la opción "Eliminar" de un platillo. 3. FoodTracker eliminar de la base de datos el platillo muestra la GUI-BORRADO-MENU-EXITOSO. (ver EX-01) 4. El DUEÑO selecciona la opción "Aceptar" de GUI-BORRADO-MENU-EXITOSO. 5. FoodTracker cierra la GUI-BORRADO-MENU-EXITOSO. 6. Fin del caso de uso.
Flujos Alternos:	
Excepciones:	EX-01: No hay conexión con el servidor 1. FoodTracker muestra la ventana GUI-ERROR-SERVIDOR.

	2. El DUEÑO selecciona la opción “Ok” de la GUI-ERROR-SERVIDOR. 3. FoodTracker cierra la GUI-ERROR-SERVIDOR. 4. Fin del caso de uso.
Postcondiciones:	POST-01: Se eliminan los datos del **platillo** asociado al **restaurante**.
Incluye:	N/A
Extiende:	N/A

Construcción

Explicación de la solución

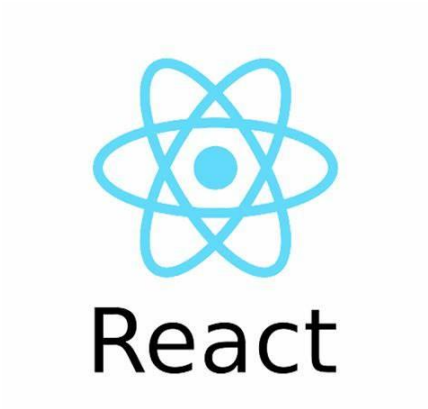
Debido a que se lograron identificar una gran cantidad de componentes y que podían ser reutilizables, se decidió hacer uso de la biblioteca de ReactJS. El uso de esta tecnología facilito la creación de los componentes y de igual manera su uso para el consumo de la API, para llevar a cabo esto se ocupó la librería Axios. ReactJS involucro muchas ventajas como la modularidad de los componentes haciendo que se lograran identificar errores de manera más rápida así como se mantuvo un bajo acoplamiento debido a esto mismo. Sin embargo, al momento de redistribuirlo haciendo uso de los contenedores con Podman, debido al desconocimiento de que ReacJS sus páginas son de una sola, es decir, Single Page Application. Se tuvieron errores con esto.

En cuanto al backend, se hizo uso del lenguaje de programación de Java apoyándose del framework de Spring Boot. El uso de este framework no represento una significativa dificultad ya que se estaba familiarizado con este lenguaje de programación y la creación de API Rest Full con otras tecnologías. Quizás lo que represento mayor dificultad fueron sus librerías como Spring Security debido a la autoconfiguración con la que cuenta este framework. De igual manera, se hizo uso de mecanismos para la autenticación y autorización con Json Web Token, esto también represento un reto ya que el configurarlo para este framework nos pareció muy distinto a como se hacía con otras tecnológicas y era necesario una mayor cantidad de pasos para configurarla. Aunque existieron dificultades, después de haber realizado las configuraciones resulto más sencillo el desarrollo debido a esta misma autoconfiguración, por ejemplo, como igual se incluyo el ORM de Hiberne y JPA, con un simple repositorio fue posible realizar la conexión a la base de datos sin que representara una gran cantidad de líneas de código; de igual manera, el mapeo de las entidades este ORM ayudo demasiado y para la creación de las consultas debido a esta autoconfiguración, el ORM lograba identificar las relaciones y el tipo de consulta que se requería únicamente con el

nombre del método dentro del repositorio, a partir de este nombre, el ORM construía la consulta lo que simplifico mucho el trabajo al no construir consultas personalizadas.

En cuanto al gestor de bases de datos, se decidió el uso de MySQL debido al conocimiento previo a esta tecnología y a la compatibilidad que existe con Java, la construcción de la base de datos no represento un reto y fue sencillo debido a que ya se tenían conocimientos previos de esto como se menciona, la base de datos se decidió almacenar en un servidor en la nube llamado Aiven Console, el almacenar la base de datos en lugares en la nube facilita mucho el desarrollo colaborativo al compartir la misma base de datos logrando hacer modificaciones sobre esta y que se reflejara para los desarrolladores de este trabajo.

Tecnologías para el desarrollo empleadas:



Repositorios de los proyectos

A continuación, se añaden los repositorios correspondientes para el cliente y servidor de FoodTracker.

Cliente web:

https://github.com/Alejandrin08/FoodtrackerClient.git

Servidor (API):

https://github.com/Alejandrin08/FoodTracker.git

Pruebas

Se utilizó Postman como herramienta para la validación funcional de los endpoints de la API. Las pruebas incluyeron la verificación de respuestas correctas para solicitudes válidas, para aquellas con errores en la solicitud, así como la validación de tokens para la autorización y autenticación. De igual manera, esta herramienta automatizada sirvió para probar la integración de la API con la base de datos. Mediante esta herramienta se permitió la detección de errores de manera rápida y aseguro que los endpoints cumplan con los requisitos esperados.

Casos de prueba en Postman

A continuación, se muestra los endpoints probados y los valores de pruebas esperados.

Id	Endpoint	Tipos de flujo	Resultado	Valores
1	Login	Exitoso	El sistema regresa un token	Datos válidos
2		Fallido	El sistema regresa un código 500	Datos inválidos
3	CreateAccount	Exitoso	El sistema regresa un código 201	Datos válidos
4		Fallido	El sistema regresa un código 500	Datos inválidos
5	UpdateClient	Exitoso	El sistema regresa un código 200	Datos válidos
6		Fallido	El sistema regresa un código 500	Datos inválidos
7		No autorizado	El sistema regresa un código 401	Solicitud sin token
8	CreateClient	Exitoso	El sistema regresa un código 201	Datos válidos
9		Fallido	El sistema regresa un código 500	Datos inválidos
10	GetClient	Exitoso	El sistema regresa un código 200	Datos válidos
11		No autorizado	El sistema regresa un código 401	Solicitud sin token
12	GetAccount	Exitoso	El sistema regresa un código 200	Datos válidos
13		No autorizado	El sistema regresa un código 401	Solicitud sin token
14	UpdatePassword	Exitoso	El sistema regresa un código 200	Datos válidos
15		Fallido	El sistema regresa un código 500	Datos inválidos
16	CreateRestaurant	Exitoso	El sistema regresa un código 201	Datos válidos
17		Fallido	El sistema regresa un código 500	Datos inválidos
18		No autorizado	El sistema regresa un código 401	Solicitud sin token
19	GetAllRestaurants	Exitoso	El sistema regresa un código 200	Datos válidos
20		No autorizado	El sistema regresa un código 401	Solicitud sin token
21	GetRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos
22		No autorizado	El sistema regresa un código 401	Solicitud sin token
23	UpdateRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos
24		Fallido	El sistema regresa un código 500	Datos inválidos
25		No autorizado	El sistema regresa un código 401	Solicitud sin token
26	GetRestauratByCategory	Exitoso	El sistema regresa un código 200	Datos válidos
27		Fallido	El sistema regresa un código 500	Datos inválidos
28	CreateMenu	Exitoso	El sistema regresa un código 201	Datos válidos
29		Fallido	El sistema regresa un código 500	Datos inválidos
30		No autorizado	El sistema regresa un código 401	Solicitud sin token
31	GetAllMenuRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos

32		Fallido	El sistema regresa un código 500	Datos inválidos
33	GetMenuDishRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos
34		Fallido	El sistema regresa un código 500	Datos inválidos
35	UpdateMenuDishRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos
36		Fallido	El sistema regresa un código 500	Datos inválidos
37		No autorizado	El sistema regresa un código 401	Solicitud sin token
38	DeleteMenuDishRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos
39		Fallido	El sistema regresa un código 500	Datos inválidos
40		No Autorizado	El sistema regresa un código 401	Solicitud sin token
41	CreateRatingRestaurant	Exitoso	El sistema regresa un código 201	Datos válidos
42		Fallido	El sistema regresa un código 500	Datos inválidos
43		No Autorizado	El sistema regresa un código 401	Solicitud sin token
44	GetAllRatingRestaurant	Exitoso	El sistema regresa un código 200	Datos válidos
45		Fallido	El sistema regresa un código 500	Datos inválidos
46	ValidateEmail	Exitoso	El sistema regresa un código 200	Datos válidos
47		Fallido	El sistema regresa un código 500	Datos inválidos
48	UpdateEmail	Exitoso	El sistema regresa un código 200	Datos válidos
49		Fallido	El sistema regresa un código 500	Datos inválidos
50		No Autorizado	El sistema regresa un código 401	Solicitud sin token
51	GetAllLocations	Exitoso	El sistema regresa un código 200	Datos válidos
52		Fallido	El sistema regresa un código 500	Datos inválidos
53	GetRestaurantOwner	Exitoso	El sistema regresa un código 200	Datos válidos
54		Fallido	El sistema regresa un código 500	Datos inválidos
55		No Autorizado	El sistema regresa un código 401	Solicitud sin token

Casos de prueba de funcionalidad.

ID	Objetivo	Entradas	Resultados esperados	Resultados observados	Resultado
1	Login-Exitoso	Email: alexsandermarin@outlook.com Password: 123Ale_	Inicia sesión exitosamente	Inicia sesión exitosamente	Paso
2	Login-Fallido	Email: alexsandermarin@outlook.com Password: 123mAR_	Regresa código 500	Regresa código 500	Paso
3	CreateAccount-Exitoso	Email: momaosiris@gmail.com Password: 123Pal_ AccountType: CLIENT	Regresa código 201	Regresa código 201	Paso
4	CreateAccount-Fallido	Email: momaosiris Password: 123Pal_ AccountType: cliente	Regresa código 500	Regresa código 500	Paso
5	UpdateClient-Exitoso	Name: Ale Phone: 22283248388	Regresa código 200	Regresa código 200	Paso
6	UpdateClient-Fallido	Name: Ale2 Phone: 22283248388	Regresa código 500	Regresa código 500	Paso
7	UpdateClient-NoAutorizado	Name: Ale Phone: 22283248388	Regresa código 401	Regresa código 401	Paso
8	CreateClient-Exitoso	Name: Alejandro sánchez marín Phone: 2283248388	Regresa código 201	Regresa código 201	Paso
9	CreateClient-Fallido	Name: Alejandro sánchez marín2 Phone: 2283248388f	Regresa código 500	Regresa código 500	Paso
10	GetClient-Exitoso	Id=1	Regresa código 200	Regresa código 200	Paso
11	GetClient-Fallido	Id=fsdfsd	Regresa código 500	Regresa código 500	Paso
12	GetClient-NoAutorizado	Id=1	Regresa código 401	Regresa código 401	

13	GetAccount-Exitoso	Id=1	Regresa código 200	Regresa código 200	Paso
14	GetAccount-Fallido	Id=sfds	Regresa código 500	Regresa código 500	Paso
15	GetAccount-NoAutorizado	Id=1	Regresa código 401	Regresa código 401	Paso
16	UpdatePassword-Exitoso	Email: momaosiris@gmail.com Password: 123Pal_	Regresa código 200	Regresa código 200	Paso
17	UpdatePassword-Fallido	Email: momaosiri Password: 123Pal_	Regresa código 500	Regresa código 500	Paso
18	UpdatePassword-NoAutorizado	Email_Login: momaosiris@gmail.com Password_Login:123Pal_	Regresa código 401	Regresa código 401	Paso
19	CreateRestaurant-Exitoso	RestaurantName: Chao Nayelli categoryName: hamburguesas Schedule: { Lunes: 10:00-20:00, Martes: 12:00-20:00 } PhoneNumber: 1234567890 Location: Calle Falsa 123 ImageUrl: https://th.bing.com/th/id/OIP.stiDu_94VmHdC0aSSfApowHaEu?rs=1&pid=ImgDetMain	Regresa código 201	Regresa código 201	Paso
20	CreateRestaurant-Fallido	RestaurantName: Chao Nayelli categoryName: hamburguesas Schedule: { Lunes: 10:00-20:00, Martes: 12:00-20:00 } PhoneNumber: 1234567890 Location: Calle Falsa 123	Regresa código 500	Regresa código 500	Paso
21	CreateRestaurant-NoAutorizado	RestaurantName: Chao Nayelli categoryName: hamburguesas Schedule: { Lunes: 10:00-20:00, Martes: 12:00-20:00 } PhoneNumber: 1234567890 Location: Calle Falsa 123 ImageUrl: https://th.bing.com/th/id/OIP.stiDu_94VmHdC0aSSfApowHaEu?rs=1&pid=ImgDetMain	Regresa código 401	Regresa código 401	Paso
22	GetAllRestaurants-Exitoso	N/A	Regresa código 200	Regresa código 200	Paso
23	GetAllRestaurants-Fallido	N/A	Regresa código 500	Regresa código 500	Paso
24	GetAllRestaurants-NoAutorizado	N/A	Regresa código 401	Regresa código 401	Paso
25	GetRestaurant-Exitoso	Id=1	Regresa código 200	Regresa código 200	Paso
26	GetRestaurant-Fallido	Id=4dsa23	Regresa código 500	Regresa código 500	Paso
27	GetRestaurant-NoAutorizado	Id=1	Regresa código 401	Regresa código 401	Paso
28	UpdateRestaurant-Exitoso	Schedule: { Lunes: 10:00-20:00, Martes: 12:00-20:00 } PhoneNumber: 1234567890 Location: Calle Falsa 123 ImageUrl: https://th.bing.com/th/id/OIP.stiDu_94VmHdC0aSSfApowHaEu?rs=1&pid=ImgDetMain	Regresa código 200	Regresa código 200	Paso
29	UpdateRestaurant-Fallido	Schedule: { Lunes: 10:00-20:00, Martes: 12:00-20:00 } PhoneNumber: dsadsadsa Location: Calle Falsa 123 ImageUrl:	Regresa código 500	Regresa código 500	Paso
30	UpdateRestaurant-NoAutorizado	Schedule: { Lunes: 10:00-20:00, Martes: 12:00-20:00 } PhoneNumber: 1234567890 Location: Calle Falsa 123	Regresa código 401	Regresa código 401	Paso

		ImageUrl: https://th.bing.com/th/id/OIP.stiDu_94VmHdC0aSSfApowHaEu?rs=1&pid=ImgDetMain			
31	GetRestaurantByCategory-Exitoso	Category=pizzas	Regresa código 200	Regresa código 200	Paso
32	GetRestaurantByCategory-Fallido	Category=ensaladas	Regresa código 500	Regresa código 500	Paso
33	GetRestaurantByCategory-NoAutorizado	Category=pizzas	Regresa código 401	Regresa código 401	Paso
34	CreateMenu-Exitoso	Dish: Pizza orilla de queso 2 Price: 500 ImageUrl: http//pizza.png Description: Pizza orilla de queso muy rica	Regresa código 201	Regresa código 201	Paso
35	CreateMenu-Fallido	Dish: Pizza orilla de queso 2 Price: dasfs ImageUrl: Description: Pizza orilla de queso muy rica	Regresa código 500	Regresa código 500	Paso
36	CreateMenu-NoAutorizado	Dish: Pizza orilla de queso 2 Price: 500 ImageUrl: http//pizza.png Description: Pizza orilla de queso muy rica	Regresa código 401	Regresa código 401	Paso
37	GetAllMenuRestaurant-Exitoso	RestaurantName=chaonayelly	Regresa código 200	Regresa código 200	Paso
38	GetAllMenuRestaurant-Fallido	RestaurantName=chaonayelly2	Regresa código 500	Regresa código 500	Paso
39	GetAllMenuRestaurant-NoAutorizado	RestaurantName=chaonayelly	Regresa código 401	Regresa código 401	Paso
40	GetMenuDishRestaurant-Exitoso	RestaurantName=chaonayelly Dish=Pizza	Regresa código 200	Regresa código 200	Paso
41	GetMenuDishRestaurant-Fallido	RestaurantName=chaonayelly2 Dish=pizza2	Regresa código 500	Regresa código 500	Paso
42	GetMenuDischRestaurant-NoAutorizado	RestaurantName=chaonayelly Dish=Pizza	Regresa código 401	Regresa código 401	Paso
43	UpdateMenuDishRestaurant-Exitoso	Dish: Pizza orilla de queso Price: 500 ImageUrl: http//pizza.png Description: Pizza orilla de queso muy rica	Regresa código 200	Regresa código 200	Paso
44	UpdateMenuDishRestaurant-Fallido	Dish: Pizza orilla de queso Price: fdfsd ImageUrl: http//pizza.png Description:	Regresa código 500	Regresa código 500	Paso
45	UpdateMenuDishRestaurant-NoAutorizado	Dish: Pizza orilla de queso Price: 500 ImageUrl: http//pizza.png Description: Pizza orilla de queso muy rica	Regresa código 401	Regresa código 401	Paso
46	DeleteMenuDishRestaurant-Exitoso	Dish=Pizza	Regresa código 200	Regresa código 200	Paso
47	DeleteMenuDishRestaurant-Fallido	Dish=Pizza1	Regresa código 500	Regresa código 500	Paso
48	DeleteMenuDishRestaurant-NoAutorizado	Dish=Pizza	Regresa código 401	Regresa código 401	Paso
49	CreateRatingRestaurant-Exitoso	Rate:5	Regresa código 201	Regresa código 201	Paso
50	CreateRatingRestaurant-Fallido	Rate:Cinco	Regresa código 500	Regresa código 500	Paso
51	CreateRatingRestaurant-NoAutorizado	Rate:5	Regresa código 401	Regresa código 401	Paso
52	GetAllRatingRestaurant-Exitoso	RestaurantName=chaonayelly	Regresa código 200	Regresa código 200	Paso

53	GetAllRatingRestaurant-Fallido	RestaurantName=chaonayelly	Regresa código 500	Regresa código 500	Paso
54	GetAllRatingRestaurant-NoAutorizado	RestaurantName=chaonayelly	Regresa código 401	Regresa código 401	Paso
55	ValidateEmail-Exitoso	Email=alexsandermarin@outlook.com	Regresa código 200	Regresa código 200	Paso
56	ValidateEmail-Fallido	Email=ale	Regresa código 500	Regresa código 500	Paso
57	ValidateEmail-NoAutorizado	Email=alexsandermarin@outlook.com	Regresa código 401	Regresa código 401	Paso
58	UpdateEmail-Exitoso	Id=1 Email: alexsandermarin@outlook.com	Regresa código 200	Regresa código 200	Paso
59	UpdateEmail-Fallido	Id=fds Email: ale	Regresa código 500	Regresa código 500	Paso
60	UpdateEmail-NoAutorizado	Id=1 Email=alexsandermarin@outlook.com	Regresa código 401	Regresa código 401	Paso
61	GetAllLocations-Exitoso	N/A	Regresa código 200	Regresa código 200	Paso
62	GetAllLocations-Fallido	N/A	Regresa código 500	Regresa código 500	Paso
63	GetRestaurantOwner-Exitoso	N/A	Regresa código 200	Regresa código 200	Paso
64	GetRestaurantOwner-Fallido	N/A	Regresa código 500	Regresa código 500	Paso
65	GetRestaurantOwner-NoAutorizado	N/A	Regresa código 401	Regresa código 401	Paso

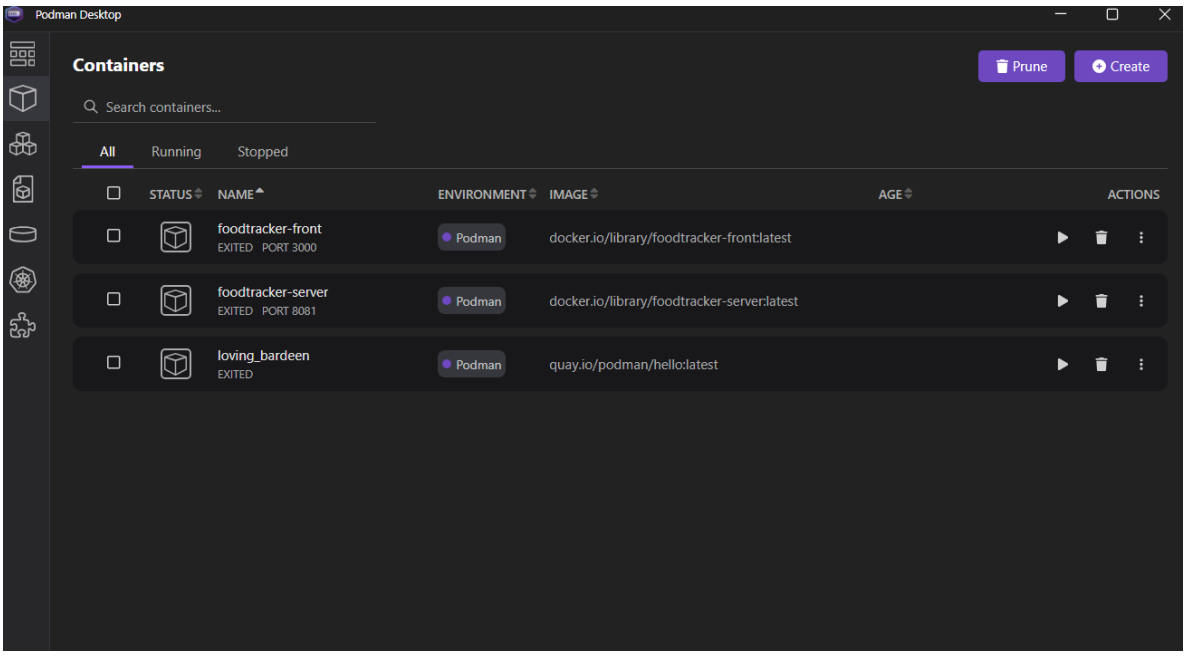
Estrategia de despliegue

La estrategia de despliegue del proyecto se basó en el uso de Podman para la creación y gestión de imágenes de contenedores. Estas imágenes permiten la distribución del proyecto a diferentes usuarios, asegurando la portabilidad y la consistencia del entorno de ejecución de la aplicación. La construcción de las imágenes se realizó localmente mediante el comando podman build, utilizando un archivo Dockerfile en cada proyecto de FoodTracker, tanto para el frontend como para el backend. Cada Dockerfile fue adaptado a las tecnologías específicas de cada proyecto.

En el caso del frontend, desarrollado con ReactJS, se enfrentaron algunos desafíos. Uno de ellos fue relacionado con la versión del gestor npm, mientras que otro se debió a los archivos generados durante la compilación, como la carpeta node_modules, que ocupan un espacio significativo. Para resolver esto, se incluyó un archivo .dockerignore para excluir estas carpetas y otros archivos innecesarios. Además, surgió un problema con el paquete de ejecución npx: al no configurarse adecuadamente en modo de Single Page Application (SPA), algunas rutas de la aplicación no se cargaban correctamente. Esto se solucionó especificando el uso del modo SPA en el comando de ejecución.

Finalmente, tras crear los contenedores, las imágenes resultantes se subieron a Docker Hub para facilitar su acceso y uso por parte de otros desarrolladores.

En general, la elección de Podman como base de la estrategia fue adecuada debido a su similitud con Docker, lo que permitió reutilizar conocimientos previos sin una curva de aprendizaje significativa.



Conclusiones

Durante el desarrollo de *FoodTracker*, nos sentimos seguros al trabajar con tecnologías familiares como Java y MySQL, las cuales facilitaron la creación del backend y la base de datos. Sin embargo, nos enfrentamos a retos significativos al implementar mecanismos de seguridad como Json Web Token (JWT) y Spring Security. Aunque ya teníamos nociones previas sobre autenticación y autorización, la configuración en Spring Boot requirió una mayor investigación y dedicación debido a su enfoque de autoconfiguración.

En el frontend, ReactJS resultó ser una herramienta muy interesante por la creación de componentes reutilizables, aunque el desconocimiento inicial sobre su arquitectura de aplicaciones de una sola página (SPA) complicó el despliegue utilizando Podman. Resolver estos problemas, como la configuración adecuada de rutas y la optimización del entorno de contenedores, requirió mayor esfuerzo, pero nos permitió aprender valiosas estrategias para gestionar aplicaciones modernas. A pesar de las dificultades técnicas, logramos concluir el proyecto, integrando tecnologías modernas y cumpliendo con los requisitos planteados.