UTEC Universidad Tecnológica | FACULTAD DE INGENIERÍA | UNIVERSIDAD DE LA REPÚBLICA URUGUAY | ANEP | UTU | DIRECCIÓN GENERAL DE EDUCACIÓN TÉCNICO PROFESIONAL

Proyecto Tecnólogo en Informática

Sistema para gestión de restaurantes: “LA PUERTA VERDE SOFTWARE”

Informe Final
Diciembre 2023

Integrantes:

Guzmán Vera Adinolfi	5.325.898-5
Hernán Cabara Brignoni	5.161.872-9
Mario Caró Díaz	5.413.009-9

Tutor: Nicolás Escobar

Tabla de Contenidos

Agradecimientos **3**
Resumen **4**
1. Introducción **5**
1.1. Propósito 5
1.2. Contexto y Justificación del Proyecto 5
1.3. Objetivos Generales y Específicos 6
1.3.1. Objetivos Generales: 6
1.3.2. Objetivos Específicos: 6
1.4. Alcance del Proyecto 7
1.5. Resultados Esperados 7
1.6. Público Objetivo 7
2. Marco Teórico **8**
2.1. Conceptos Fundamentales Relacionados con el Proyecto 8
2.1.1. Fundamentos Teóricos: 8
2.2. Tecnologías y Herramientas Utilizadas 10
2.2.1. Plataformas y Lenguajes de Programación: 10
2.2.2. Frameworks y Librerías: 10
2.2.3. Herramientas de Desarrollo y Colaboración: 10
2.2.3. Herramientas de Pruebas y Testing: 11
3. Descripción del Proyecto **13**
3.1. Detalles Técnicos del Proyecto 13
3.1.1. Visión General 13
3.1.2. Requisitos del Sistema: 13
3.2. Arquitectura del Sistema 14
3.2.1. Diseño de Alto Nivel: 14
3.2.2. Modelo de dominio: 15
3.2.3. Componentes del Sistema: 16
3.2.4. Tecnologías y Frameworks: 17
3.3 Descripción de Funcionalidades 18
3.3.1 Funcionalidades: 18
3.3.2. Flujos de Usuario: 19
4. Implementación **20**
4.1. Metodologías de Trabajo 20
4.1.1. Selección de Metodología: 20
4.1.2. Principios y Prácticas: 20
4.2. Procesos y Fases de Desarrollo 21
4.2.1. Fases del Proyecto: 21

4.2.2. Gestión de Tareas y Priorización:.....22
4.3. Herramientas de Gestión y Desarrollo Utilizadas.....26
4.3.1. Herramientas de Desarrollo y Colaboración:..... 26
4.3.2. Herramientas de Comunicación:.....26
4.3. Interfaces de Usuario.....27
5. Pruebas Realizadas y Sus Resultados.....32
5.1. Pruebas de Front-end e Integración.....32
5.1.1. Plan de Testing Inicial con Cypress..... 32
5.1.2. Transición al Testing Exploratorio.....32
5.1.3. Puesta en marcha de Testing Exploratorio.....32
5.1.4. Impacto del Testing Exploratorio.....33
5.2. Pruebas en backend.....34
5.2.1. Pruebas Unitarias con Jest.....34
5.2.2. Pruebas de Integración con Supertest.....34
5.2.3. Pruebas de Integración Continua (CI) con Husky.....34
5.2.4. Impacto del Testing en Backend.....35
6. Conclusiones y Trabajo Futuro..... 37
6.1. Evaluación General del Proyecto..... 37
6.1.1. Resumen de Logros.....37
6.1.2. Evaluación del Proceso.....37
6.2. Aprendizajes y Desafíos Enfrentados.....38
6.2.1. Lecciones Aprendidas:..... 38
6.2.2. Desafíos y Obstáculos:.....38
6.3. Propuestas para Trabajos Futuros o Mejoras..... 38
6.3.1. Áreas de Mejora:..... 38
6.3.2. Ideas para Trabajos Futuros:..... 39
7. Referencias.....40

Agradecimientos

Al llegar a esta etapa final, reflexionamos sobre el camino que emprendimos hace unos años, un viaje que nunca recorrimos solos, sino acompañados por personas que se convirtieron en pilares fundamentales en nuestro progreso. Este espacio lo dedicamos para agradecerles.

Primero, queremos agradecer a nuestras familias, cuyo apoyo incondicional y motivación constante han sido esenciales a lo largo de nuestra carrera.

Un agradecimiento especial a todos los docentes que formaron parte de esta aventura educativa, compartiendo su conocimiento y experiencia con un compromiso admirable.

Nuestra gratitud se extiende al Ing. Cristian Bauza, coordinador de la carrera, por su disponibilidad constante y diligencia en resolver cualquier inquietud administrativa. Asimismo, queremos expresar un agradecimiento especial a todo el personal de la sede de UTEC San José — administrativos, técnicos, seguridad y limpieza — que han trabajado y continúan trabajando, creando un ambiente acogedor y haciéndonos sentir "como en casa" cada vez que visitamos la sede.

Un reconocimiento especial para nuestro tutor Nicolás Escobar, por su orientación y apoyo constante en el desarrollo de este proyecto.

Finalmente, pero no menos importante, agradecemos a los miembros del tribunal por su tiempo y dedicación en la evaluación de nuestro trabajo.

Resumen

El proyecto "La Puerta Verde Software" representa una innovadora solución de digitalización enfocada en transformar la gestión de pedidos y operaciones del pub "La Puerta Verde" en San José de Mayo. Mediante la implementación de interfaces específicas para clientes, mozos y cajeros, el software facilita la toma de pedidos y la comunicación eficiente con la cocina. Se integra además un back office para gestión de inventario, la administración general y un dashboard que centraliza el control de caja, la gestión de órdenes, la ocupación de mesas, y el seguimiento de botellas de bebidas. Su objetivo es optimizar la experiencia de clientes y empleados, proporcionando un sistema eficiente y digitalizado que mejore la gestión y el registro de actividades del local. Este software promete revolucionar la operativa interna y la experiencia del cliente en el establecimiento.

1. Introducción

1.1. Propósito

El propósito de este informe es establecer un marco claro y comprensivo para el proyecto "La Puerta Verde Software", un sistema integral de gestión para el emblemático pub "La Puerta Verde" en San José de Mayo. Este proyecto no solo representa un salto hacia la modernización y eficiencia, sino que también es un reflejo de la evolución y adaptación del negocio a las nuevas demandas y tecnologías del sector gastronómico.

Desde su concepción, "La Puerta Verde" ha mantenido una operación tradicional, con pedidos tomados a lápiz y papel, un método que, si bien ha servido durante años, ahora se percibe como ineficiente y propenso a errores. La introducción de un menú digital fue un primer paso hacia la digitalización, pero esta solución parcial evidenció la necesidad de una automatización y personalización más robustas para optimizar las operaciones del local.

La evolución de las necesidades del pub, especialmente en lo que respecta a la toma de órdenes y el control del inventario, ha sido un factor crucial en la iniciativa de este proyecto. Los procesos actuales, aunque funcionales, no están alineados con las expectativas modernas de gestión eficiente y experiencia del cliente. El desarrollo de "La Puerta Verde Software" apunta a una revolución en la gestión diaria del local, buscando integrar completamente las operaciones en un sistema digitalizado y automatizado.

En un entorno de negocios donde la digitalización se ha convertido en una tendencia dominante, los dueños de "La Puerta Verde" han reconocido la importancia de adaptarse y liderar el cambio en su sector. Esta decisión de avanzar hacia la digitalización no solo es un reflejo de la adaptabilidad y visión del negocio, sino que también establece un nuevo estándar en la experiencia del cliente y la eficiencia operativa.

1.2. Contexto y Justificación del Proyecto

Como se ha mencionado, el proyecto "La Puerta Verde Software" surgió como una solución integral a varios desafíos enfrentados por el pub "La Puerta Verde". Entre estos desafíos se incluyen la reducción de los tiempos de espera de los clientes, la minimización de errores en los pedidos, y la mejora de la gestión del inventario. El método tradicional de toma de pedidos y gestión manual del inventario había resultado ineficiente, a

menudo conduciendo a errores y retrasos que afectan la experiencia del cliente y la operatividad del local.

Además, existía la necesidad de un sistema que permitiera a los dueños del pub dejar el negocio en manos de sus empleados con la confianza de que todas las operaciones serían registradas de manera precisa. La capacidad de registrar el ingreso de stock en tiempo real, mientras se atiende al proveedor, también se identificó como un aspecto crítico para la eficiencia operativa y la precisión del inventario.

1.3. Objetivos Generales y Específicos

1.3.1. Objetivos Generales:

1. Digitalizar y automatizar los procesos de toma de pedidos y gestión del inventario.
2. Mejorar la eficiencia operativa y la experiencia del cliente en "La Puerta Verde".
3. Implementar un sistema confiable y completo para la gestión diaria del pub.

1.3.2. Objetivos Específicos:

1. Reducir significativamente los tiempos de procesamiento de pedidos.
2. Aumentar la precisión en la gestión del inventario y el registro de transacciones.
3. Facilitar la delegación de responsabilidades a los empleados con un sistema de seguimiento y registro confiable.
4. Proporcionar un registro detallado de las ventas y consumos, especialmente para clientes preferenciales, incluyendo la gestión de créditos y promociones basadas en la fidelidad.

Estos objetivos reflejan no solo la necesidad de modernización tecnológica en "La Puerta Verde", sino también el compromiso del pub con una operación eficiente y una experiencia de cliente excepcional. Por esto "La Puerta Verde Software" está diseñado para ser una solución completa que aborde estas necesidades y establezca un nuevo estándar en la gestión de restaurantes y pubs.

1.4. Alcance del Proyecto

El sistema incluye:

1. Gestión de pedidos y comunicación con cocina.
2. Dashboard para control de operaciones.
3. Back office para administración.
4. Estadísticas y análisis de datos.
5. Gestión de inventario y registro de clientes preferenciales.
6. Funcionalidades de apertura y cierre de mesas y botellas.

1.5. Resultados Esperados

Se espera que "La Puerta Verde Software" proporcione un sistema eficiente para la gestión de pedidos y operaciones, mejorando significativamente la experiencia de clientes y empleados, y proporcionando un registro digitalizado de todas las actividades del local.

1.6. Público Objetivo

Este software está destinado para uso interno por los empleados de "La Puerta Verde", así como para mejorar la experiencia de los clientes del pub.

2. Marco Teórico

2.1. Conceptos Fundamentales Relacionados con el Proyecto

2.1.1. Fundamentos Teóricos:

Arquitectura de Software en Capas: Esta estructura divide el sistema en capas funcionales independientes - presentación, negocio y acceso a datos - facilitando su escalabilidad y mantenimiento. Es un patrón de arquitectura software utilizado en la mayoría de los sistemas, permitiendo un desarrollo organizado y distribuido. (Blancarte, n.d.), (\/, 2020) (Durán, 2023)

Programación en JavaScript y TypeScript: JavaScript y TypeScript son elegidos por su eficacia en el desarrollo tanto del back-end como del front-end. TypeScript, siendo un superconjunto de JavaScript, ofrece una mejor organización del código y facilita la programación orientada a objetos, mientras que JavaScript es ampliamente utilizado por su flexibilidad y compatibilidad. (Documentation, n.d.) (Javascript, 2023)(Novoseltseva, 2020), (Fierro, n.d.)

Bases de Datos Relacionales (MySQL): MySQL, una base de datos relacional, destaca por su sencillez en el uso y aprendizaje, gracias a su sintaxis accesible y amplia documentación, facilitando su adopción por desarrolladores de todos los niveles. Su compatibilidad con múltiples plataformas, incluyendo Windows, Linux y macOS, lo convierte en una solución versátil para diferentes entornos de desarrollo. Resalta en su capacidad para escalar eficientemente y manejar grandes volúmenes de datos sin sacrificar rendimiento, debido a su arquitectura optimizada y uso de múltiples hilos de ejecución. Además, MySQL se beneficia de una extensa comunidad global de usuarios y desarrolladores, ofreciendo un soporte y recursos considerables a través de foros, grupos de discusión y documentación en línea. Estas características hacen de MySQL una herramienta confiable y potente para el almacenamiento y gestión de datos en aplicaciones web, adecuada para proyectos como "La Puerta Verde". (Salmerón, 2023) (Londoño, 2023)

REST y Websockets: REST, conocido por su protocolo de escalabilidad, se destaca por la separación entre cliente y servidor, permitiendo escalar el producto con relativa facilidad. Su flexibilidad y portabilidad facilitan la migración de un servidor a otro y cambios en la base de datos, posibilitando alojar el front y el back en servidores diferentes, lo que aporta una ventaja significativa en el manejo. La independencia entre cliente y

servidor permite desarrollos independientes en diferentes partes de un proyecto, adaptándose a diversos entornos y sintaxis. Además, las API REST suelen tener una documentación clara y accesible, facilitando su comprensión y uso.

Websockets, por otro lado, ofrece comunicación bidireccional en tiempo real, permitiendo que tanto el cliente como el servidor envíen y reciban mensajes instantáneamente sin esperar solicitudes previas. Su eficiencia se manifiesta en la conexión persistente, reduciendo la sobrecarga de establecer y cerrar conexiones repetidas. Además, los Websockets emplean puertos estándar HTTP y HTTPS, lo que facilita su uso a través de firewalls y proxies sin restricciones, y se pueden utilizar en una variedad de aplicaciones cliente-servidor más allá de las aplicaciones web tradicionales.

En conjunto, REST y Websockets ofrecen un conjunto de características complementarias que son cruciales para el desarrollo de aplicaciones web modernas. REST proporciona una base sólida para aplicaciones escalables y flexibles, mientras que Websockets amplían las capacidades con comunicaciones en tiempo real y eficientes, adecuadas para aplicaciones que requieren interacciones instantáneas y continuas entre cliente y servidor. (Montes, n.d.), (Rodríguez, 2023)

Scrum: Scrum es un marco de trabajo ágil para la gestión de proyectos que ayuda a los equipos a estructurar y administrar su trabajo mediante un conjunto de valores, principios y prácticas. Inspirado en un equipo de rugby (de donde proviene su nombre) que se entrena para un gran juego, Scrum fomenta que los equipos aprendan a través de la experiencia, se autoorganicen mientras trabajan en un problema y reflexionen sobre sus éxitos y fracasos para mejorar continuamente. (Drumond, n.d.)

Diagrama de Gantt: Un diagrama de Gantt es una herramienta de gestión de proyectos que asiste en la planificación y programación de proyectos de todos los tamaños; son particularmente útiles para visualizar proyectos. Se define como una representación gráfica de la actividad contra el tiempo; ayuda a los profesionales del proyecto a monitorear el progreso. (Naybour, n.d.)

Relación entre Teoría y Proyecto:
Estos fundamentos teóricos se aplican directamente en el proyecto para abordar eficientemente los retos de la gestión de un pub. La arquitectura en capas proporciona una base sólida y flexible, mientras que la combinación de tecnologías modernas asegura un desarrollo ágil y una experiencia de usuario optimizada.

2.2. Tecnologías y Herramientas Utilizadas

2.2.1. Plataformas y Lenguajes de Programación:

JavaScript y TypeScript: Elegidos por su flexibilidad, popularidad y capacidades específicas. JavaScript se usa en el back-end por su naturaleza basada en eventos, ideal para operaciones asincrónicas en aplicaciones en tiempo real. TypeScript en el front-end proporciona tipado estático y una estructura de código más robusta, mejorando la legibilidad y mantenibilidad. (Documentation, n.d.) (Javascript, 2023)(Novoseltseva, 2020), (Fierro, n.d.)

Node.js y Sequelize: Node.js se selecciona para el back-end debido a su naturaleza no bloqueante y orientada a eventos, crucial para manejar múltiples solicitudes en el ambiente dinámico de un pub. Sequelize, como ORM, se elige por su eficiencia en la gestión de bases de datos relacionales, facilitando operaciones con una sintaxis amigable y proporcionando seguridad adicional. (nodeJS, n.d.) (Sequelize, n.d.)(Blanch, 2017)

2.2.2. Frameworks y Librerías:

Angular y Express.js: Angular se utiliza para el front-end por su estructura MVC, permitiendo desarrollar una interfaz de usuario rica y reactiva. Express.js en el back-end se elige por su simplicidad y flexibilidad, adecuado para construir una API RESTful escalable y manejar solicitudes en tiempo real. (Angular, n.d.) (ExpressJS, n.d.) (Jesús, 2023) (Martínez, 2021)

Razones de Elección sobre Otras Opciones:

La madurez, la extensa comunidad y el soporte, junto con su popularidad, fueron factores clave en la elección de estas tecnologías. Aseguran una mayor disponibilidad de recursos de aprendizaje y soporte comunitario, vital para abordar desafíos técnicos durante el desarrollo.
Esta selección de tecnologías y herramientas está estratégicamente alineada con los objetivos de eficiencia, escalabilidad y robustez del proyecto "La Puerta Verde".

2.2.3. Herramientas de Desarrollo y Colaboración:

IDEs y Entornos de Desarrollo:

Visual Studio Code: Utilizado por su versatilidad y las extensiones que facilitan el desarrollo, como Console Ninja para manejar los console.log en

el propio IDE y GitHub Copilot para sugerencias de código en tiempo real, mejorando la eficiencia y precisión en la escritura del código. (VSCode, n.d.) (ConsoleNinja, n.d.) (Github Copilot, n.d.)

Herramientas de Control de Versiones:

GitHub: Esencial para la colaboración y el seguimiento de cambios en el código. Facilita la gestión de versiones, revisiones de código y la colaboración entre los miembros del equipo. (Github, n.d.)

Herramientas de Colaboración y Comunicación:

Trello, Discord y Google Docs: Estas herramientas son fundamentales para la coordinación y el flujo de trabajo del equipo, permitiendo una comunicación efectiva, gestión de tareas y documentación compartida. (Trello, n.d.) (Discord, n.d.) (Docs, n.d.)

Figma y Draw.io: Utilizados para el diseño de interfaces y la creación de diagramas, respectivamente, facilitando la visualización y planificación del proyecto. (Figma, n.d.) (DrawIo, n.d.)

Chat GPT: Ofrece asesoramiento y apoyo en diversas áreas del desarrollo del proyecto. (ChatGPT, n.d.)

2.2.3. Herramientas de Pruebas y Testing:

Jest: Es un marco de pruebas para JavaScript que se enfoca en la simplicidad. Diseñado para garantizar la corrección de cualquier código JavaScript, Jest permite escribir pruebas con una API rica en funciones, accesible y fácil de usar, proporcionando resultados rápidos. Jest se integra sin configuraciones en la mayoría de los proyectos JavaScript y es ampliamente documentado y mantenido. (JestJS, n.d.)

SuperTest: Es una biblioteca impulsada por super-agent para probar servidores HTTP en Node.js, utilizando una API fluida. La motivación detrás de SuperTest es proporcionar una abstracción de alto nivel para pruebas HTTP, manteniendo la capacidad de acceder a la API de bajo nivel proporcionada por superagent. Una vez instalado, se puede referenciar simplemente llamando a require('supertest');. SuperTest es compatible con cualquier marco de pruebas, lo que lo hace versátil para diferentes entornos y necesidades de prueba. (SuperTest, 2023)

Husky: Es una herramienta moderna que facilita el uso de ganchos de Git nativos. Mejora los commits y ofrece la posibilidad de ejecutar linters para mensajes de commit, pruebas y linting de código en acciones de commit o push. Husky es ligero, sin dependencias externas, y está potenciado por una nueva característica de Git (core.hooksPath). Es compatible con macOS, Linux, Windows, interfaces gráficas de Git, directorios de hooks personalizados, proyectos anidados y monorepos. Además, sigue las mejores prácticas de npm y Yarn con respecto a la instalación automática y ofrece mensajes amigables al usuario. (Husky, n.d.)

3. Descripción del Proyecto

3.1. Detalles Técnicos del Proyecto

3.1.1. Visión General

El sistema ha sido concebido como una solución integral para transformar la operativa y la experiencia en el pub. Este sistema está diseñado para abordar tanto las necesidades funcionales como las no funcionales del negocio, incorporando características innovadoras y centradas en el usuario. Los elementos funcionales clave incluyen la gestión de usuarios, inventario, clientes, menús, así como la administración de órdenes y mesas. El sistema también se enfoca en aspectos no funcionales vitales como la respuesta rápida del sistema, la seguridad de los datos y un diseño que se adapta tanto a dispositivos móviles como a PCs. Estas características están alineadas para asegurar una gestión eficiente y moderna del pub, permitiendo una interacción fluida y segura tanto para el personal como para los clientes. Para una comprensión profunda de estos requisitos y su implementación en el desarrollo del proyecto, se sugiere revisar el anexo "Requerimientos del Sistema", que ofrece detalles exhaustivos de cada aspecto.

3.1.2. Requisitos del Sistema:

El sistema ha sido diseñado con una serie de requisitos funcionales y no funcionales cuidadosamente detallados para garantizar una gestión eficiente y moderna del pub. Entre los requisitos funcionales, se incluyen funciones como el registro e inicio de sesión de usuarios, gestión LABM (Listar, Agregar, Borrar, Modificar) de inventario, clientes, empleados y menús, así como la administración de órdenes y mesas. En cuanto a los no funcionales, se destacan la necesidad de respuesta rápida del sistema, seguridad de datos y un diseño adaptable a dispositivos móviles y PCs. Además, se han establecido requisitos específicos de interfaz y rendimiento, adaptados a las necesidades del pub. Para una comprensión detallada de estos requisitos y su implementación en el desarrollo del proyecto, se recomienda consultar el anexo adjunto, "Requerimientos del Sistema", donde se profundiza en cada aspecto de forma exhaustiva.

3.2. Arquitectura del Sistema

3.2.1. Diseño de Alto Nivel:

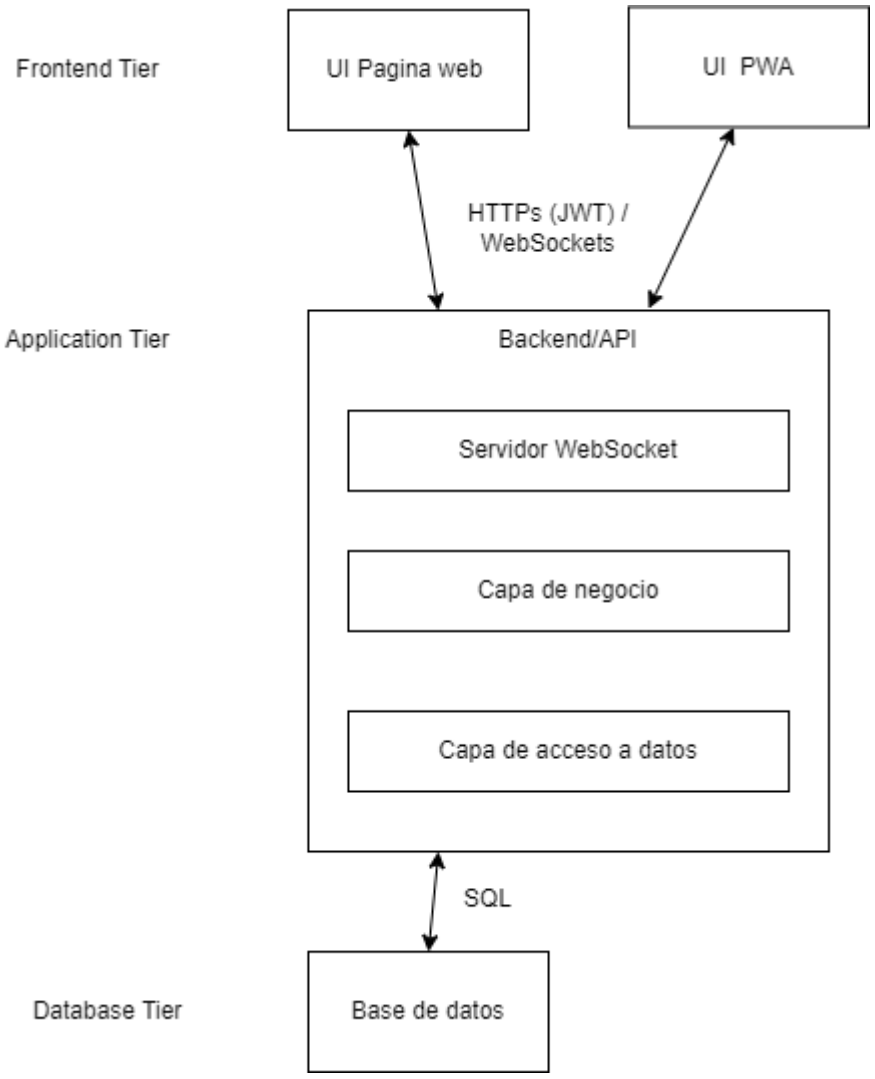


Figura 1. Diseño de arquitectura de alto nivel.

Este diseño ha sido seleccionado por su modularidad y facilidad para separar responsabilidades, facilitando el mantenimiento y escalabilidad.
Aquí solo se presenta una abstracción de alto nivel de la arquitectura.

3.2.2. Modelo de dominio:

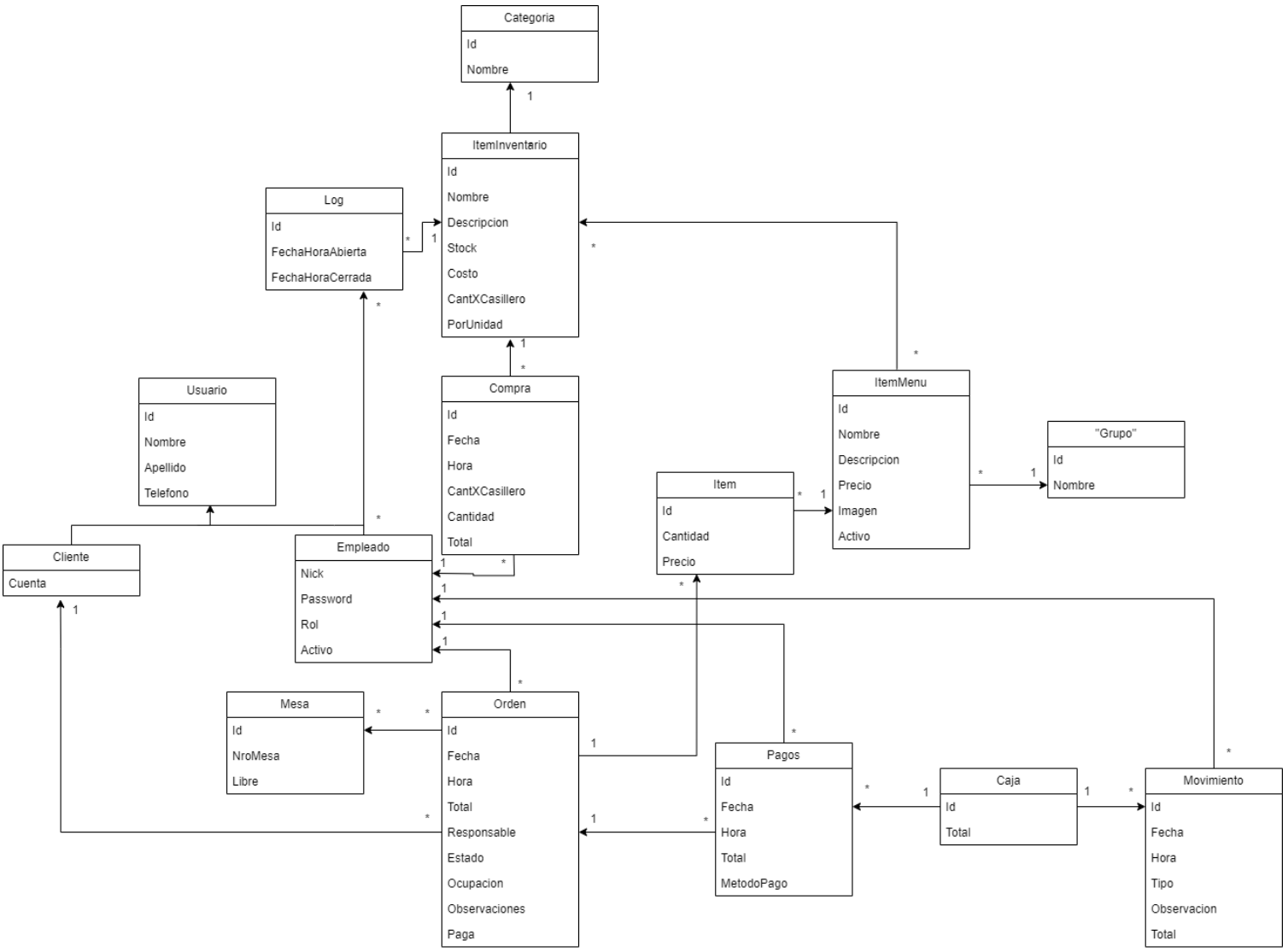


Figura 2. Modelo de dominio

En este modelo de dominio, se identifican los conceptos significativos del mundo real (dominio del problema) y se ilustran sus relaciones. Esto es fundamental para entender el sistema desde una perspectiva de alto nivel y para discutir el diseño con las partes interesadas no técnicas.

3.2.3. Componentes del Sistema:

El sistema se ha concebido con una arquitectura claramente definida, que divide sus operaciones en componentes específicos para maximizar la eficiencia y la escalabilidad. El Frontend se encarga de toda la interacción del usuario, asegurando una experiencia intuitiva y reactiva, mientras que el Backend sostiene la lógica empresarial, manteniendo la seguridad y la integridad de los datos. La Base de Datos es el núcleo de almacenamiento, diseñada para transacciones rápidas y seguras. Cada componente ha sido diseñado para trabajar en conjunto de manera armónica, con el frontend y backend en constante comunicación a través de una API REST y Websockets para actualizaciones instantáneas. Para una exploración en profundidad de cada componente, su función y cómo se complementan entre sí.

3.2.3.1. Diagrama de despliegue:

Este diagrama de despliegue (Figura 3) detalla una arquitectura de aplicación que integra servicios en la nube y tecnologías de frontend y backend. En la interfaz de usuario, se utiliza una Aplicación de Página Única (SPA) construida con Angular y hospedada en Vercel. Los dispositivos inician una "Static File Request" hacia Vercel para cargar la SPA en el navegador. Este proceso se realiza a través del protocolo HTTPS, asegurando que la transferencia de archivos estáticos sea segura y eficiente.

Una vez que la SPA está cargada, los dispositivos, ya sea un smartphone con una aplicación web progresiva (PWA) o un PC a través de un navegador web, interactúan directamente con el backend. Hacen solicitudes directas a la API a través de HTTPS, que es manejada por servicios en Google Cloud. La base de datos está gestionada por Google Cloud SQL con MySQL, y la lógica del servidor se ejecuta en Google Cloud Run utilizando Express.js para exponer la API REST.

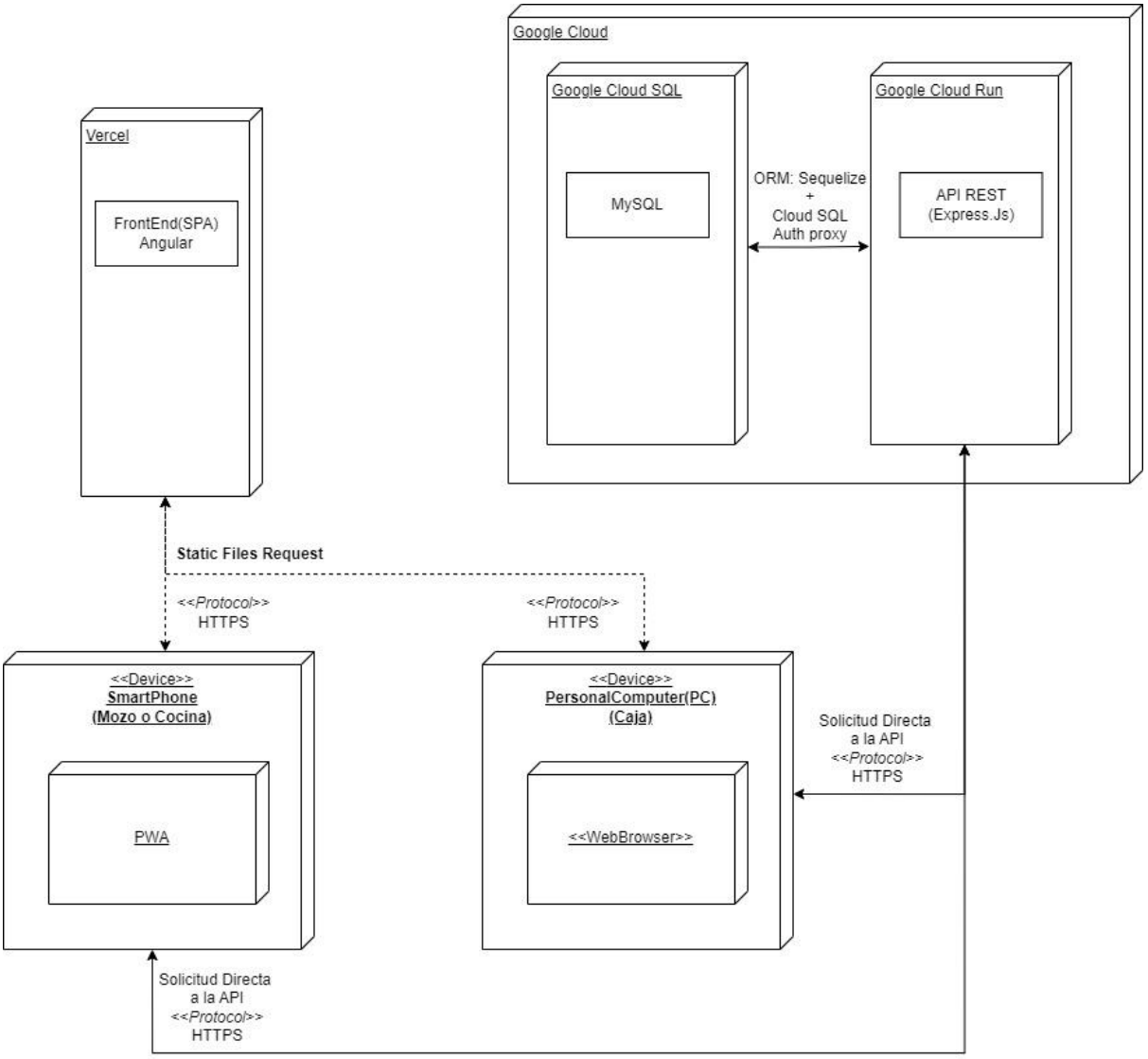


Figura 3. Diagrama de Despliegue

3.2.4. Tecnologías y Frameworks:

El sistema ha sido implementado utilizando una combinación de tecnologías y frameworks de vanguardia para asegurar un desempeño robusto y una experiencia de usuario excepcional. En el corazón del frontend, Angular en su versión 16.2.12 proporciona una interfaz rica y dinámica, apoyada en las fortalezas de TypeScript para mejorar la escalabilidad y mantenibilidad del código. El backend es impulsado por Node.js y Express.js, creando un ecosistema eficiente para la lógica del servidor y la gestión de las API.

El enfoque de aplicación de página única (SPA) y la implementación de características de aplicación web progresiva (PWA) llevan la interactividad y la accesibilidad a un nuevo nivel, ofreciendo una experiencia similar a una aplicación nativa en la web. Docker se integra en el flujo de trabajo para encapsular y desplegar componentes de manera consistente, mientras que MySQL establece el sistema de gestión de bases de datos con Sequelize como el ORM elegido, simplificando la interacción con la base de datos y asegurando la integridad de las operaciones de datos.

Estas tecnologías no solo fueron seleccionadas por su rendimiento y fiabilidad, sino también por su sinergia al trabajar juntas, creando un sistema cohesivo y bien integrado. Para obtener una descripción detallada de cómo estas tecnologías y frameworks se utilizan en la arquitectura de nuestro sistema y su contribución al cumplimiento de nuestros objetivos.

3.3 Descripción de Funcionalidades

3.3.1 Funcionalidades:

A continuación se listan las principales funcionalidades del sistema, para mayor detalle de cada una de ellas se puede consultar el documento anexo "Requerimientos del Sistema".

3.3.1.1 Menú para toma de órdenes por los usuarios:
Permitirá a los usuarios seleccionar y ordenar alimentos y bebidas.

3.3.1.2 Caja con Dashboard de Control General: Para visualizar y controlar diversas métricas y operaciones importantes en tiempo real.

3.3.1.3 Disponibilidad de mesas: Mostrará la disponibilidad actual de mesas en el restaurante.

3.3.1.4 Ventas Rápidas de Bebidas: Facilitará la venta rápida de bebidas al público desde la barra.

3.3.1.5 Toma de Órdenes por Parte de Mozos: Permitirá a los empleados tomar órdenes de forma eficiente.

3.3.1.6 Login y autenticación: Para acceso seguro al sistema.

3.3.1.7 Administración del Sistema: CRUD de usuarios, productos, etc.

3.3.1.8 Gestión de Inventario: Permitirá controlar el stock y demás aspectos relacionados al inventario.

3.3.1.9 Estadísticas: Reportes y análisis de datos relevantes.

3.3.2. Flujos de Usuario:

Se destacan 3 flujos de usuario muy importantes:

Solicitar una Orden: Este proceso comienza cuando un cliente o empleado accede al sistema para realizar un pedido. Incluye la selección de ítems del menú, la personalización del pedido según las preferencias del cliente y la confirmación final del pedido. Este flujo es esencial para garantizar una experiencia de usuario fluida y eficiente, permitiendo a los clientes y empleados interactuar de manera intuitiva con el menú y personalizar los pedidos a su gusto.

Entregar una Orden: Una vez que se ha solicitado y preparado un pedido, el siguiente paso es su entrega. Este proceso abarca desde la notificación al personal de cocina hasta la entrega efectiva del pedido al cliente. Este flujo asegura que los pedidos se manejen de manera oportuna y eficiente, minimizando los tiempos de espera y mejorando la satisfacción del cliente.

Cobrar una Orden: El último paso crucial en la interacción del usuario con el sistema es el proceso de facturación y cobro. Incluye la generación de la factura, la selección del método de pago y el procesamiento del pago. Este proceso es vital para cerrar el ciclo de servicio, asegurando una transacción segura y eficiente, y proporcionando al cliente todas las opciones de pago necesarias.

Los diagramas asociados con estos procesos pueden ser consultados en el documento anexo "Arquitectura de Software" en la sección "2.4.3. Vista de Procesos".

4. Implementación

Esta sección tratará de explicar la organización y gestión del desarrollo abordadas. Este enfoque ha sido crucial para la adaptación a las necesidades cambiantes y para garantizar un progreso eficiente y efectivo.

4.1. Metodologías de Trabajo

4.1.1. Selección de Metodología:

Para este proyecto, se ha adoptado una metodología ágil personalizada, que comparte similitudes con Scrum. Esta decisión se tomó debido a la naturaleza del proyecto y a las características del equipo, compuesto por estudiantes. De hecho, podría decirse que, por ya conocerse y haber trabajado durante la carrera en otros proyectos, el flujo de trabajo se ha dado de forma natural.

Esta metodología se caracteriza por:

- Ciclos de desarrollo cortos (sprints de una semana), permitiendo así una revisión y adaptación rápida.
- Planificación colaborativa al inicio de cada sprint, considerando el avance individual y general del proyecto.
- Un enfoque de equipo autogestionado, donde cada miembro aporta en todas las fases del proyecto.

La selección de esta metodología se basó en la necesidad de adaptabilidad, eficiencia en la comunicación y colaboración continua, alineándose con las limitaciones de tiempo y los recursos disponibles del equipo.

4.1.2. Principios y Prácticas:

Los principios clave de esta metodología incluyen:

- Iteraciones Cortas y Adaptabilidad: Se adoptaron sprints de una semana, lo que permitió agilidad y respuesta rápida a los cambios o retroalimentación.
- Colaboración y Comunicación Continua: Se mantuvo una comunicación constante y efectiva a través de herramientas como Discord, lo que fortaleció la colaboración del equipo.

- Revisión y Mejora Continua: Al final de cada sprint, se revisan logros y se discuten mejoras, lo cual promueve un ciclo de aprendizaje y adaptación.
- Enfoque Pragmático: Aunque se comenzó con una flexibilidad considerable en la fase de diseño, posteriormente se limitó la adaptabilidad para priorizar el cumplimiento de los requisitos definidos.

Las prácticas específicas adoptadas incluyen:

- Planificación de Sprints: Al inicio de cada sprint, se realiza una sesión de planificación para definir objetivos y tareas, basados en urgencia y dependencias.
- Reuniones de Revisión: Al final de cada sprint, se evalúa el progreso y se ajustan los objetivos para el siguiente ciclo.
- Relación con Stakeholders: Se mantuvo una interacción regular con el tutor, quien actuó como un guía, y con los clientes para alinear expectativas y recibir retroalimentación.

En resumen, en esta metodología personalizada se combinan elementos de Scrum con adaptaciones específicas para satisfacer las necesidades del equipo. Esto ha resultado en un enfoque equilibrado que prioriza la flexibilidad, la eficiencia y una fuerte colaboración, asegurando que cada paso del desarrollo esté alineado con los objetivos del proyecto.

4.2. Procesos y Fases de Desarrollo

4.2.1. Fases del Proyecto:

Fase 1: Planificación y Diseño (18/08/2023 al 17/09/2023)

Actividades Principales:

- Relevamiento de Requerimientos: Interacción constante con el cliente para negociar y definir el alcance del proyecto. Esta etapa enfatiza la importancia de ser sinceros y realistas sobre lo que se puede alcanzar.
- Desarrollo de Prototipo de Interfaz: Uso de Figma para crear un prototipo interactivo de la interfaz de usuario.
- Prototipos con Tecnologías Seleccionadas: Desarrollo de prototipos utilizando las tecnologías elegidas para el proyecto, con un enfoque en la mitigación de riesgos.

Entregables:

- Documento de Arquitectura.
- Documento de Estado del Arte.
- Documento de Requerimientos.
- Documento de Mitigación de Riesgos.

Fase 2: Desarrollo e Implementación (18/09/2023 al 19/11/2023)

Actividades Principales:

- Implementación de Requerimientos: Desarrollo de todas las funcionalidades requeridas, asegurando que sean funcionales y estén correctamente implementadas.
- Testing: Realización de pruebas para validar la funcionalidad y rendimiento de las implementaciones.

Fase 3: Despliegue y Documentación (19/11/2023 al 05/12/2023)

Actividades Principales:

- Implantación con el Cliente: Despliegue del software y asegurarse de que esté operativo y accesible para el cliente.
- Elaboración de Documentación:
  - Manual de Usuario: Creación de un manual detallado para guiar a los usuarios en el uso del software.
  - Manual de Instalación: Desarrollo de una guía para la instalación y configuración del software.
  - Informe final y documentos anexos.

4.2.2. Gestión de Tareas y Priorización:

4.2.2.1 Gestión Semanal y Evaluación de Avances:

Las prioridades del proyecto se han manejado semanalmente, ajustándose según el progreso realizado. Esta gestión dinámica ha permitido una adaptación continua a los desafíos y cambios del proyecto.
Cada semana se realiza una revisión de los objetivos alcanzados y se establecen los objetivos para la semana siguiente. Esta revisión ayuda a mantener el enfoque en las metas a corto plazo, garantizando un avance constante. La guía proporcionada por el tutor ha sido clave para asegurar que el equipo se mantenga en el camino correcto, ofreciendo retroalimentación valiosa para ajustar las prioridades y estrategias.

4.2.2.2 Diagrama de Gantt:

Con el objetivo de ilustrar eficazmente la distribución del tiempo y las fases del proyecto se puede observar el Diagrama de Gantt en la figura 4 y 5.

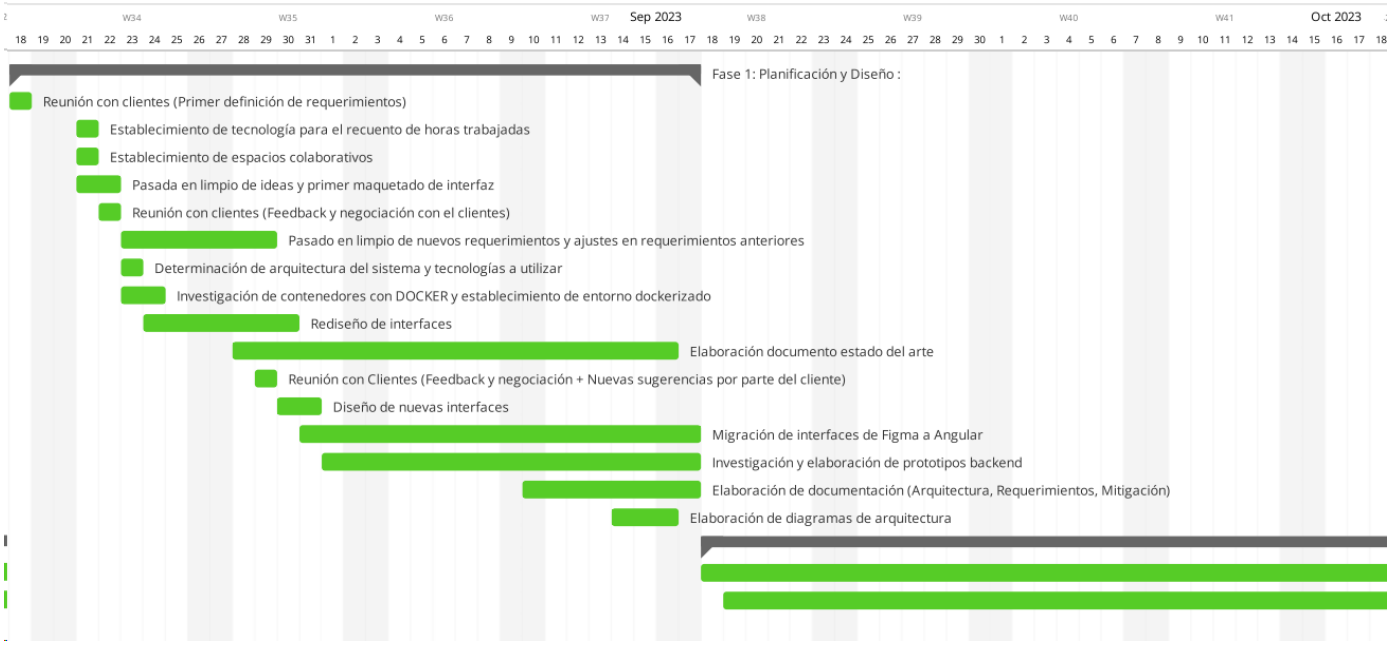


Figura 4. Diagrama de Gantt

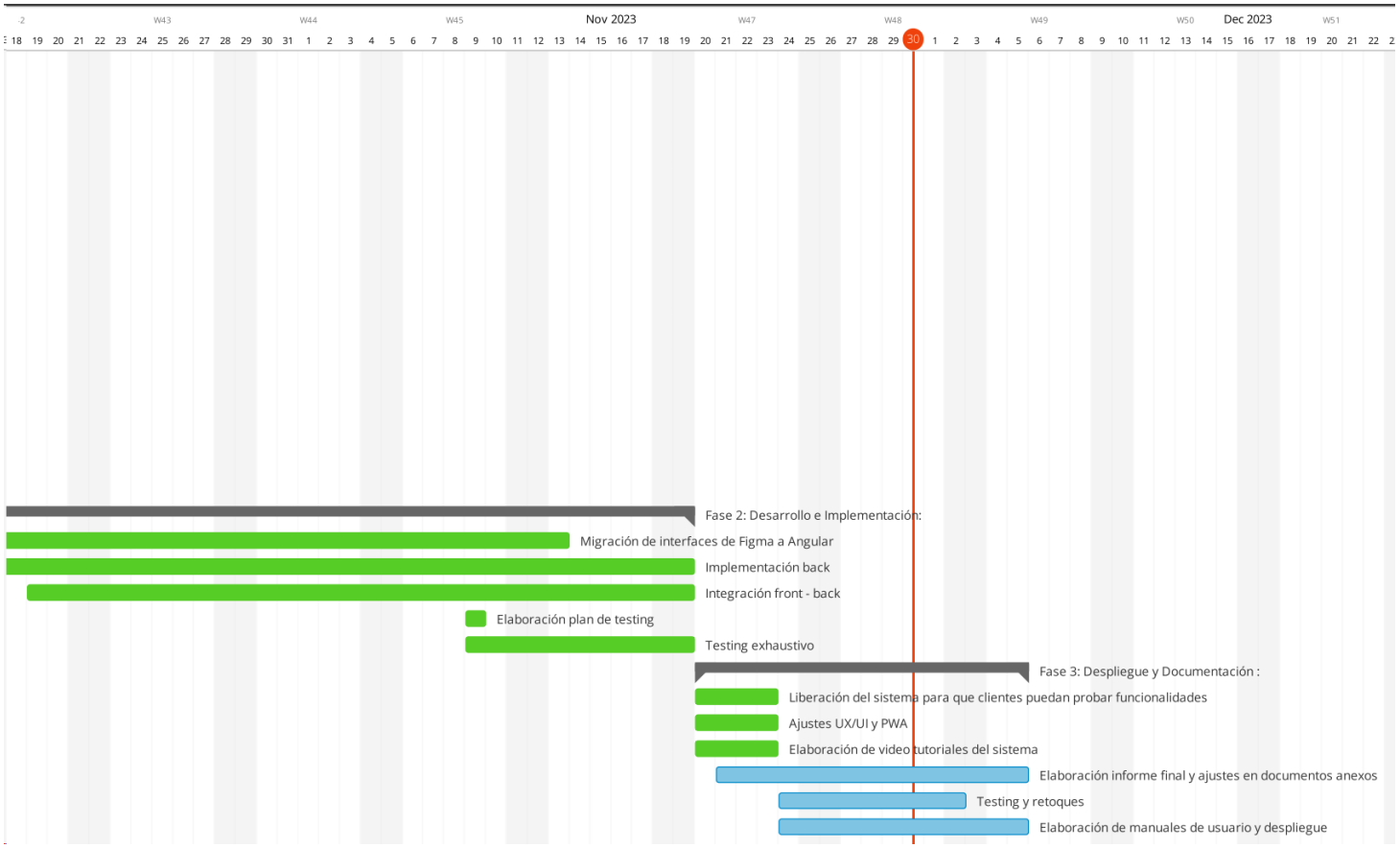


Figura 5. Diagrama de Gantt

4.2.2.3 División y Asignación de Tareas:

Las tareas se han dividido entre los miembros del equipo de manera equitativa, tomando en cuenta las habilidades y fortalezas de cada integrante. Se ha fomentado la práctica de especialización en ciertas áreas, mientras se promueve la colaboración y el apoyo mutuo para tareas complejas o desafiantes. A continuación se presentan tablas descriptivas relacionadas a la división y dedicación en las tareas a lo largo de las tres fases.

		Guzmán	Hernán	Mario
Fase 1	Definición de requerimientos	✔	✔	✔
	Maquetado interfaces	✔	✔	
	Elaboración diagramas arquitectura	✔	✔	✔
	Elaboración documentación	✔	✔	
	Setup Front Angular			✔
	Migración interfaces de Figma a Angular	✔		✔
	Investigación y setup docker		✔	✔
	Investigación y elaboración de prototipos backend		✔	
	Setup Back		✔	
Horas Fase 1		71	76	76

Tabla 1. División y Asignación de Tareas Fase 1

		Guzmán	Hernán	Mario
Fase 2	Migración de interfaces de Figma a Angular	✔		✔
	Implementación back		✔	
	Integración front - back	✔	✔	✔
	Elaboración plan de testing		✔	✔
	Testing exhaustivo	✔	✔	✔
Horas Fase 2		191	212	203

Tabla 2. División y Asignación de Tareas Fase 2

		Guzmán	Hernán	Mario
Fase 3	Liberación del sistema		✔	✔
	Ajustes UX/UI y PWA	✔		
	Elaboración de videotutoriales del sistema			✔
	Elaboración informe final y documentos anexos		✔	
	Testing y retoques	✔	✔	✔
	Elaboración de manuales de usuario y despliegue	✔	✔	
Horas Fase 3		52	61	43

Tabla 3. División y Asignación de Tareas Fase 3

Total horas dedicadas: 985 horas hombre.

	Guzmán	Hernán	Mario	Total
Fase 1	71	76	76	223
Fase 2	191	212	203	606
Fase 3	52	61	43	156
Total	**314**	**349**	**322**	**985**

Tabla 4. Resumen de horas dedicadas

4.3. Herramientas de Gestión y Desarrollo Utilizadas

4.3.1. Herramientas de Desarrollo y Colaboración:

Visual Studio Code (VSCode): Su uso ha sido indispensable para el desarrollo tanto del frontend como el backend. Además hemos aprovechado las extensiones como Console Ninja y GitHub Copilot para mejorar la eficiencia en la escritura y depuración del código. Este enfoque ha aumentado la productividad del equipo y ha permitido un desarrollo más ágil.
GitHub: Ha jugado un papel crucial en la colaboración del equipo, siendo utilizado para la gestión de versiones y revisiones de código. La capacidad de GitHub para rastrear y fusionar cambios en el código ha sido fundamental para mantener un desarrollo coherente y coordinado.
Chat GPT: Ha ofrecido asesoramiento y apoyo en varias áreas del desarrollo, proporcionando información relevante y sugerencias útiles.
Aunque ha sido una herramienta valiosa, el equipo ha sido consciente de sus limitaciones, especialmente en cuanto a la actualización de información y la interpretación de problemas complejos.

Estas herramientas han sido indispensables en el día a día del proyecto y para un desarrollo de calidad. La integración de estas en el flujo de trabajo ha permitido un seguimiento más eficaz de las tareas, ha facilitado una colaboración más fluida y efectiva entre los miembros del equipo así como también ha enriquecido el abordaje de algunas tareas.

4.3.2. Herramientas de Comunicación:

Trello, Discord, y Google Docs: Estas herramientas han sido esenciales para la gestión de tareas, la comunicación continua y el intercambio de documentación. Trello ha permitido un seguimiento claro del progreso de las tareas, mientras que Discord ha sido la plataforma principal para las discusiones diarias y las reuniones virtuales. Google Docs ha facilitado la colaboración en documentos y la consolidación de la documentación del proyecto.
Figma y Draw.io: Han sido utilizados para diseñar interfaces y crear diagramas respectivamente, lo que ha permitido una visualización clara de los objetivos del proyecto y ha ayudado en la planificación y el diseño.

La implementación de estas herramientas de comunicación ha contribuido significativamente a mejorar la coordinación del equipo. Han facilitado una comunicación efectiva y han permitido un intercambio de información ágil y organizado, lo cual ha sido vital para el éxito del proyecto.

4.3. Interfaces de Usuario

A continuación se presenta una selección de las principales interfaces.
Solicitar una orden (Menú cliente):

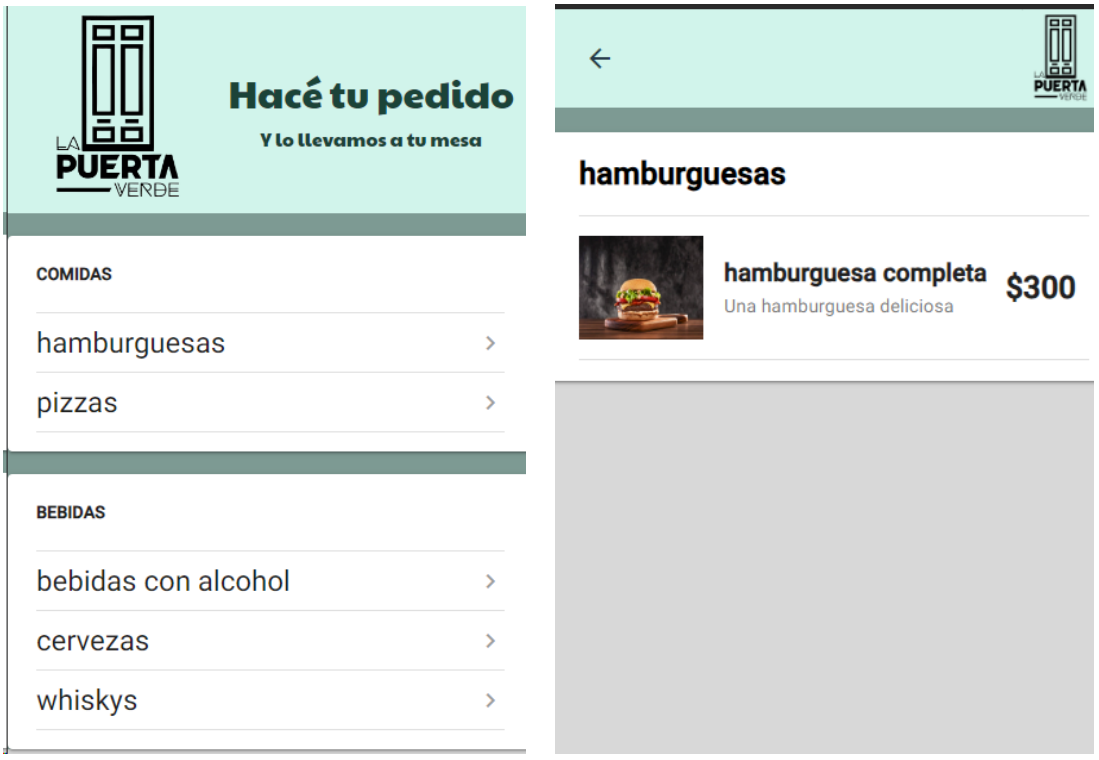


Figura 6 y 7. Menú cliente

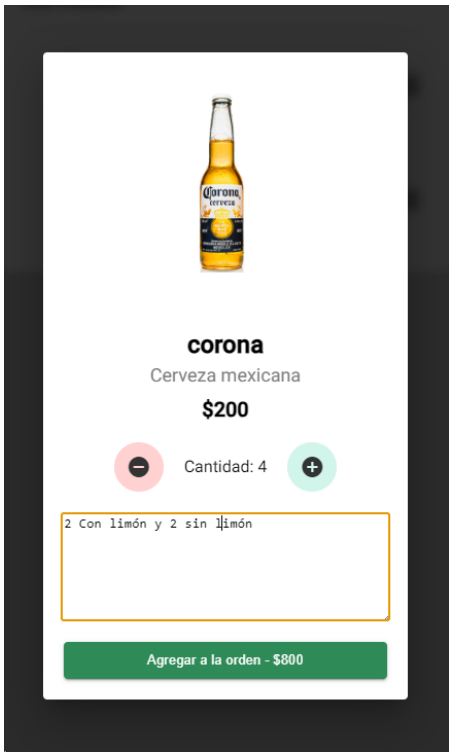


Figura 8 y 9. Agregar items a la orden.

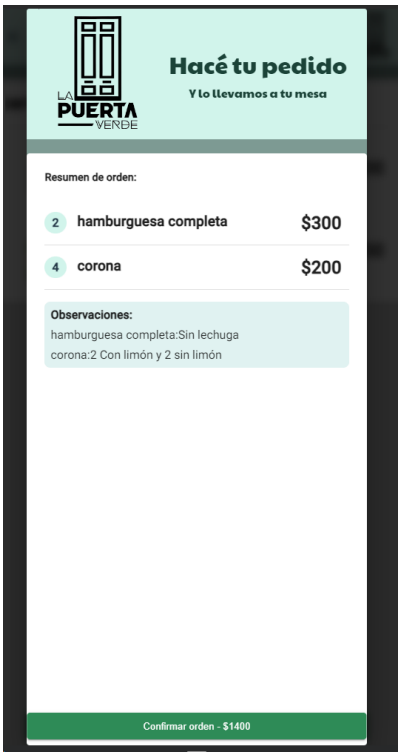


Figura 10. Resumen Orden

Figura 11. Agregar datos adicionales a la orden

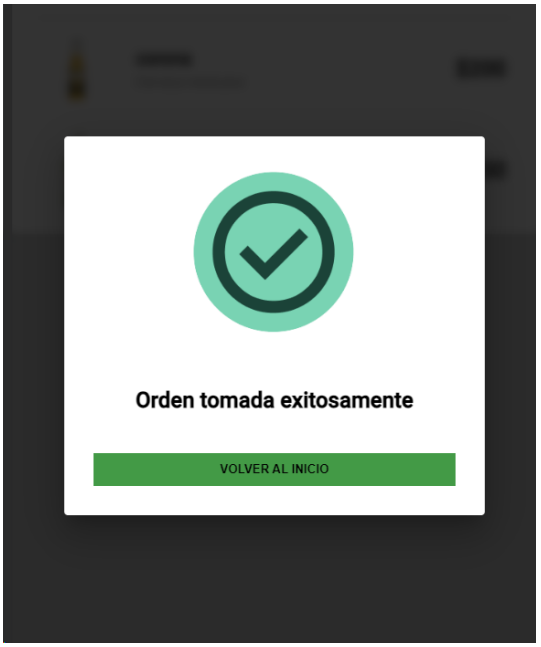


Figura 12. Feedback orden tomada

Dashboard Caja:

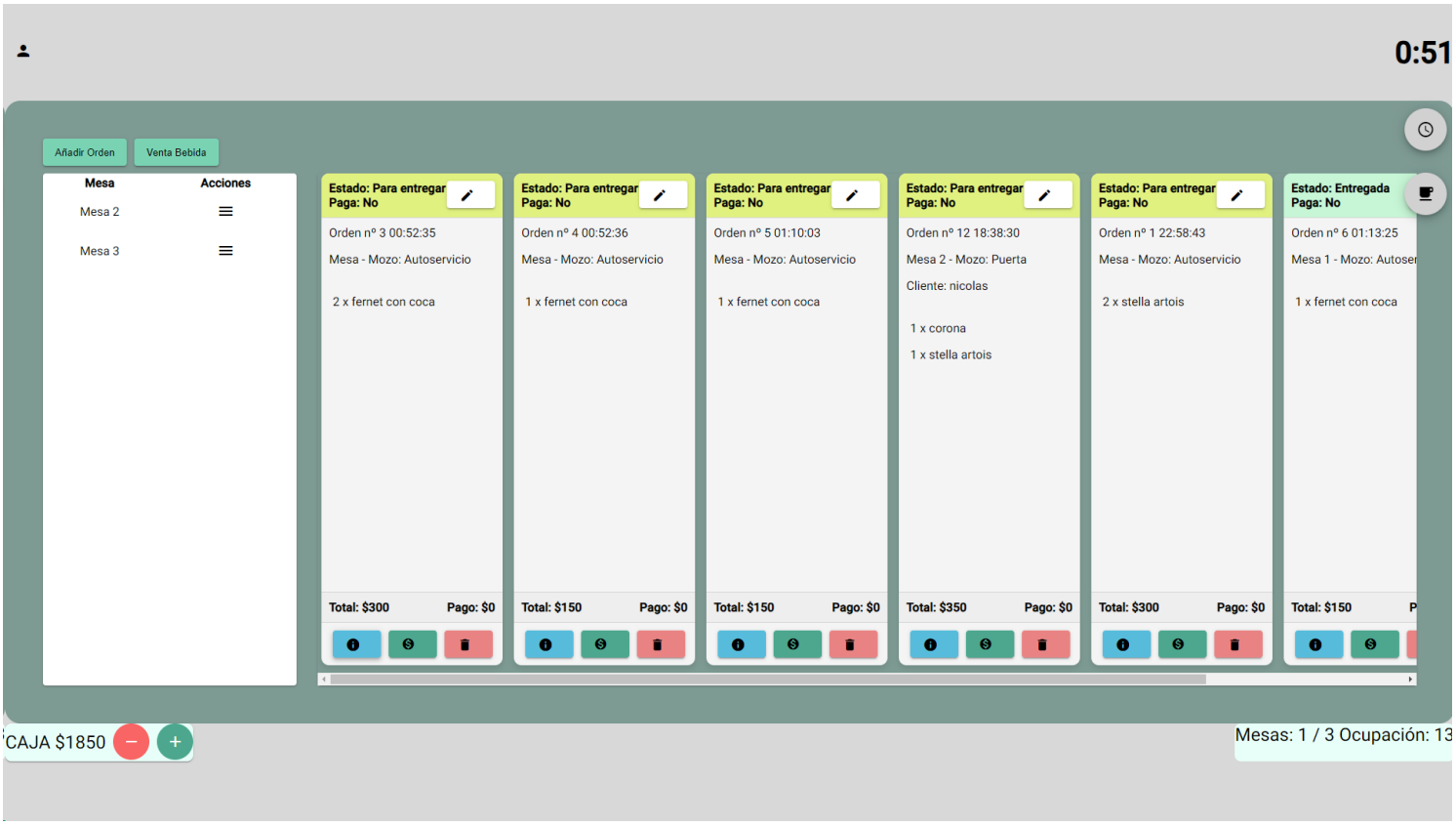
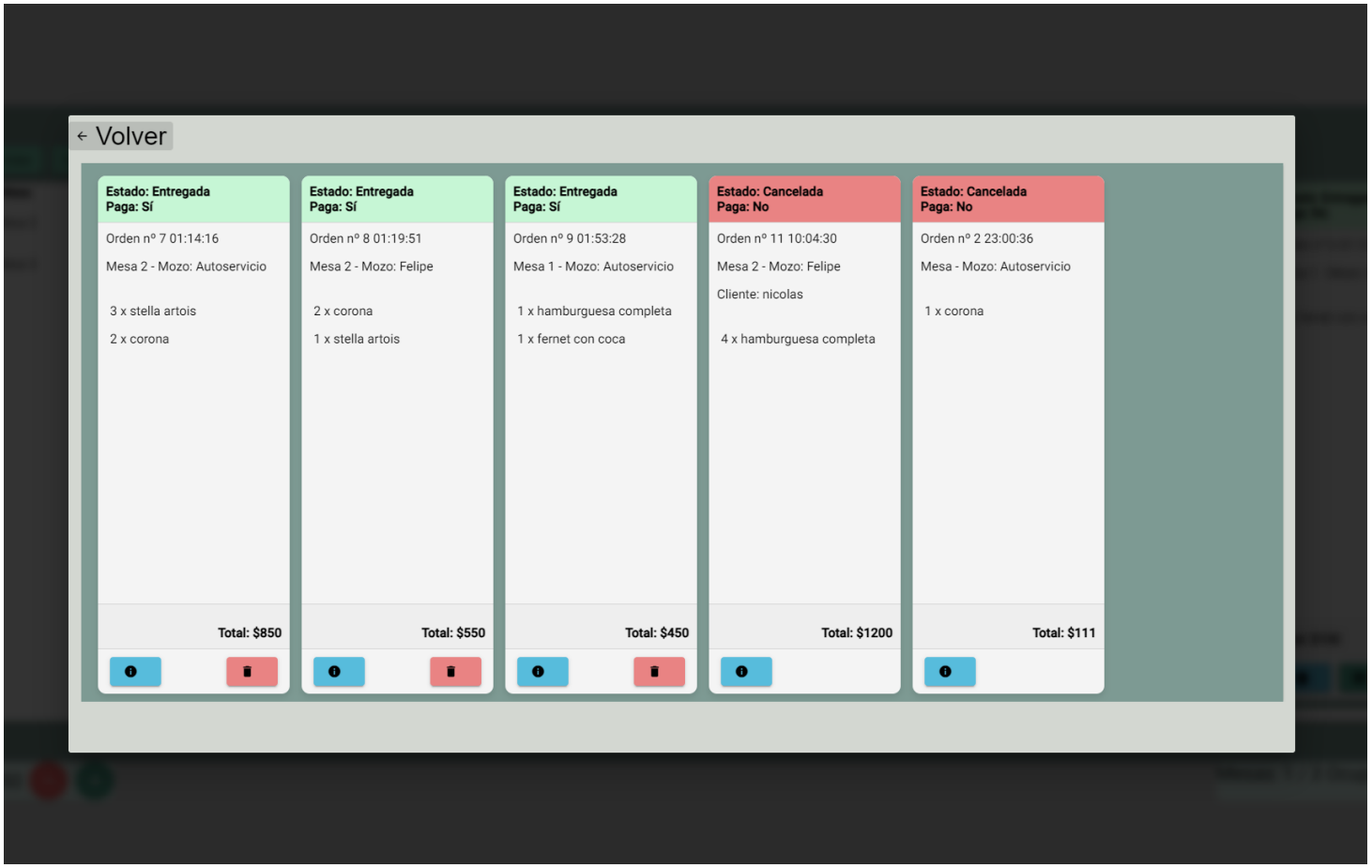


Figura 13. Home Caja

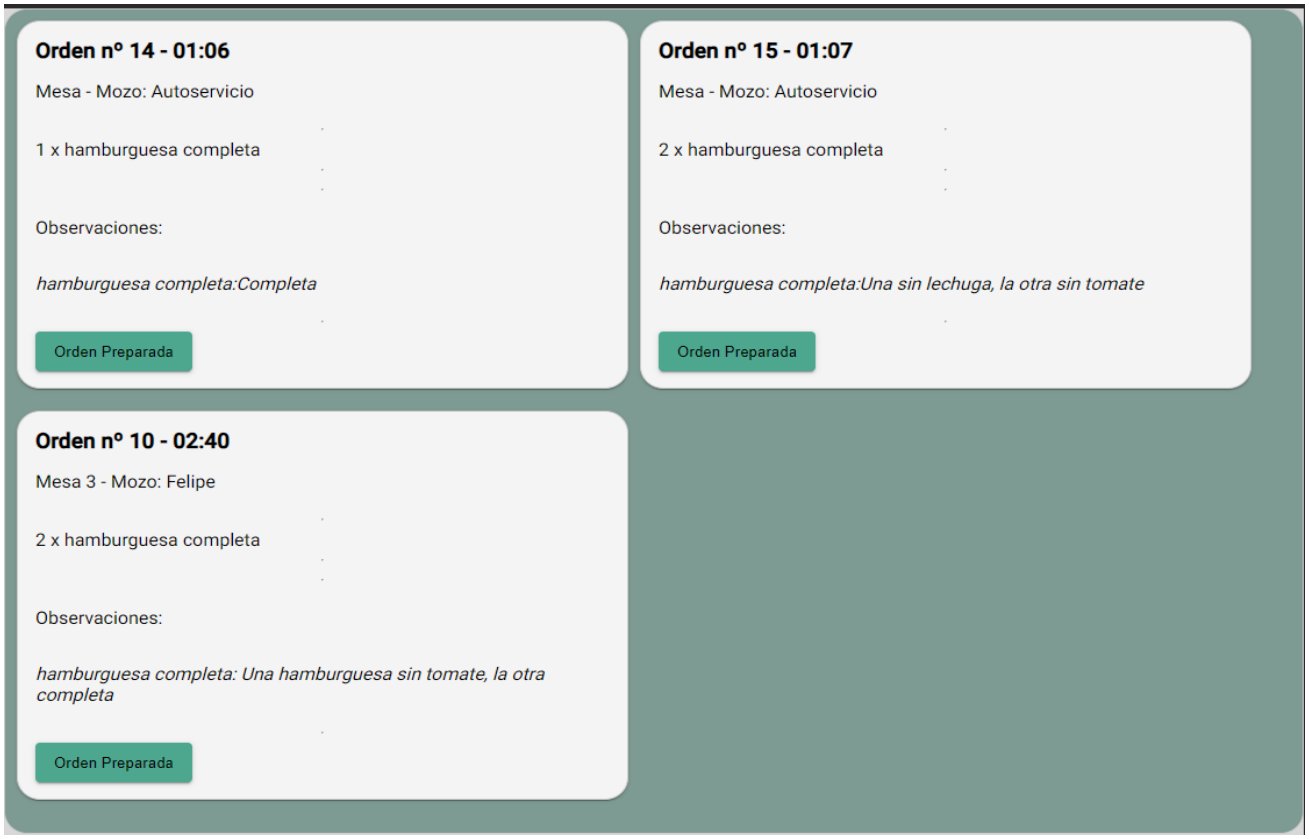
Panel Mesas (Dashboard Caja):

Figura 14. Panel Mesas

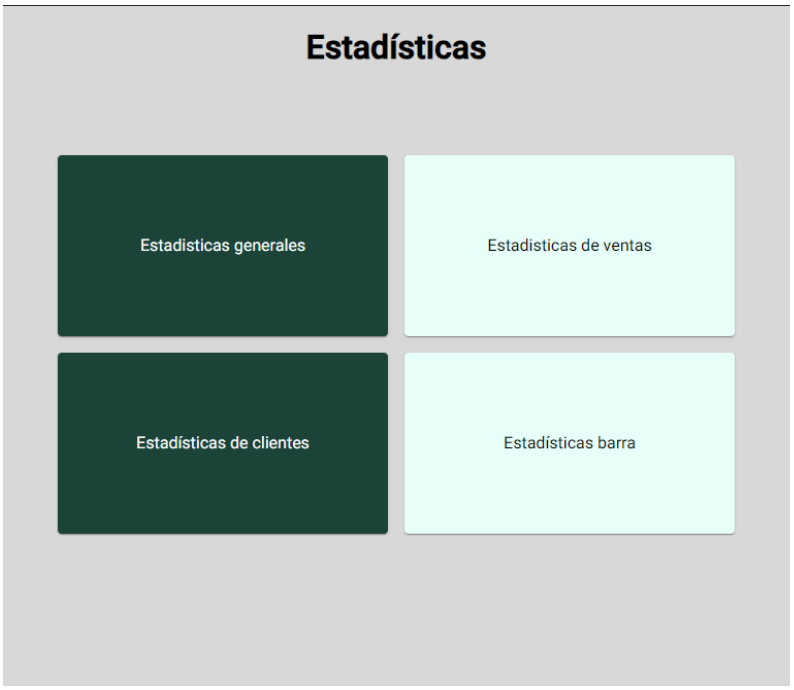
Historial de ordenes (Dashboard Caja):

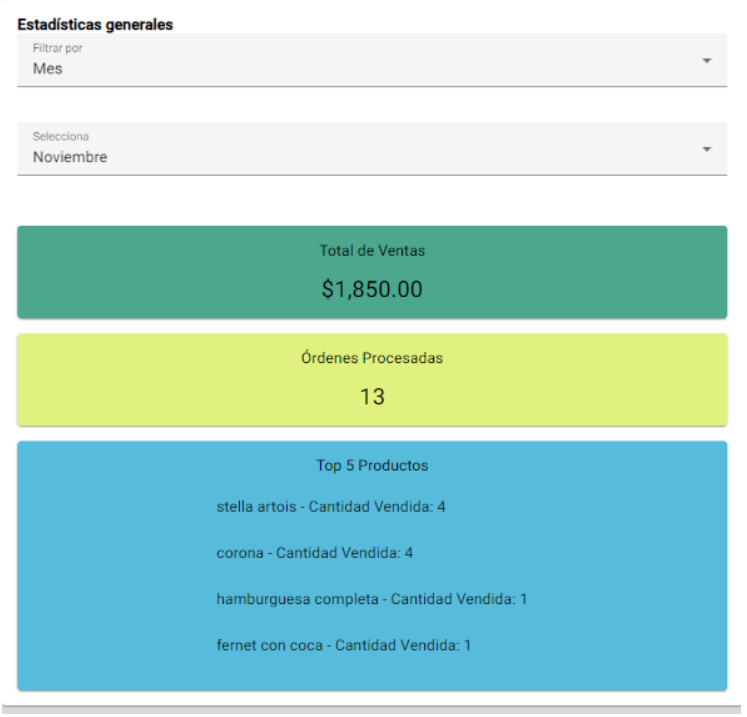


Vista con comandas en cocina:

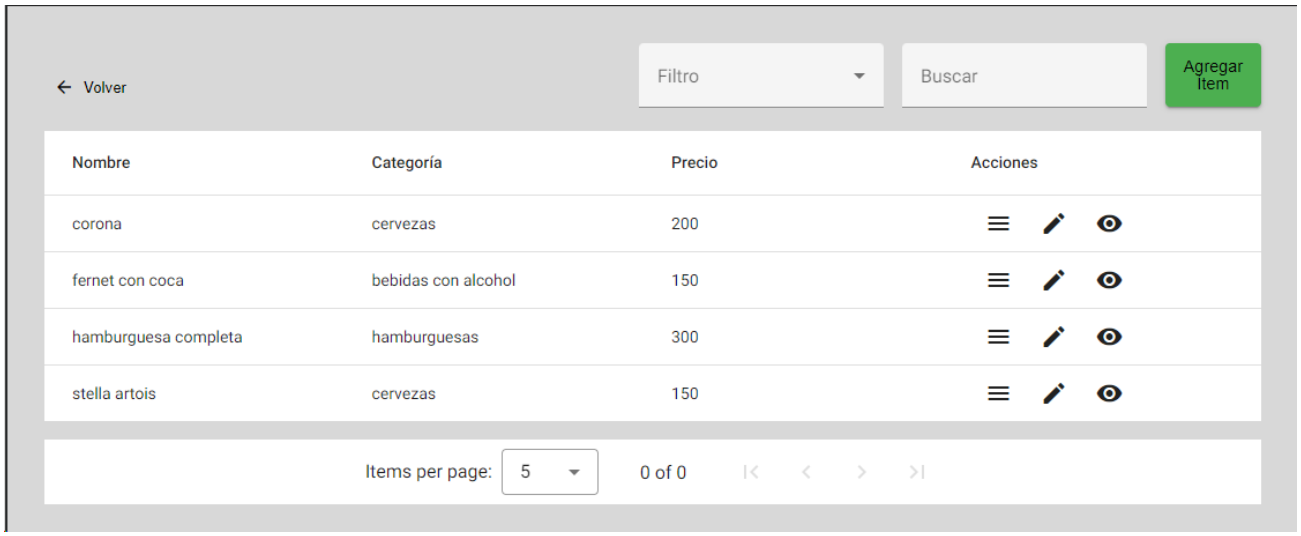


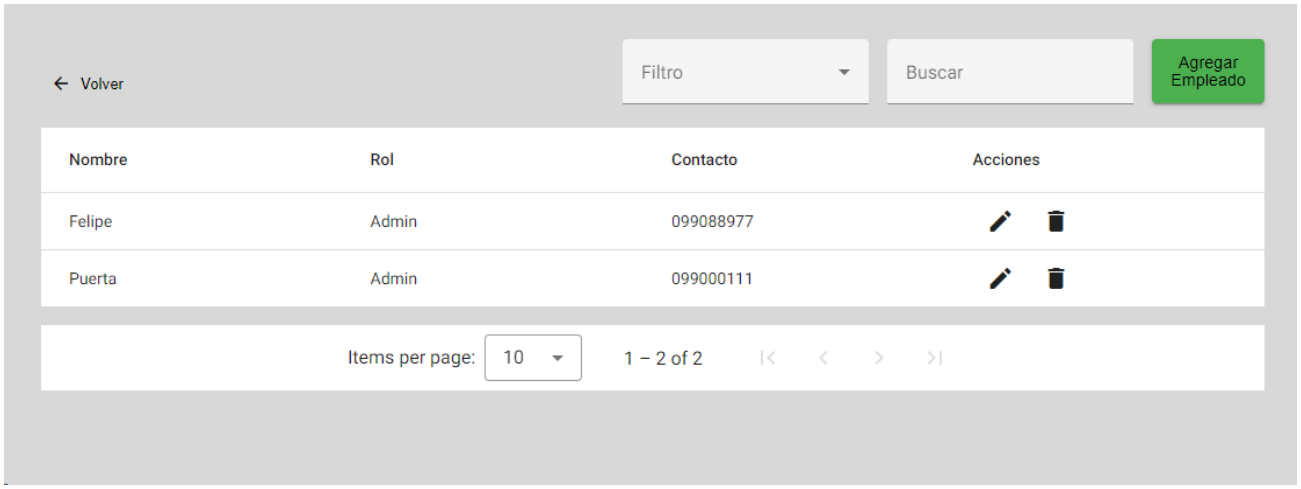
Estadísticas:





Interfaces para mantenimiento:





5. Pruebas Realizadas y Sus Resultados

5.1. Pruebas de Front-end e Integración

5.1.1. Plan de Testing Inicial con Cypress

El plan de testing inicial del equipo para el front-end e integración se centró en el uso de Cypress, una herramienta de testing de frontend todo en uno. Este plan incluía pruebas unitarias, de integración y end-to-end (E2E), con el objetivo de probar exhaustivamente cada componente y servicio tanto individualmente como en conjunto. El equipo logró incorporar la tecnología y ejecutar varios casos de prueba, resaltando la eficacia de Cypress para simular interacciones del usuario y validar el comportamiento esperado de los componentes y servicios. No obstante, a medida que avanzaba el proyecto, se hizo evidente que, a pesar de la potencia y flexibilidad de Cypress, el enfoque riguroso y el tiempo requerido para implementar completamente este plan de testing resultó ser un desafío significativo dadas las restricciones de tiempo del equipo.

5.1.2. Transición al Testing Exploratorio

Frente a esta situación, el equipo tomó la decisión estratégica de transitar hacia un enfoque de testing exploratorio. Este cambio permitió realizar un análisis más inmediato y profundo de la aplicación, enfocándose en la identificación de errores y problemas en tiempo real y en un contexto más práctico.

5.1.3. Puesta en marcha de Testing Exploratorio

A través de este método, el equipo pudo examinar detalladamente cada componente de Angular, evaluando aspectos como la responsividad, el manejo de datos de entrada/salida, la integración con otros componentes y la accesibilidad. Este enfoque permitió identificar y solucionar una variedad de problemas, incluyendo errores visuales, problemas de lógica y fallos en la integración, lo que resultó en un sistema significativamente más robusto y confiable.

El equipo utilizó un documento meticulosamente diseñado para registrar los distintos problemas identificados en el sistema. En este, se detalla la

descripción de cada inconveniente, el comportamiento actual observado, el comportamiento esperado, y se incluían ejemplos o capturas de pantalla cuando era pertinente. Gracias a este método organizado y sistemático, se lograron detectar un total de 34 ocurrencias, los cuales fueron abordados y resueltos tan pronto como fueron advertidos.

Sección	Ocurrencias	
General	100%	4/4
Mozo	100%	7/7
Admin	100%	2/2
Caja	100%	7/7
Cocina	100%	1/1
Back Menu	100%	5/5
Inventario	100%	5/5
Clientes	100%	1/1
Empleados	100%	1/1
Grupos	100%	1/1

Tabla 5. Análisis de ocurrencias en testing exploratorio

5.1.4. Impacto del Testing Exploratorio

El impacto del testing exploratorio en la calidad y rendimiento del software fue notablemente positivo. Al abordar cada problema de manera individual, desde desafíos en la interfaz de usuario hasta inconsistencias en la lógica de los componentes, el equipo logró optimizar la experiencia del usuario y la eficiencia del sistema. Los errores identificados, como problemas en la redirección de usuarios, mensajes inadecuados en los toast y desajustes visuales, fueron atendidos y solucionados de manera efectiva. Esta metodología enseñó la importancia de la flexibilidad en el proceso de pruebas, demostrando que un enfoque más abierto y adaptativo puede ser crucial para el éxito del proyecto, especialmente bajo restricciones de tiempo. Estas lecciones serán aplicadas en futuros proyectos para mejorar la eficiencia y la efectividad de las estrategias de testing.

5.2. Pruebas en backend

El enfoque para las pruebas de backend en el proyecto ExpressJS involucró la integración de herramientas especializadas, cada una seleccionada por su eficacia y sinergia con el entorno de desarrollo. Se utilizaron Jest, Supertest y Husky, herramientas que juntas formaron un marco robusto para garantizar la calidad y eficiencia del código.

5.2.1. Pruebas Unitarias con Jest

Objetivo: El propósito principal de las pruebas unitarias fue validar la funcionalidad de cada función y método en aislamiento, asegurando su correcto funcionamiento.
Método: El equipo empleó Jest para escribir pruebas precisas, utilizando técnicas como jest.fn() y mockResolvedValue para mockear dependencias externas. Este enfoque permitió simular respuestas de APIs o servicios externos, enfocándose en la precisión del código interno.

5.2.2. Pruebas de Integración con Supertest

Objetivo: Estas pruebas tuvieron como objetivo verificar la correcta integración y funcionamiento conjunto de los distintos módulos y servicios del backend.
Método: Utilizando Supertest, el equipo realizó peticiones a las rutas de la API, evaluando y asegurando respuestas adecuadas. Esta fase fue crucial para confirmar la correcta conexión entre rutas y controladores, y para garantizar respuestas coherentes y precisas.

5.2.3. Pruebas de Integración Continua (CI) con Husky

Objetivo: Asegurar la integridad continua del código, manteniéndolo siempre en un estado listo para el despliegue.
Método: Se configuró un pipeline de CI que automatizó la ejecución de pruebas unitarias y de integración en cada actualización del repositorio, ya sea mediante push o pull request.

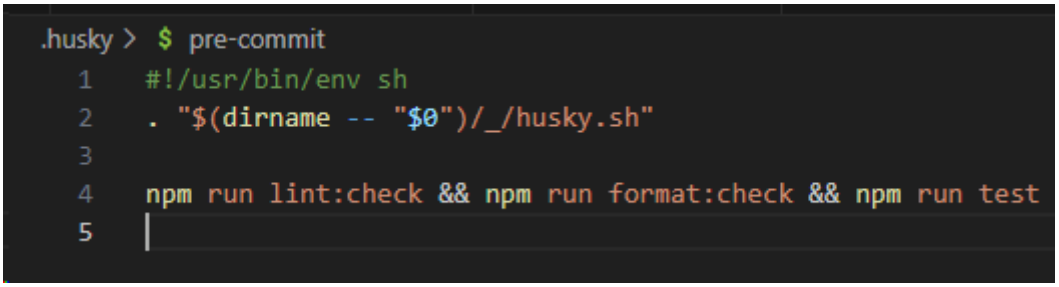


Figura 10. Configuración del pre-commit con Husky

```
Test Suites: 25 passed, 25 total
Tests:       241 passed, 241 total
Snapshots:   0 total
Time:        11.926 s, estimated 13 s
Ran all test suites.
[main 9756f99] Prueba Husky2
 46 files changed, 3924 insertions(+), 3367 deletions(-)
 create mode 100644 .husky/pre-commit
```

Figura 11. Feedback en línea de comandos luego de ejecutado el pre-commit

5.2.4. Impacto del Testing en Backend

La integración de Jest y Supertest ha sido fundamental para establecer un marco de testing robusto en nuestro backend ExpressJS. A través de Husky, hemos implementado pruebas automáticas que garantizan la calidad del código en cada commit. Es notable que, aunque la cobertura de pruebas alcanzó el 62%, este porcentaje refleja una concentración en áreas críticas del sistema, priorizando la profundidad y calidad de las pruebas sobre la cobertura total. Este enfoque selectivo ha sido clave para mantener la integridad y fiabilidad del código en las etapas finales del proyecto, asegurando que los componentes más esenciales estén debidamente validados y optimizados.

All files

62.08% Statements 1598/2574 **47.66%** Branches 500/1049 **50.85%** Functions 208/409 **61.96%** Lines 1556/2511

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
config	100%	2/2	100%	0/0	100%	0/0	100%	2/2
src	100%	18/18	100%	0/0	100%	1/1	100%	18/18
src/constants/estados	100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/constants/grupos	100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/constants/metodosPago	100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/constants/movimientos	100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/constants/roles	100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/controllers	93.15%	721/774	79.58%	382/480	100%	101/101	93.65%	679/725
src/error-handling	100%	13/13	85.71%	6/7	100%	2/2	100%	13/13
src/middleware	76%	19/25	56.25%	9/16	100%	4/4	76%	19/25
src/models	97.63%	124/127	70%	7/10	97.22%	35/36	97.63%	124/127
src/repositories	14.63%	65/444	0%	0/174	0%	0/130	14.84%	65/438
src/routes	100%	226/226	100%	0/0	100%	1/1	100%	226/226
src/routes/validations	100%	80/80	100%	0/0	100%	0/0	100%	80/80
src/services	37.5%	318/848	26.66%	96/360	48.06%	62/129	37.81%	318/841
src/webSocket	28.57%	2/7	0%	0/2	25%	1/4	33.33%	2/6
tests	100%	5/5	100%	0/0	100%	1/1	100%	5/5

Figura 12. Informe de cobertura Jest

File	Statements	
Config	100%	2/2
Src	100%	18/18
Src/Constants/estados	100%	1/1
Src/Constants/grupos	100%	1/1
Src/Constants/metodosPago	100%	1/1
Src/Constants/movimientos	100%	1/1
Src/Constants/roles	100%	1/1
Src/controllers	93.15%	721/774
Src/error_handling	100%	13/13
Src/middleware	76%	19/25
Src/models	97.63%	124/127
Src/repositories	14.63%	65/444
Src/Src/routes	100%	226/226
Src/validations	100%	80/80
Src/services	37.5%	318/848
Src/WebSocket	28.57%	2/7
tests	100%	5/5

Tabla 6. Resumen de cobertura

6. Conclusiones y Trabajo Futuro

6.1. Evaluación General del Proyecto

El proyecto para "La Puerta Verde Software" representa un hito significativo tanto en términos de desarrollo técnico como de crecimiento profesional del equipo. Los siguientes puntos sirven como argumentos para justificar esta reflexión.

6.1.1. Resumen de Logros

En retrospectiva, observar la transformación de una idea abstracta en un sistema funcional y tangible, que cumple con todos los requisitos planteados, logrado a través de un trabajo constante, intenso y perseverante, es motivo de orgullo y emoción para el equipo.
Además, el sistema tendrá un impacto positivo en la toma de órdenes y la gestión del negocio, cumpliendo con el objetivo de digitalizar los flujos de trabajo del local.

6.1.2. Evaluación del Proceso

Una particularidad de este proyecto es que el equipo de desarrollo ha trabajado en otros proyectos a lo largo de la carrera. Es por esta razón que se tiene un conocimiento de las habilidades y personalidades de los miembros del equipo. En consecuencia, el proceso de desarrollo de este sistema se ha dado de forma muy fluida.
Sin embargo, se han adoptado prácticas, metodologías y herramientas a lo largo del proyecto que reflejan un aprendizaje continuo y una mejora notable en la eficiencia y eficacia del desarrollo en comparación a performances anteriores del equipo.
Siendo específicos, la implementación de prácticas como reuniones semanales, comunicación constante y una división de tareas planificada y crítica, junto con el uso de herramientas como Trello, Figma y una utilización más avanzada de Git son ejemplos de prácticas adoptadas al inicio de este proyecto.
Se puede concluir entonces, que el equipo ha dado su mejor esfuerzo a lo largo de este proceso de trabajo logrando así su mejor cara, pero teniendo en cuenta que siempre se puede seguir mejorando.

6.2. Aprendizajes y Desafíos Enfrentados

6.2.1. Lecciones Aprendidas:

El equipo ha aprendido gracias a esta experiencia que llevar a cabo un proyecto de este porte desde que es solo una idea pasando por las diferentes fases para luego culminar con su liberación no es nada trivial. Se destaca el hecho de que se debe estar muy atento a cada etapa del desarrollo y a siempre mantener un ritmo intenso de trabajo para alcanzar los objetivos.
Además esta experiencia ha fortalecido la cohesión del equipo y la capacidad de tomar decisiones estratégicas en la asignación de tareas.

6.2.2. Desafíos y Obstáculos:

A lo largo del proyecto se presentaron múltiples desafíos, pero se destacan aquellos más grandes y que sobrellevarlos dejó un aprendizaje enriquecedor.
En primer lugar, el hecho de enfrentarse con un cliente real con el cual se debe interactuar en un ida y vuelta constante, negociando cara a cara, sumado a la responsabilidad de ser sinceros con las limitaciones del equipo pero sin decepcionar las expectativas del cliente fue un gran desafío en la etapa más temprana del proyecto.
Otro desafío fue que se tomó la decisión de optar por un backend en un lenguaje y framework nuevo, lo cual requirió una etapa de intensa investigación y desarrollo de prototipos con el objetivo de luego poder pasar a la implementación del sistema con un conocimiento más amplio.
La fase de deploy también fue obstáculo a sortear, debido a la inexperiencia del equipo en esta fase. Al igual que con la tecnología backend, esto requirió una intensa investigación, pruebas de ensayo y error en diferentes proveedores de hosting y una gran compenetración con conceptos de infraestructura.

6.3. Propuestas para Trabajos Futuros o Mejoras

6.3.1. Áreas de Mejora:

Aunque ciertas funcionalidades, como el envío de resumen de compra por WhatsApp y la implementación de un tótem de autogestión, fueron

relegadas por cuestiones de alcance y tiempo, su futura integración podría mejorar significativamente la experiencia del usuario y la eficiencia del negocio.

Como otra posible mejora a la solución está el sistema de menú de un cliente, dicho sistema puede mejorarse en más de un aspecto, como por ejemplo restringir el acceso al mismo sólo cuando se está utilizando la red del local, para así evitar fallas de seguridad. A su vez, este sistema podría enviar un mensaje al cliente cuando su órden esté lista, esto evitaría contacto cliente-mozo volviendo más rápido el proceso de órden.

6.3.2. Ideas para Trabajos Futuros:

Continuar el proceso de pulir y perfeccionar las prácticas, metodologías y herramientas utilizadas, aprendiendo de la experiencia y adaptándose a nuevos desafíos.

La posibilidad de explorar y adoptar nuevas tecnologías y frameworks que puedan aportar soluciones más eficientes y avanzadas.

7. Referencias

Blancarte, O. (n.d.). *Arquitectura en Capas - Estilo arquitectónico*. Reactive Programming. Retrieved November 21, 2023, from https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

\/. (2020, November 25). TortugaCode. Retrieved November 21, 2023, from https://www.tortugacode.com/arquitectura-en-capas-que-es-y-cuando-usarla/

Durán, M. (2023, April 11). *Qué es la arquitectura en capas, ventajas y ejemplos*. Blog de HubSpot. Retrieved November 21, 2023, from https://blog.hubspot.es/website/que-es-arquitectura-en-capas

Londoño, P. (2023, January 17). *Qué es MySQL, para qué sirve y características principales*. Blog de HubSpot. Retrieved November 21, 2023, from https://blog.hubspot.es/website/que-es-mysql

Montes, J. C. (n.d.). *¿Qué beneficios y ventajas tiene una API REST y por qué elegirla?* Chakray. Retrieved November 22, 2023, from https://www.chakray.com/es/cuales-son-las-ventajas-de-una-api-rest/

Rodríguez, F. (2023, August 17). *Principales características de los WebSockets*. KeepCoding. Retrieved November 22, 2023, from https://keepcoding.io/blog/principales-caracteristicas-de-los-websockets/

Salmerón, A. (2023, July 1). *MySQL Ventajas y Desventajas: Un Análisis Completo*. Informatica y Tecnologia Digital. Retrieved November 22, 2023, from https://informatecdigital.com/bases-de-datos/mysql-ventajas-y-desventajas-un-analisis-completo/

Blanch, A. (2017, April 12). *NodeJS: qué es y qué ventajas ofrece*. Arsys. Retrieved November 22, 2023, from https://www.arsys.es/blog/nodejs-framewok

Documentation, T. (n.d.). *TypeScript documentation — DevDocs*. DevDocs. Retrieved November 22, 2023, from https://devdocs.io/typescript/

Fierro, E. (n.d.). *Javascript en el backend: Todo lo que necesitas saber para empezar – Eduardo Fierro Pro – Blog*. Eduardo Fierro Pro. Retrieved November 23, 2023, from https://eduardofierro.pro/blog/javascript-en-el-backend-todo-lo-que-necesitas-saber-para-empezar/

Javascript, D. (2023, July 24). *JavaScript | MDN*. MDN Web Docs. Retrieved November 23, 2023, from https://developer.mozilla.org/es/docs/Web/JavaScript

nodeJS, N. (n.d.). *Documentation*. Node.js. Retrieved November 23, 2023, from https://nodejs.org/en/docs

Novoseltseva, E. (2020, December 5). *Las principales ventajas de usar Typescript*. Apiumhub. Retrieved November 23, 2023, from https://apiumhub.com/es/tech-blog-barcelona/usar-typescript/

Sequelize, S. (n.d.). *Sequelize v6*. Sequelize. Retrieved November 23, 2023, from https://sequelize.org/docs/v6/

Angular, D. (n.d.). *Introduction to the Angular docs*. Angular. Retrieved November 23, 2023, from https://angular.io/docs

ExpressJS, D. (n.d.). Express - Infraestructura de aplicaciones web Node.js. Retrieved November 23, 2023, from https://expressjs.com/es/

Jesús, .. (2023, November 12). *Introducción a Express.js: Explorando sus Funciones y Beneficios*. Dongee. Retrieved November 23, 2023, from https://www.dongee.com/tutoriales/que-es-y-para-que-sirve-express-js/

Martínez, D. (2021, May 3). *Qué es Angular, características y versiones*. OpenWebinars. Retrieved November 23, 2023, from https://openwebinars.net/blog/que-es-angular-2021/

ChatGPT, .. (n.d.). ChatGPT. Retrieved November 23, 2023, from https://chat.openai.com/

ConsoleNinja, D. (n.d.). *wallabyjs/console-ninja: Repository for Console Ninja questions and issues*. GitHub. Retrieved November 23, 2023, from https://github.com/wallabyjs/console-ninja#get-started

Discord, .. (n.d.). Discord | Your Place to Talk and Hang Out. Retrieved November 23, 2023, from https://discord.com/

Docs, .. (n.d.). Ayuda de Editores de Documentos de Google. Retrieved November 23, 2023, from https://support.google.com/docs/?hl=es#topic=1382883

DrawIo, .. (n.d.). draw.io. Retrieved November 23, 2023, from https://app.diagrams.net/

Figma, .. (n.d.). Figma: The Collaborative Interface Design Tool. Retrieved November 23, 2023, from https://www.figma.com/

Github, D. (n.d.). *GitHub Docs*. GitHub Docs. Retrieved November 23, 2023, from https://docs.github.com/es

Github Copilot, D. (n.d.). *GitHub Copilot · Your AI pair programmer · GitHub*. GitHub. Retrieved November 23, 2023, from https://github.com/features/copilot

Trello, D. (n.d.). *Guías de Trello: Introducción a Trello*. Trello. Retrieved November 23, 2023, from https://trello.com/es/guide

VSCode, D. (n.d.). *Documentation for Visual Studio Code*. Visual Studio Code. Retrieved November 23, 2023, from https://code.visualstudio.com/docs

Husky. (n.d.). *Modern native git hooks made easy*. husky | husky. Retrieved December 4, 2023, from https://typicode.github.io/husky/

JestJS. (n.d.). *Documentation*. Jest · Delightful JavaScript Testing. Retrieved December 4, 2023, from https://jestjs.io/

SuperTest. (2023, 04 12). *ladjs/supertest*. GitHub. Retrieved December 4, 2023, from https://github.com/ladjs/supertest

Drumond, C. (n.d.). What is Scrum? [+ How to Start]. Atlassian. Retrieved December 5, 2023, from https://www.atlassian.com/agile/scrum

Naybour, P. (n.d.). What Is a Gantt Chart? | Definition & Examples. APM. Retrieved December 5, 2023, from https://www.apm.org.uk/resources/find-a-resource/gantt-chart/