Reingeneria Comandas AS Burger

Instituto Tecnológico Sanmiguelense de Estudios Superiores

Alumno:

Erick Gael Barbosa Cuarenta

Materia:

Reingeneria

Docente:

Lizeth Abril Gonzalez Vazquez

10 de Diciembre de 2025

Indice

Capítulo 2 Etapas de la Reingeniería

Antecedentes:

Este capítulo pretende orientar en las actividades que pueden aplicarse a un proceso de reingeniería de software, bajo la óptica de los diferentes componentes que integran un sistema automático de datos, esto es, programas, bases de datos, documentación, y hardware, los mismos que deberán aplicarse según las circunstancias que rodean a cada caso específico.

Es de entenderse que cada caso es particular, por lo que su resolución puede requerir la aplicación de una o varias de los pasos de reingeniería con un nivel de abstracción diferente.

A continuación se definen las etapas de un proceso de reingeniería:

2.1 Análisis del Inventario:

Atributo	Valor
Nombre de la Aplicación	ComandAS Burger
Año de Creacion	Abril 2024
Número de cambios Importantes	3
Esfuerzo empleado en cambios	20 horas
Fecha del último cambio	Agosto 2024
Base de datos	MySQL
Numero de usuarios	N/A
Numero de máquinas instaladas	1
Complejidad de Arquitectura	Simple
Codigo	Compleja
Documentación	No existe
Calidad de la Documentación	No existe
Longevidad del proyecto	4 meses

Número de cambios aproximadamente (cada 36 meses)	N/A

2.2 Reestructuración de Documentos:

En vista que el sistema actualmente no dispone de ningún tipo de documentación, es parte de las necesidades del usuario contar con documentación completa de manuales técnicos y de usuario.

Estos manuales se adjuntan a este documento.

2.3 Ingeniería Inversa.

A continuación se presenta un cuadro en el cual se especifica la funcionalidad del sistema, con sus respectivos módulos y opciones actuales, el mismo que servirá de base a los nuevos requerimientos del usuario.

Adicionalmente se adjunta en los anexos 1 y 2, los respectivos diagramas de entidad relación, y los diagramas de flujo de datos (DFDs) que representan la estructura actual de la aplicación.

2.3.1 Modulos del Frontend

Componentes Publicos

Módulo / Componente	Descripción
Public Component	Componente que contiene las rutas públicas de la aplicación y muestra las vistas de acceso, como el inicio de sesión.
SignIn Component	Componente encargado del inicio de sesión. Valida las credenciales del usuario y lo redirige según su rol.
SignUp Component	No tiene funcionalidad en el estado actual

Componentes Privados

Módulo / Componente	Descripción
Private Component	Componente que contiene la estructura y rutas principales de la parte privada de la aplicación, accesible solo para usuarios autenticados.
User-View Component	Muestra la lista de usuarios en una tabla con búsqueda, ordenamiento y paginación. Permite visualizar sus datos y el rol asignado.
User Component	Componente para crear, editar o eliminar usuarios. Muestra un formulario con los datos del usuario y envía los cambios al sistema.

Orders-View Component	Muestra todas las órdenes organizadas por estado. Permite ver detalles, completar o cancelar órdenes y actualiza la lista automáticamente en tiempo real.
Order-View Component	El usuario revisa su orden antes de enviarla. Aquí puede ver productos, extras, totales, elegir el tipo de pedido y finalmente confirmar la compra.
Order-Detail Component	Muestra los detalles de una orden dentro de una ventana emergente y permite marcarla como lista, avisando en tiempo real a los demás usuarios.
Menu Component	Muestra todos los productos del menú y permite armar una orden. El usuario puede agregar o quitar productos, elegir ingredientes y ver un panel lateral con los detalles de lo que va seleccionando.
Dialog-Complete Component	Muestra una ventana para confirmar que una orden está completada. Al aceptarlo, actualiza el estado de la orden y envía la notificación en tiempo real.
Dialog-Cancel Component	Muestra una ventana para confirmar la cancelación de una orden. Al aceptar, cambia su estado a cancelada y envía la actualización en tiempo real.
Dialog Component	Muestra una ventana para iniciar el proceso de una orden. Al continuar, cambia su estado a “en proceso”, abre los detalles de la orden y envía la actualización en tiempo real.

Dash-Admin Component	Muestra un panel con estadísticas del negocio, como ventas, productos más vendidos y mejores clientes. Los datos se actualizan en tiempo real para mantener las gráficas al día.
Chef-Order-View Component	Muestra al cocinero las órdenes que acaban de llegar. Permite ver sus detalles, comenzar a prepararlas y recibe actualizaciones en tiempo real conforme entran nuevas órdenes.

Servicios del Frontend

Módulo / Componente	**Descripción**
Ingredient Interface	Define la estructura básica de un ingrediente, incluyendo su nombre, costo, stock y si es obligatorio o extra. Sirve para organizar y manejar la información de cada ingrediente dentro de la aplicación.
Product Interface	Define la información esencial de un producto, como su nombre, precio, categoría y la lista de ingredientes que lo componen. Ayuda a organizar y mostrar cada producto dentro de la app.

Guards del Frontend

Módulo / Componente	Descripción
Auth Guard	Función que decide si un usuario puede entrar a una ruta. En este caso, siempre permite el acceso.

Environment

Módulo / Componente	Descripción
Environment	Guarda configuraciones básicas de la app, como si está en modo desarrollo y la dirección del servidor de sockets.

2.3.2 Modulos del Backend

Controller

Módulo / Controller	Descripción
Auth Controller	Controla el inicio de sesión y registro de usuarios, enviando los datos al modelo.
Graphics Controller	Se encarga de obtener datos estadísticos de la aplicación, como ventas totales, productos más vendidos, mejores clientes, ventas por mes y tiempo promedio.
Menu Controller	Se encarga de manejar las solicitudes relacionadas con el menú, específicamente para obtener la lista de productos e ingredientes.

Order Controller	Gestiona las operaciones relacionadas con las órdenes. Permite crear nuevas órdenes, obtener todas las órdenes o una específica, actualizar el estado de una orden y recuperar la última orden de un usuario.
User Controller	Gestiona usuarios: ver todos, ver uno, actualizar datos y eliminar, siempre devolviendo respuestas en JSON.

Models

Módulo / Models	**Descripción**
Auth Model	Maneja el inicio y registro de usuarios, generando tokens y ocultando datos sensibles como la contraseña.
Graphics Model	Obtiene y organiza datos estadísticos como ventas totales, productos más vendidos, clientes frecuentes y tiempos promedio de pedidos.
Menu Model	Proporciona la información de productos e ingredientes disponibles en el menú.
Order Model	Gestiona las órdenes, incluyendo creación, visualización, actualización de estado y consulta de la última orden de un usuario.
User Model	Gestiona la información de los usuarios, permitiendo ver, actualizar y eliminar datos a través del servicio correspondiente.

Service

Módulo / Service	Descripción
Auth Service	Maneja la autenticación de usuarios: inicio de sesión con verificación de contraseña y registro de nuevos usuarios con hash de contraseña y envío opcional de webhook.
Graphics Service	Proporciona datos estadísticos para gráficos: mejores productos, mejores clientes, ventas mensuales y tiempo promedio de preparación.
Menu Service	Gestiona la información de productos e ingredientes: obtiene productos activos y sus ingredientes asociados.
Order Service	Gestiona las órdenes: creación completa de órdenes con detalles e ingredientes removidos, actualización de estado, consulta de órdenes activas o por usuario y última orden de un cliente.
User Service	Gestiona los usuarios: ver todos o uno, actualizar datos y desactivar usuarios en la base de datos.

Config / Database

Módulo / Config	Descripción
Connection	Configura la conexión a la base de datos MySQL usando PDO, con manejo de errores.

Methods	Proporciona métodos para ejecutar consultas SQL: query (varios resultados), query_one (un solo resultado), save (insert/update) y save_transaction (varias consultas en transacción), con manejo de errores y cierre de conexión.

Config / Jwt

Config/ Jwt	**Descripción**
Jwt	Maneja la generación y verificación de tokens JWT. Incluye métodos SignIn para crear un token con datos del usuario, Check para validar un token según IP y emisión, y GetData para extraer los datos del token.

Config / Utils

Config/ Utils	**Descripción**
Custom Exception	Maneja errores personalizados con códigos y mensajes predefinidos.
Utils	Funciones útiles: generar UUID, encriptar/verificar contraseñas, revisar parámetros vacíos, obtener IP y enviar datos a webhooks o sockets.

Public

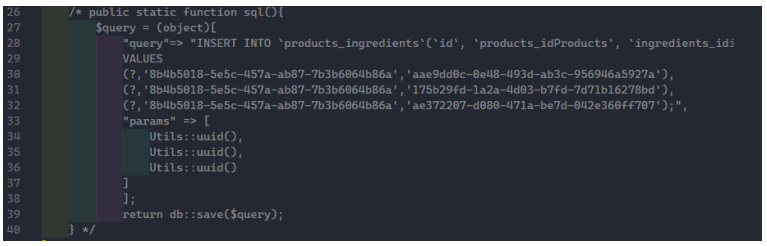
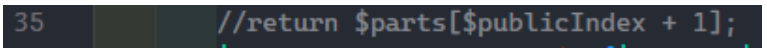
Módulo / Public	Descripción
Index	Punto de entrada del backend: carga clases, configura headers, obtiene la ruta y el método HTTP de la solicitud, y llama al router para manejar la petición.
.htaccess	Redirige todas las solicitudes a Index.php si no existe el archivo o directorio solicitado, para permitir URLs limpias (pretty URLs) con RewriteEngine.

2.4. Reestructuración de datos y de código:

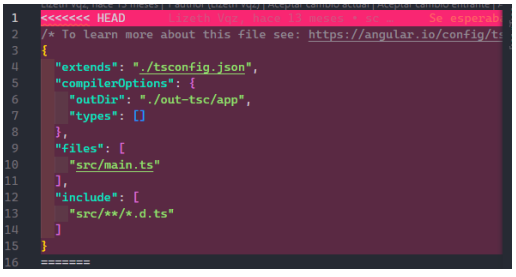
La reestructuración de datos y código es un proceso fundamental de limpieza y optimización del sistema existente antes de implementar nuevas funcionalidades. Este proceso busca eliminar elementos obsoletos, mejorar la organización del código y optimizar la estructura de datos para facilitar el mantenimiento y la escalabilidad del sistema..

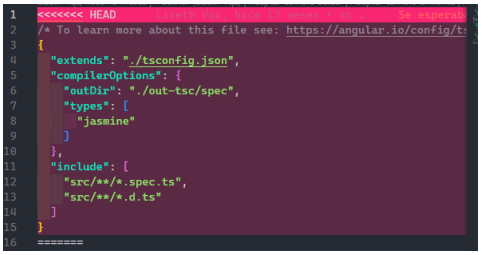
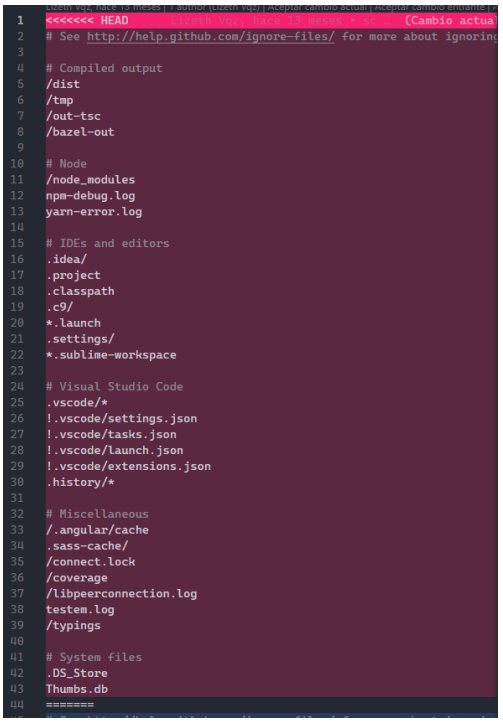
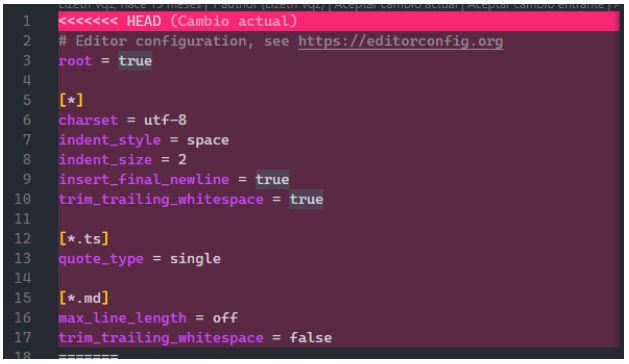
Al analizar el sistema ComandAS Burger en su versión actual, se detectaron varias áreas que necesitan mejoras y reestructuración, especialmente en el código:

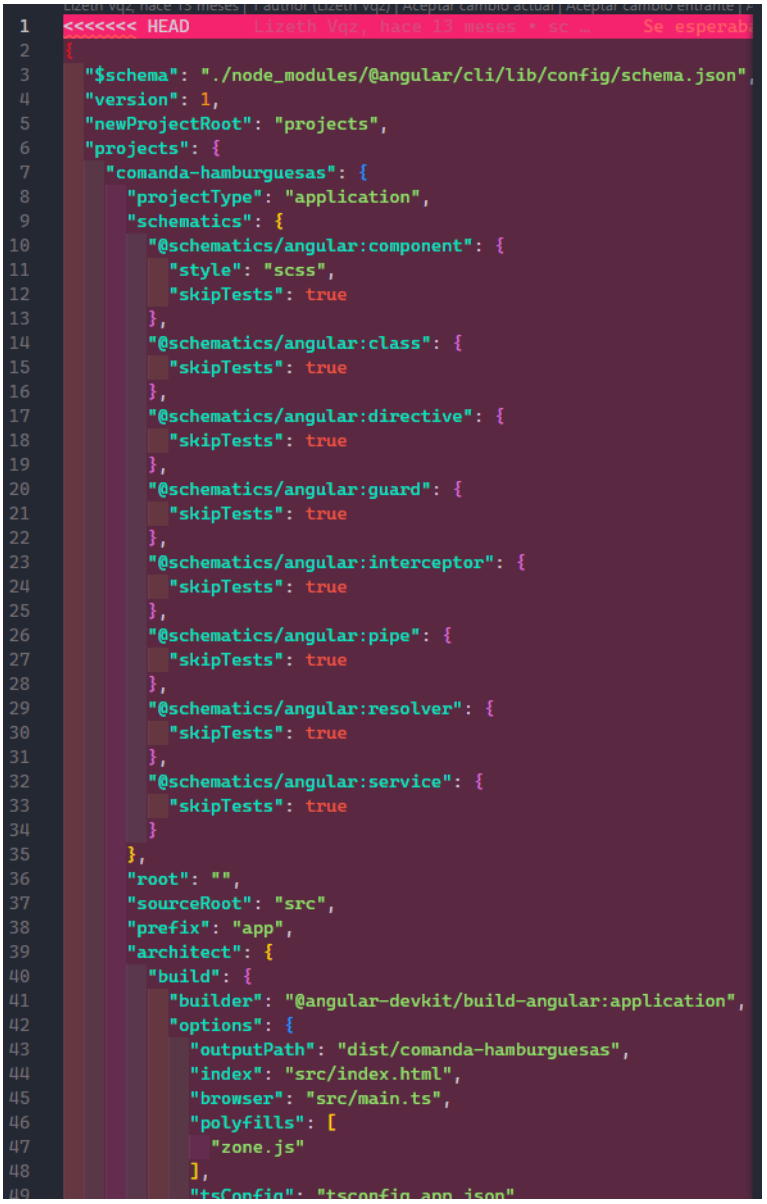
Backend

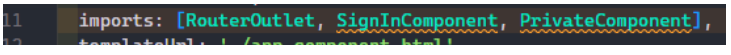
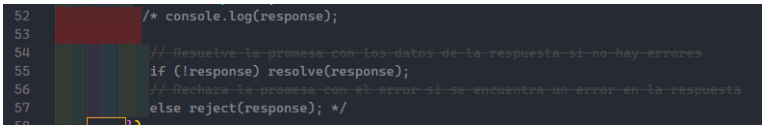
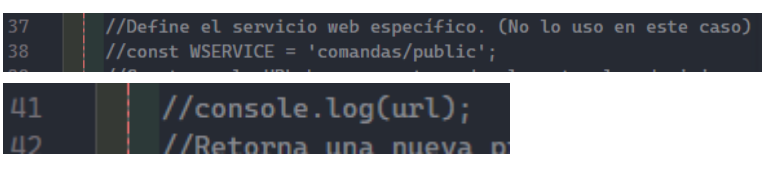
BACKEND	
Código comentado y sin uso	
Archivo	**Codigo encontrado**
MenuService.php	
Index.php	

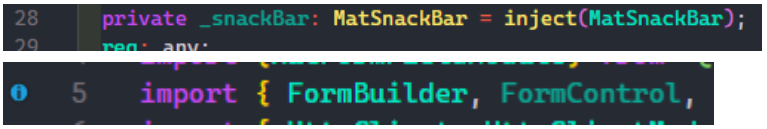
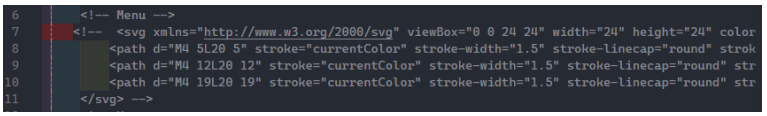
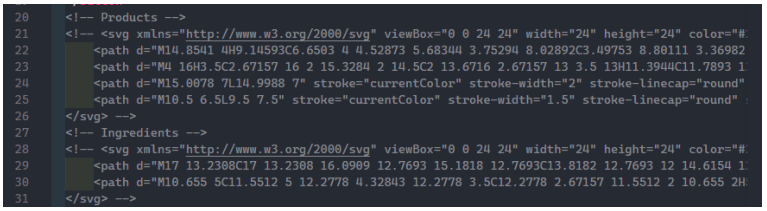
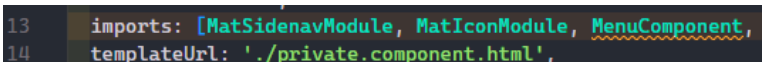
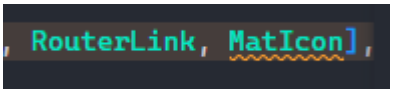
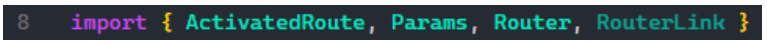
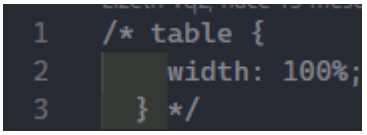
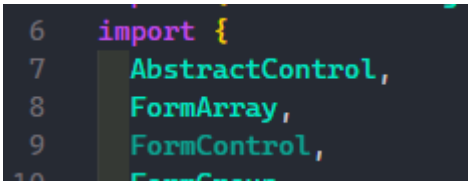
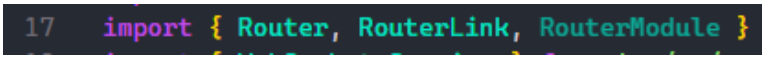
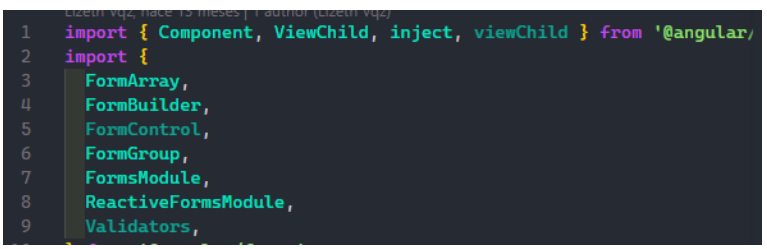
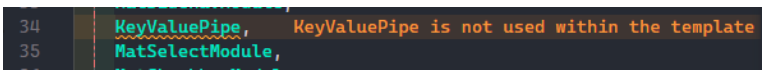
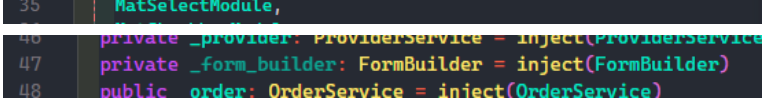
Frontend

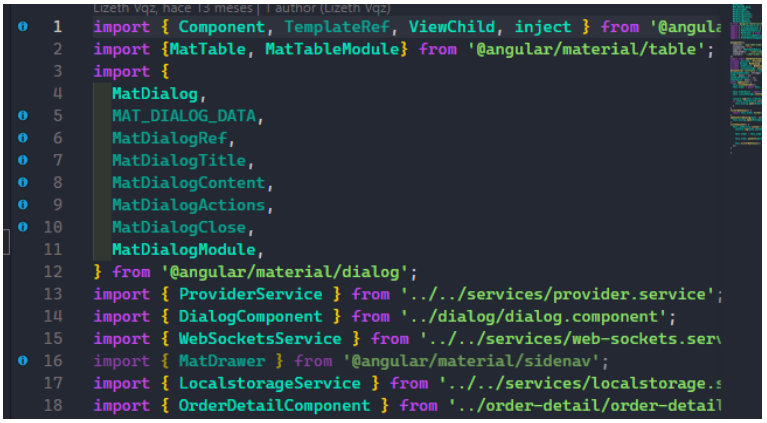
FRONTEND	
Correcciones (merge conflict)	
Archivo	**Codigo encontrado**
tsconfig.json	
tsconfig.app.json	

tsconfig.spec.json	
.gitignore	
.editorconfig	

angular.json	

FRONTEND	
Importaciones, código y variables sin usar	
Archivo	**Codigo encontrado**
app.component.ts	
web-sockets.service.ts	
provider.service.ts	

sign-in.component.ts	
private.component.html	 
private.component.ts	
user-view.component.ts	
user.component.ts	 
orders-view.component.ts	
orders-view.component.scss	
order-view.component.ts	  
menu.component.ts	   

chef-order-view.component.ts	
auth.guard.ts	

Una vez finalizado este proceso de revisión, corrección y optimización del código, la aplicación estará en condiciones óptimas para implementar nuevas funcionalidades, garantizando una base de código clara, organizada y consistente, que facilitará el desarrollo, la escalabilidad y el mantenimiento a largo plazo.

2.5. Ingeniería Progresiva:

La teoría define a la Ingeniería Progresiva como la "*actividad de manejar sistemas antiguos con tecnología del presente para las necesidades del futuro*".

Bajo esta óptica, se implementarán los requerimientos que se detallan a continuación, asegurando que los procedimientos actuales se documenten de forma que los cambios puedan aplicarse sin afectar significativamente las bases y procesos existentes. El sistema se parametrizará lo más posible para adaptarse a las nuevas funcionalidades.

Los requerimientos a implementar incluyen:

- Hacer la conexión con los servicios de php.
- Colocar un botón en la sección de login para permitir el registro de nuevos clientes.
- Permitir que el cliente registrado pueda ver sus órdenes (Registradas, En cocina, Orden lista, Completadas y Canceladas), consultar el menú, agregar órdenes y editar su perfil.
- Quitar el despliegue del menú del cocinero, manteniendo únicamente la opción de salir.
- Rediseñar la página según criterio del equipo de desarrollo.
- En el perfil del administrador, crear un apartado para un CRUD de usuarios (excluyendo la creación de clientes, únicamente los demás roles).
- Implementar el envío de correos al cliente al registrarse en la aplicación y cuando su orden esté lista, utilizando Pipedream y consumiéndolo desde Node.js.

Capítulo 3 Análisis de Requerimientos

Los requerimientos actuales buscan escalar el sistema de una herramienta de gestión interna a una plataforma mixta (interna y externa), permitiendo la interacción directa del cliente final. A continuación se detallan los puntos clave para la evolución del software:

1. Implementar la funcionalidad de auto-registro en el módulo de autenticación:

Con la finalidad de permitir que usuarios externos creen sus propias credenciales sin intervención administrativa.

- De esta forma, el ecosistema de usuarios se amplía a cuatro roles jerarquizados: Administrador, Cajero y Cocinero (personal interno), y ahora el Cliente (usuario externo).
- Actualmente, el ingreso depende exclusivamente de cuentas creadas por la administración. Al incorporar un botón visible ("Registrarse") en la pantalla de login, se descentraliza la gestión de usuarios, permitiendo que el cliente ingrese sus datos personales y acceda inmediatamente a los servicios, agilizando el flujo de venta.

2. Desarrollar un panel de autogestión exclusivo para el perfil de Cliente:

Que permita al usuario interactuar con el negocio de manera autónoma una vez autenticado.

- El cliente debe contar con herramientas para consultar el menú digital, agregar productos al carrito, editar su información de perfil y visualizar en tiempo real el ciclo de vida de sus órdenes mediante estados definidos: *Registrada, En cocina, Orden lista, Completada* y *Cancelada*.
- Esto responde a la necesidad de transparencia en el servicio, eliminando la dependencia de consultar verbalmente al mesero o cajero sobre el estatus de un pedido.

3. Implementar un módulo de administración de personal (CRUD) restringido:

Diseñado para que el Administrador gestione únicamente a la fuerza laboral del restaurante.

- Se requiere una interfaz para Crear, Leer, Actualizar y Eliminar cuentas de empleados (Cajeros, Cocineros y otros Administradores), excluyendo explícitamente la manipulación de las cuentas de Clientes, las cuales son de propiedad privada del usuario.
- Anteriormente, la gestión de usuarios no distinguía claramente entre roles operativos y consumidores. Esta separación garantiza la integridad de los datos de los clientes y focaliza la labor administrativa en el personal operativo.

4. Optimizar y restringir la interfaz gráfica del perfil de Cocinero:

Con el objetivo de maximizar la eficiencia en el área de preparación de alimentos.

- Se eliminarán opciones innecesarias para este rol, como el despliegue del menú de ventas o configuraciones de cuenta, manteniendo en pantalla únicamente el tablero de comandas activas y la opción de cerrar sesión ("Salir").
- Al ser un entorno de trabajo bajo presión, el sistema actual presentaba distractores visuales. La nueva interfaz "limpia" reduce la carga cognitiva y evita errores operativos involuntarios.

5. Integrar servicios de notificación automatizada y conexión Backend:

Para cerrar el ciclo de comunicación entre el sistema y el usuario final.

- Se implementará el consumo de servicios PHP existentes y se integrará una capa de notificaciones mediante *Pipedream* y *Node.js* para enviar correos electrónicos en dos momentos críticos: confirmación de registro de cuenta y alerta de "Orden Lista".
- Esto moderniza la experiencia del usuario, brindando confirmaciones de sus acciones y avisos oportunos sin necesidad de estar mirando la pantalla constantemente.

6. Rediseñar la interfaz de usuario (UI) bajo nuevos criterios estéticos:

- La actualización gráfica se aplicará transversalmente a todos los módulos para mejorar la usabilidad.
- Dado que el sistema será utilizado ahora por clientes finales y no solo por personal capacitado, la interfaz debe ser más intuitiva, moderna y fácil de navegar.

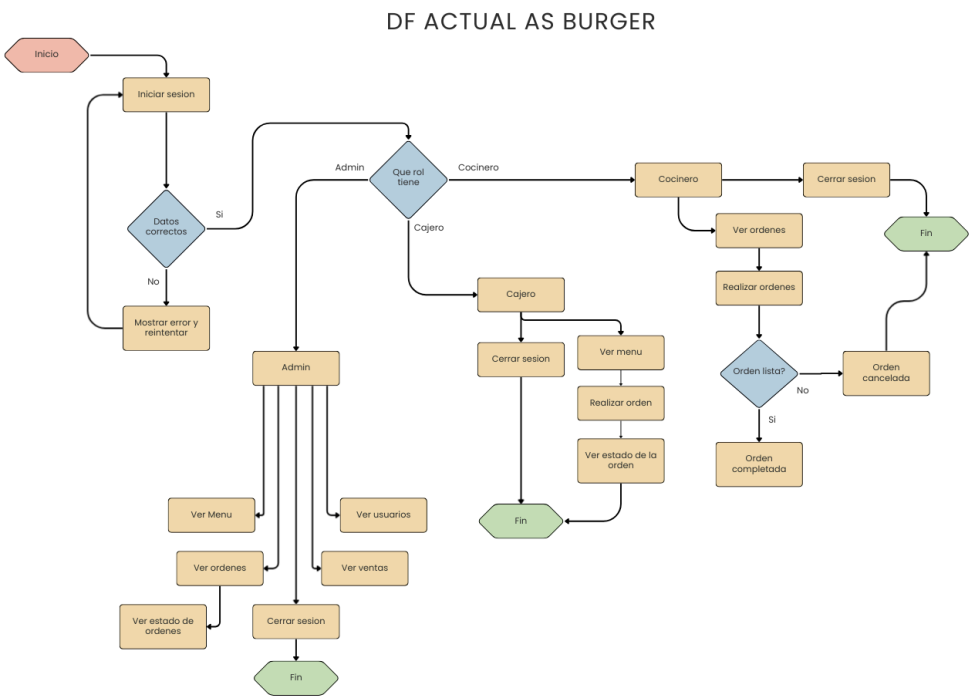
Capítulo 4 Desarrollo del Sistema

4.1 Diagramas UML

A continuación se presenta los diagramas UML necesarios para documentar e implementar los requerimientos planteados por el usuario del sistema.

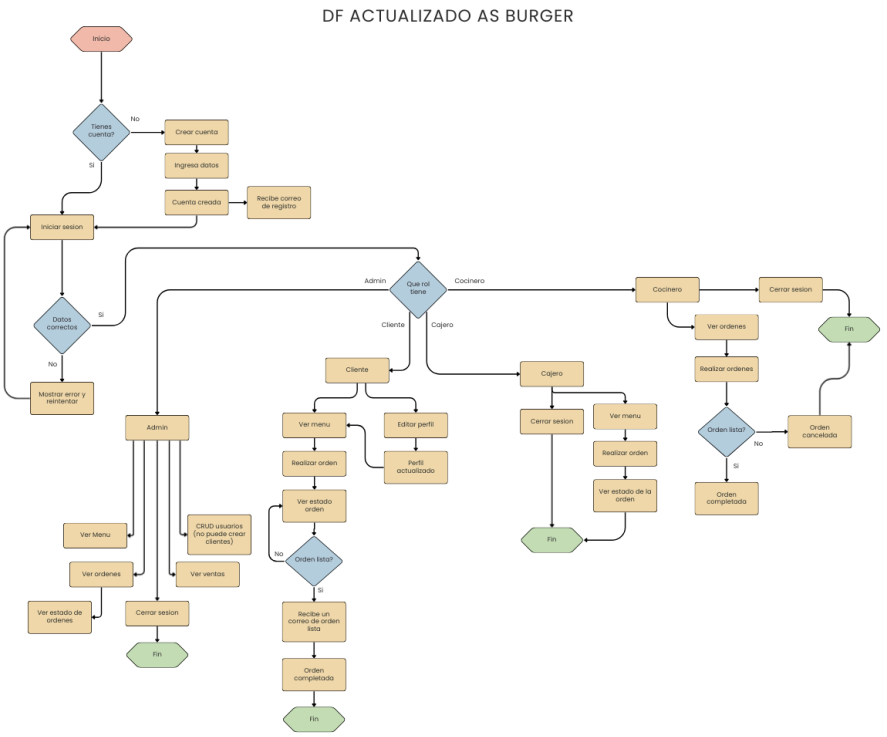
4.1.1 Diagrama de flujo (Actual)

Enlace al diagrama actual



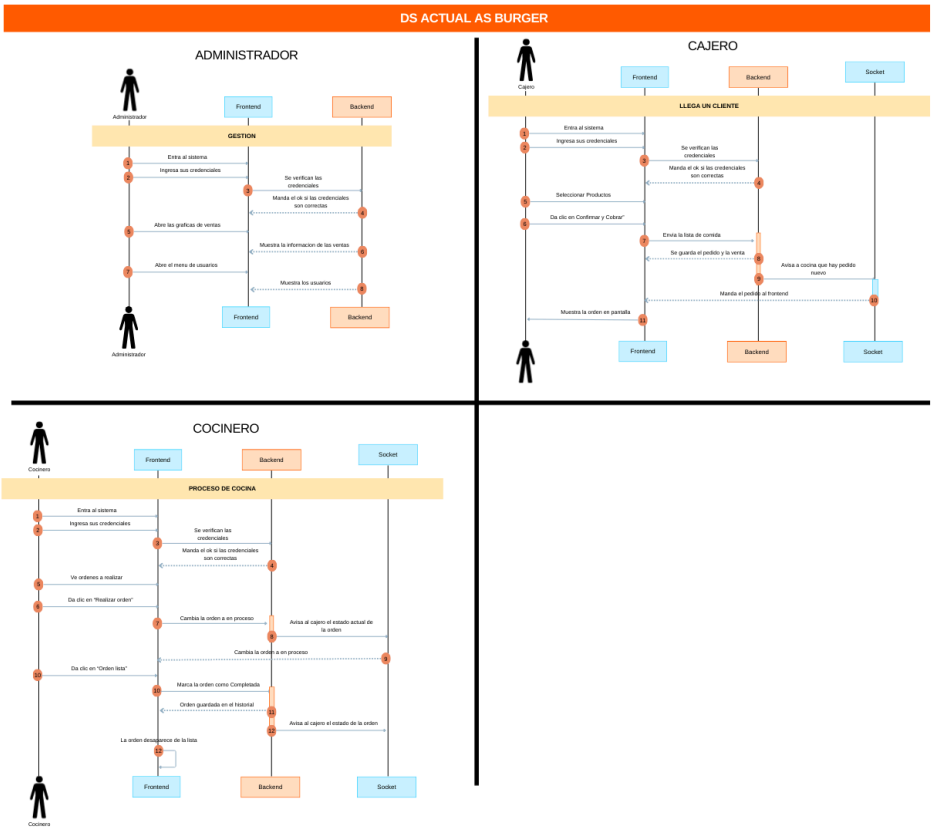
4.1.2 Diagrama de flujo (Actualizado)

Enlace al diagrama actualizado



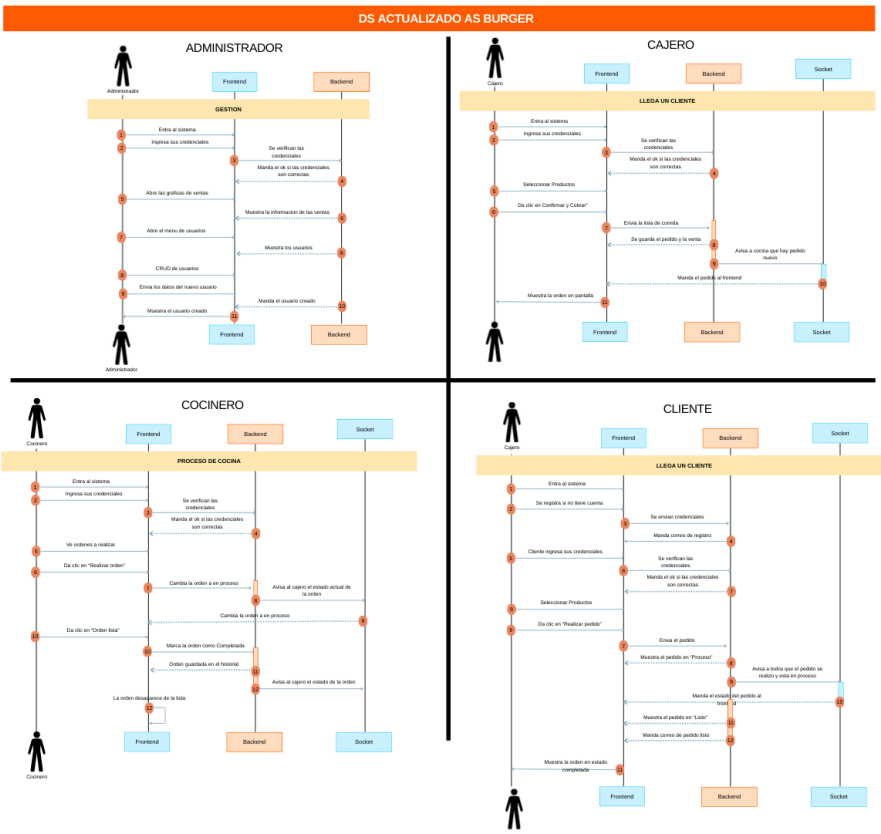
4.2.1 Diagrama de Secuencia (Actual)

[Enlace al Diagrama de Secuencia (actual)]



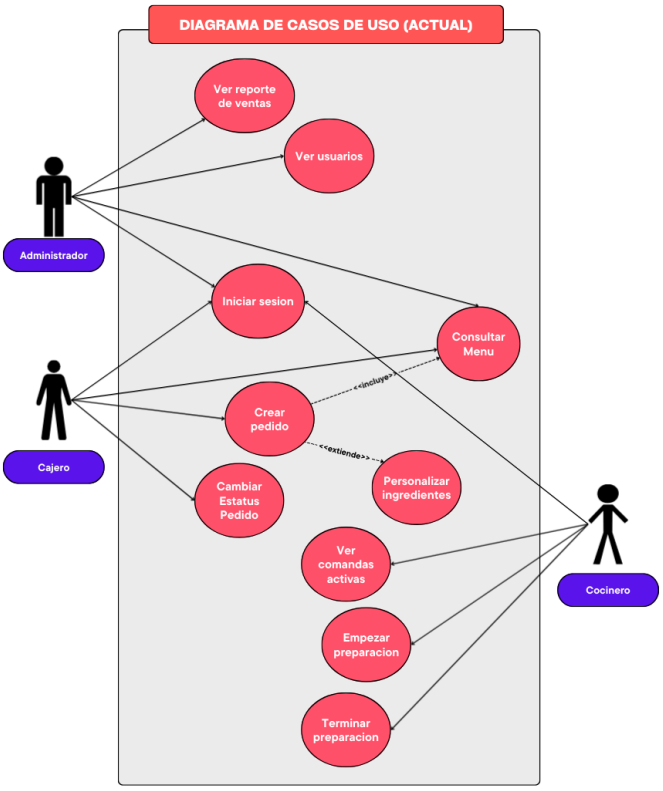
4.2.2 Diagrama de Secuencia (Actualizado)

[Enlace al Diagrama de Secuencia (Actualizado)]



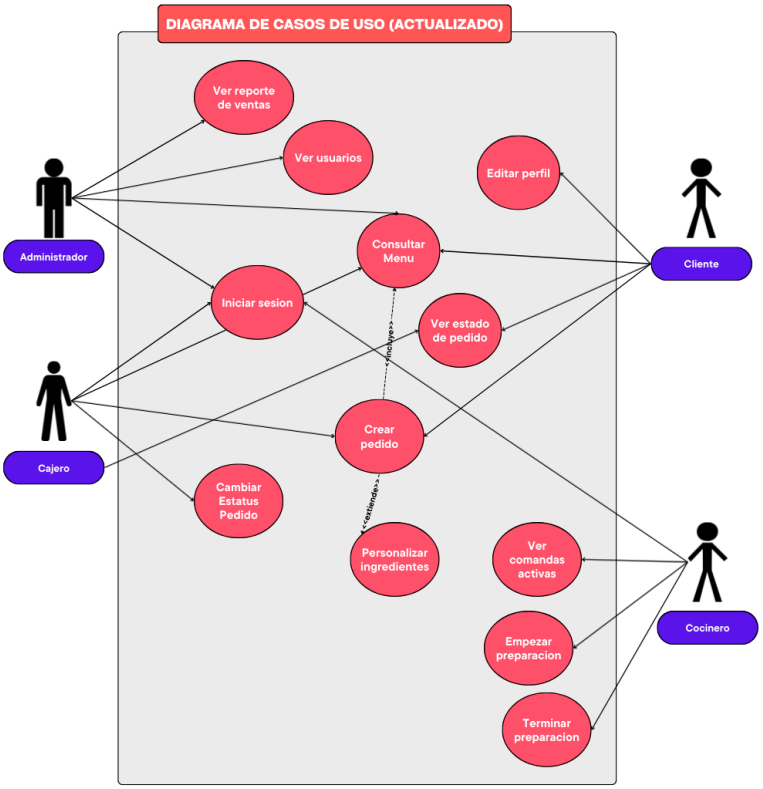
4.3.1 Diagrama de Casos de Uso (Actual)

[Enlace al Diagrama de Casos de Uso (Actual)]



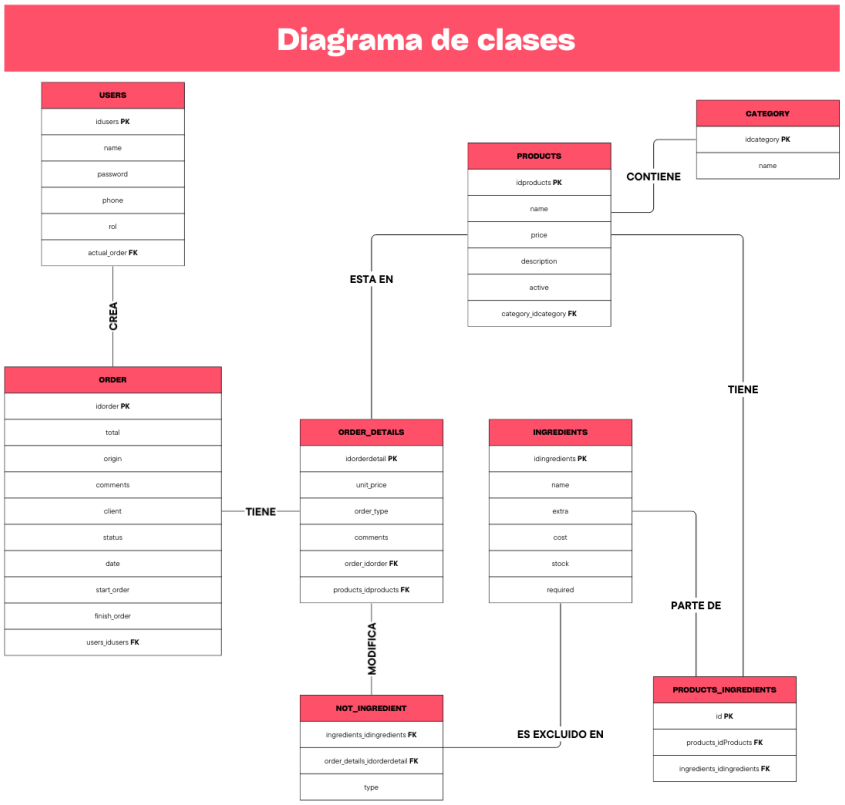
4.3.2 Diagrama de Casos de Uso (Actualizado)

[Enlace al Diagrama de Casos de Uso (Actualizado)]



4.4.1 Diagrama de Clases

[Enlace al Diagrama de Clases]



Conclusion

En conclusión, la reingeniería es algo que se vive mucho en la industria real. Casi todas las empresas ya tienen sistemas funcionando a su manera. Es raro que alguna empresa te pida hacer algo desde cero, casi todo ya esta hecho y te ponen a hacer un trabajo como este, desde ver como funciona el sistema hasta intentar descifrar que base de datos es la del sistema. Ahí es donde se pone en práctica un trabajo como este, entender ese sistema aunque no haya manuales ni documentación completa.

En este proceso la ingeniería inversa ayuda muchísimo para entender el sistema desde el código y no solo viendo las pantallas. También la reestructuración, que no es solo borrar "código basura" o arreglar fallos, sino tratar de entender la lógica que tenía el desarrollador original. El programar también es intentar entender a los demás.

Esta tarea me pareció un desafío considerable, sobre todo al inicio, ya que al no contar con la base de datos prácticamente tuve que deducir su estructura. Sin embargo, conforme fui avanzando y tomando ritmo, el sistema se volvió mucho más claro; entre más analizaba el código, mejor lo entendía. Siento que fue una experiencia muy valiosa que me servirá para enfrentar trabajos futuros.