16/06/2025

GASTRO & HUB

Trabajo de Fin de Grado

Abel Moro Paje

DESARROLLO DE APLICACIONES MULTIPLATAFORMA 2024/25

CONTENTS

Bienvenido a la documentación del proyecto "Gastro & Hub", una aplicación multiplataforma para la gestión hostelera desarrollada como Trabajo Final de Grado (TFG) del CFGS Desarrollo de Aplicaciones Multiplataforma (DAM).

1. RESUMEN

"Gastro & Hub" es una aplicación multiplataforma diseñada para revolucionar la gestión de bares y restaurantes, desde pequeños negocios locales hasta establecimientos medianos. En un sector donde la rapidez, la coordinación y el control de recursos son cruciales, este proyecto resuelve los problemas de comunicación lenta entre camareros y cocina, la falta de visibilidad sobre el inventario y la complejidad de gestionar pagos y finanzas de forma sencilla. La solución permite a los camareros registrar comandas desde sus móviles en tiempo real, al dueño controlar existencias con análisis predictivo para prever compras, y a todo el equipo coordinarse con notificaciones instantáneas y un diseño claro de mesas.

Además, ofrece pagos digitales accesibles y un cierre diario que simplifica la contabilidad. Desarrollada como parte de la formación en Desarrollo de Aplicaciones Multiplataforma, "Gastro & Hub" destaca por ser una herramienta moderna, de muy bajo coste (solo requiere hosting de la base de datos), adaptable a cualquier negocio y accesible desde dispositivos cotidianos, mejorando la operación diaria y la experiencia del cliente en el entorno hostelero.

2. INTRODUCCIÓN

Este proyecto nace en el marco del Trabajo Final de Grado del CFGS en Desarrollo de Aplicaciones Multiplataforma (DAM), como una oportunidad para aplicar los conocimientos adquiridos en un caso real y relevante: la gestión tecnológica de bares y restaurantes, un sector clave en España que enfrenta retos como la digitalización accesible y la eficiencia operativa. "Gastro & Hub" es una propuesta ambiciosa que combina modernidad y practicidad para ofrecer una solución integral, adaptada a las demandas actuales del mercado hostelero.

MÓDULOS IMPLICADOS EN EL DESARROLLO DEL PROYECTO

- **Programación:** Desarrollo del back-end con Java y Spring Boot, y del front-end con Dart y Flutter, aplicando lógica para coordinar funciones en tiempo real.
- **Bases de Datos:** Diseño y manejo de PostgreSQL para almacenar y organizar información de comandas, existencias y empleados.
- **Sistemas Informáticos:** Uso de Docker para configurar y desplegar el back-end en entornos locales o escalables.
- **Desarrollo de Interfaces:** Creación de pantallas intuitivas y accesibles en Flutter, adaptadas a móviles y ordenadores.

INTRODUCCIÓN A LA SOLUCIÓN OBJETO DE ESTE PROYECTO

"Gastro & Hub" actúa como un centro operativo digital para bares y restaurantes, conectando a camareros, cocineros y dueños en una plataforma única y moderna. Su objetivo es centralizar las operaciones del negocio en una única herramienta eficiente y accesible.

La aplicación permite registrar comandas al instante desde cualquier dispositivo móvil, facilitando así la comunicación entre sala y cocina. También ofrece la posibilidad de gestionar mesas y reservas mediante un diseño visual intuitivo, mejorando la organización del servicio.

Además, permite controlar las existencias tanto de ingredientes como de platos mediante análisis predictivo, ayudando a prever necesidades de compra y evitar problemas de stock. La solución también incluye la aceptación de pagos digitales directamente desde la app, lo que agiliza el cobro y mejora la experiencia del cliente.

Otra funcionalidad clave es la generación de informes detallados, que permiten a los responsables tomar decisiones informadas basadas en datos reales y actualizados. Esto fortalece la gestión y el control del negocio en su día a día.

Construida con tecnologías actuales como Spring Boot, Flutter y PostgreSQL, y desplegada mediante Docker, "Gastro & Hub" es una solución moderna, de muy bajo coste, adaptable a cualquier tipo de negocio hostelero y accesible desde dispositivos estándar. Su enfoque práctico y su tecnología avanzada la convierten en una herramienta que realmente marca la diferencia en el sector.

3. JUSTIFICACIÓN DEL PROYECTO Y OBJETIVOS

ESTADO DEL ARTE

El mercado de soluciones tecnológicas para hostelería incluye sistemas como Toast POS, Square y OpenTable, que gestionan comandas, reservas y pagos. No obstante, estas herramientas suelen ser costosas, requerir hardware especializado (como terminales POS) y están dirigidas a grandes cadenas o mercados extranjeros.

Esto deja a muchos bares y restaurantes pequeños en España con pocas opciones. En su lugar, recurren a procesos manuales (libretas, tickets) o software genérico sin coordinación instantánea, análisis predictivo ni pagos digitales.

Según estimaciones de 2024, más del 60 % de estos negocios no adopta soluciones integradas por su coste o complejidad. Gastro & Hub cubre ese vacío con una propuesta moderna, de muy bajo coste (solo requiere hosting de la base de datos), accesible desde móviles estándar y adaptable a cualquier modelo de negocio.

Su enfoque en análisis predictivo, pagos digitales y coordinación en tiempo real la posiciona como una alternativa innovadora frente a los competidores tradicionales.

OBJETIVOS DEL PROYECTO

Desarrollar una aplicación multiplataforma para:

- Permitir a los camareros registrar comandas y gestionar mesas desde sus móviles en tiempo real, asegurando un servicio rápido y sin errores.
- Ayudar al dueño a controlar el inventario de ingredientes y platos, con análisis predictivo que sugiera compras basadas en tendencias para evitar faltantes y desperdicios.
- Facilitar pagos digitales directamente desde la app y un cierre diario de caja claro, simplificando la gestión financiera del negocio.
- Generar informes detallados sobre comandas, existencias y reservas, con predicciones que apoyen decisiones como ajustar menús o planificar personal.
- Coordinar al equipo con notificaciones instantáneas sobre comandas, existencias bajas o mesas reservadas, mejorando la comunicación interna.
- Permitir al dueño gestionar menús y ofrecer sugerencias de comandas populares, adaptando la oferta a los clientes.
- Ofrecer una solución moderna, de muy bajo coste y escalable que funcione en cualquier móvil o tablet, adaptándose a bares y restaurantes de todos los tamaños.

4. ANÁLISIS

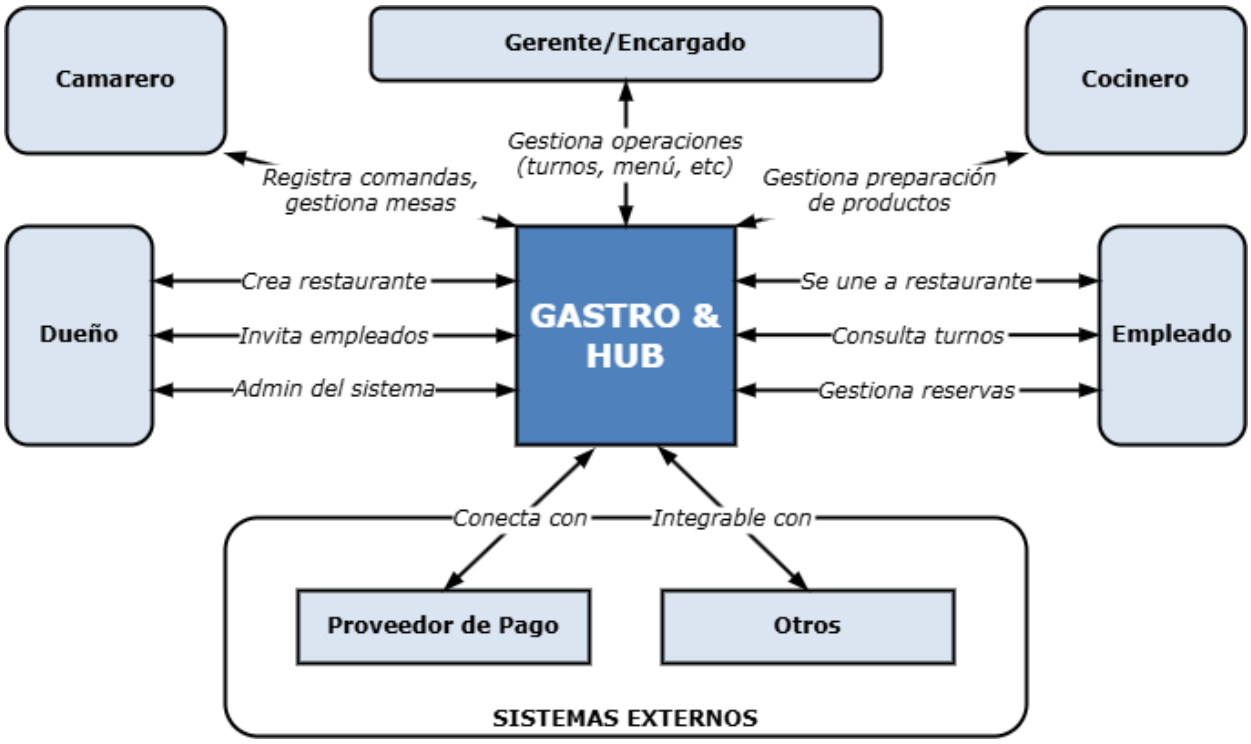
INTRODUCCIÓN

El apartado de análisis detalla las capacidades principales de Gastro & Hub, definiendo cómo la aplicación apoyará las operaciones diarias de bares y restaurantes. Este análisis tiene como objetivo ofrecer una visión clara y estructurada del funcionamiento general del sistema.

En este contexto, se describen los módulos funcionales que dan forma a la solución, abarcando desde el registro inicial del negocio y sus empleados hasta la gestión de comandas y el análisis del rendimiento. Cada módulo representa una parte esencial del flujo de trabajo hostelero, permitiendo una comprensión clara de sus funciones y de cómo interactúan con los distintos tipos de usuario.

Este análisis sienta así las bases para especificar cómo la aplicación responderá a las necesidades reales del equipo hostelero. Se prioriza la simplicidad de uso, la coordinación en tiempo real entre los distintos roles del personal, y un acceso controlado que parte del dueño como creador y administrador del restaurante.

DIAGRAMA DE CONTEXTO



REQUISITOS FUNCIONALES

1. REGISTRO DEL RESTAURANTE

Este módulo permite al dueño iniciar el uso de la aplicación registrándose y creando un restaurante único para su negocio. Tras completar el pago del servicio, que cubre el hosting de los datos, se genera un espacio exclusivo donde toda la operativa del establecimiento quedará organizada, sentando las bases para las demás funciones.

RF-001: Pantalla de OnBoarding

- La aplicación muestra información sobre la misma para atraer usuarios y principalmente dueños que quieran registrar su local.
- Condición: Ninguna, es el primer paso para usar la aplicación.

RF-002: Registrar usuario

- El usuario inicia el uso de la aplicación creando una cuenta personal con su nombre, email y contraseña.
- Condición: Ninguna, es el primer paso para usar la aplicación.

RF-003: Crear un restaurante

- El dueño define un restaurante en la aplicación, dándole un nombre único que lo identifique como su negocio.
- Condición: El usuario debe estar registrado (RF-002).

RF-004: Seleccionar un plan de pago

- El dueño selecciona un plan de pago de los disponibles en la aplicación.
- Condición: El restaurante debe estar creado (RF-003)

RF-005: Pagar el servicio del restaurante

- El dueño realiza un pago anual o mensual online mediante una pasarela segura para activar el restaurante, según el tamaño del negocio (mesas y empleados).
- Condición: El restaurante debe estar creado (RF-003) y el plan de pago debe estar seleccionado (RF-004).

RF-006: Activar el espacio del restaurante

- La aplicación genera un espacio único para el restaurante del dueño, donde se gestionarán todas sus operaciones. Se genera un espacio principal para el restaurante.
- Condición: El pago del servicio debe estar confirmado (RF-005).

2. GESTIÓN DE USUARIOS Y ROLES

Una vez creado el restaurante, el dueño invita a los empleados (camareros, cocineros, gerentes) y les asigna roles que determinan lo que cada uno puede hacer dentro de la aplicación. Los gerentes tienen permisos adicionales, como organizar turnos o reservas, mientras que solo el dueño puede añadir nuevos usuarios o cambiar la configuración principal, asegurando un control claro y jerárquico sobre quién participa en la gestión.

RF-007: Consultar código de invitación

- El dueño puede consultar el código de invitación desde el menú situado en el panel de control del local, el sistema proporciona el código como QR y en texto plano.
- Condición: El restaurante debe estar activado (RF-006).

RF-008: Regenerar código de invitación

- El dueño puede regenerar el código de invitación en cualquier momento, esto evita que nuevos usuarios se unan al local con el código antiguo.
- Condición: El restaurante debe estar activado (RF-006).

RF-009: Acceder al restaurante

- El empleado accede al restaurante mediante un código de invitación proporcionado por el dueño.

- Condición: El empleado debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el código de invitación debe estar disponible (RF-007).

RF-010: Acceder al restaurante (QR)

- El empleado accede al restaurante mediante el escaneo de un código QR proporcionado por el dueño.
- Condición: El empleado debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el código QR debe estar disponible (RF-007).

RF-011: Consultar empleados

- El dueño puede consultar todos los usuarios que forman parte de su establecimiento, así como sus datos personales relevantes como nombre, email y teléfono.
- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

RF-012: Asignar un rol a un empleado

- Una vez que el empleado se ha unido al restaurante, el dueño le asigna rol (como camarero, cocinero, gerente) para definir sus permisos dentro del espacio de trabajo.
- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

RF-013: Modificar el rol de un empleado

- El dueño cambia el rol de un empleado para darle acceso a las distintas funcionalidades del restaurante.
- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

RF-014: Expulsar a un empleado

- El dueño quita a un empleado del restaurante, bloqueando su acceso a la aplicación. Esta acción se registra en un historial de auditoría para trazabilidad.
- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

RF-015: Consultar perfil y datos personales

- El usuario puede consultar sus datos personales y rol desde una pantalla de perfil del restaurante.
- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

RF-016: Editar datos personales

- El usuario puede editar sus datos personales desde la pantalla de perfil del restaurante.

- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

RF-017: Abandonar restaurante o local

- El usuario puede decidir abandonar el restaurante.
- Condición: El usuario debe estar registrado (RF-002), el restaurante debe estar activado (RF-006), y el usuario debe haber accedido al restaurante (RF-009 o RF-010).

3. ORGANIZACIÓN DE MESAS

Este módulo permite al dueño o gerente diseñar un plano visual del restaurante, colocando las mesas disponibles con sus números correspondientes. Este diseño sirve como base para asignar comandas y reservas, proporcionando una vista clara del espacio físico que todos los empleados usarán en su día a día.

RF-018: Crear una zona

- El dueño o gerente crea una zona del restaurante y le da un nombre, este corresponde a una zona física de este (interior, terraza, etc).
- Condición: El restaurante debe estar activado (RF-006) y el usuario debe tener rol de dueño o gerente (RF-012).

RF-019: Editar una zona

- El dueño o gerente edita el nombre de una zona del restaurante, este corresponde a una zona física de este (interior, terraza, etc).
- Condición: El restaurante debe estar activado (RF-006) y el usuario debe tener rol de dueño o gerente (RF-012).

RF-020: Eliminar una zona

- El dueño o gerente o gerente crea un plano del restaurante y le da un nombre, este corresponde a una zona de este (interior, terraza, etc).
- Condición: El restaurante debe estar activado (RF-006), el usuario debe tener rol de dueño o gerente (RF-012), y la zona debe existir sin mesas activas (RF-018 y RF-021).

RF-021: Añadir una mesa al plano

- El dueño o gerente agrega una mesa a un plano, asignándole un número único y capacidad.
- Condición: La zona debe existir (RF-018) y el usuario debe ser dueño o gerente (RF-012).

RF-022: Editar una mesa del plano

- El dueño o gerente edita una mesa de a un plano, cambiando su número único y/o capacidad.

- Condición: La mesa debe existir en la zona (RF-021) y el usuario debe ser dueño o gerente (RF-012).

RF-023: Eliminar una mesa del plano

- El dueño o gerente elimina una mesa del plano cuando ya no se usa en el restaurante.
- Condición: La mesa debe existir en la zona (RF-021), no estar reservada ni ocupada (RF-027 o RF-029), y el usuario debe ser dueño o gerente (RF-012).

RF-024: Diseñar el plano de mesas

- El dueño o gerente distribuye las mesas del plano en un grid que simula su posición física.
- Condición: Deben existir mesas en la zona (RF-021) y el usuario debe ser dueño o gerente (RF-012).

RF-025: Mover una mesa en el plano

- El dueño o gerente cambia la posición de una mesa en el plano para ajustarla al espacio real.
- Condición: La mesa debe existir en la zona (RF-021) y el usuario debe ser dueño o gerente (RF-012).

RF-026: Consultar el plano de mesas

- Todos los empleados ven el plano del restaurante con las mesas numeradas y su estado (disponible, reservada, ocupada).
- Condición: El plano debe estar creado (RF-024).

4. GESTIÓN DE RESERVAS

Con las mesas definidas, el equipo puede registrar reservas para los clientes, indicando fechas, horas y el número de personas. El módulo muestra qué mesas están disponibles, reservadas o ocupadas, permitiendo actualizar su estado en cualquier momento para planificar el servicio de manera eficiente.

RF-027: Hacer una reserva de mesa

- El gerente o camarero registra una reserva para un cliente, seleccionando una mesa, día, hora y número de personas.
- Condición: Deben existir mesas en el plano (RF-021) y el usuario debe tener rol de gerente o camarero (RF-012).

RF-028: Hacer una reserva de varias mesas

- El gerente o camarero registra una reserva para un cliente, seleccionando múltiples mesas y definiendo día, hora y número de personas.

- Condición: Deben existir mesas en el plano (RF-021) y el usuario debe tener rol de gerente o camarero (RF-012).

RF-029: Confirmar la llegada de una reserva

- El gerente o camarero marca una mesa reservada como ocupada cuando el cliente llega al restaurante.
- Condición: La mesa debe estar reservada (RF-027 o RF-028).

RF-030: Cancelar una reserva

- El gerente o camarero marca una mesa reservada como disponible si el cliente no llega o cancela la reserva.
- Condición: La mesa debe estar reservada (RF-027 o RF-028).

RF-031: Ver el estado de las mesas

- El equipo consulta el plano del restaurante para ver qué mesas están disponibles, reservadas u ocupadas, con la hora de cada estado.
- Condición: El plano de mesas debe estar creado (RF-024).

RF-032: Visualizar monitor de reservas

- El equipo puede acceder a una pantalla dedicada que muestra todas las reservas actuales y próximas, con detalles como mesa asignada, hora y estado (pendiente, confirmada, cancelada), para planificar el servicio de forma eficiente.
- Condición: Deben existir reservas registradas (RF-027 o RF-028).

5. GESTIÓN DE INVENTARIO

Este módulo ayuda al dueño a controlar las existencias de ingredientes, bebidas y platos preparados, registrando lo que hay disponible y alertando cuando algo está por agotarse. Incluye análisis predictivo basado en tendencias de consumo para sugerir compras, ayudando a evitar faltantes o desperdicios y optimizando los recursos del negocio. Además, este módulo permite al dueño definir tipos de atributos, como alérgenos o características dietéticas, y asignarlos a los ingredientes, mejorando la información disponible para clientes con necesidades específicas.

RF-033: Registrar un ingrediente en el inventario

- El dueño añade un nuevo ingrediente, como "carne" o "harina", indicando su nombre, unidad (kilos, litros), cantidad inicial, alérgenos, etc.
- Condición: El restaurante debe estar activado (RF-006).

RF-034: Registrar un ingrediente compuesto en el inventario

- El dueño añade un nuevo ingrediente compuesto, como "salsa de tomate" o "masa de pizza", indicando su nombre, unidad (kilos, litros), cantidad inicial, alérgenos, etc y los ingredientes simples que lo componen con sus cantidades.
- Condición: El restaurante debe estar activado (RF-006) y deben existir ingredientes simples (RF-033).

RF-035: Registrar un producto en el inventario

- El dueño añade un producto, como "hamburguesa" o "refresco", indicando su nombre y precio, etc y los ingredientes que lo componen.
- Condición: El restaurante debe estar activado (RF-006).

RF-036: Añadir más cantidad a un ingrediente o ingrediente compuesto

- El dueño aumenta la cantidad de un ingrediente existente, como al recibir una comanda del proveedor.
- Condición: El ingrediente debe haber sido registrado (RF-033 o RF-034).

RF-037: Modificar datos de un ingrediente o ingrediente compuesto

- El dueño cambia el nombre, unidad o cantidad de un ingrediente para corregirlo o actualizarlo.
- Condición: El ingrediente debe estar registrado (RF-033 o RF-034).

RF-038: Modificar datos de un producto

- El dueño cambia el nombre o cantidad de un producto para ajustarlo.
- Condición: El producto debe estar registrado (RF-035).

RF-039: Eliminar un ingrediente o ingrediente compuesto del inventario

- El dueño quita un ingrediente del inventario cuando ya no se usa en el restaurante.
- Condición: El ingrediente debe estar registrado (RF-033 o RF-034) y no estar en el menú (RF-048) ni en comandas activas (RF-060 o RF-069).

RF-040: Eliminar un producto del inventario

- El dueño quita un producto del inventario cuando ya no se usa.
- Condición: El producto debe estar registrado (RF-035) y no estar en el menú (RF-048) ni en comandas activas (RF-060 o RF-069).

RF-041: Establecer un nivel mínimo para un ingrediente o ingrediente compuesto

- El dueño define cuántas unidades de un ingrediente deben quedar antes de que se considere bajo.
- Condición: El ingrediente debe estar registrado (RF-033 o RF-034).

RF-042: Consultar el inventario

- El dueño o gerente consulta una lista de todos los ingredientes, mostrando su cantidad y si están bajos o agotados.
- Condición: Debe haber ingredientes registrados (RF-033 o RF-034).

RF-043: Restar ingredientes al completar una comanda

- La aplicación reduce la cantidad de ingredientes usados en una comanda cuando este se completa, según los productos seleccionados.
- Condición: La comanda debe estar completado (RF-075).

RF-044: Alertar sobre ingredientes bajos

- El dueño recibe una alerta cuando un ingrediente cae por debajo de su nivel mínimo, indicando que debe reponerlo.
- Condición: El nivel mínimo debe estar definido (RF-041).

RF-045: Consultar alérgenos y otros atributos

- El sistema permite a los usuarios acceder a la lista de alérgenos y otros atributos para consultar su icono representativo, nombre y descripción.
- Condición: El usuario debe estar asignado al restaurante (RF-009 o RF-010).

RF-046: Asignar Atributos a Ingredientes

- El sistema permite al dueño o gerente asociar atributos específicos (ej. "contiene gluten") a ingredientes registrados.
- Condición: Deben existir ingredientes registrados (RF-033 o RF-034).

6. GESTIÓN DE MENÚS

Con el inventario establecido, este módulo permite al dueño crear y actualizar los menús del restaurante, usando los ingredientes y existencias disponibles para definir una carta fija y ofertas especiales (como menús del día o promociones). Ofrece la posibilidad de sugerir comandas populares a los camareros o clientes, adaptando la oferta según las preferencias y el consumo habitual, para mejorar la experiencia y las ventas. Además, se pueden definir menús contextuales para distintos momentos, como un 'Menú Desayuno' con precios diferenciados, y crear ofertas especiales, como un 'Menú del Día' con precio reducido, asociando productos y definiendo descuentos para atraer a más clientes.

RF-047: Crear una categoría de menú

- El dueño añade una categoría, como "bebidas" o "postres", para organizar los productos que se ofrecen a los clientes.
- Condición: El restaurante debe estar activado (RF-006).

RF-048: Añadir un producto al menú

- El dueño incluye un producto del inventario en el menú, como un plato o bebida, asignándole un precio y una categoría.
- Condición: El producto debe estar registrado (RF-035) y la categoría creada (RF-047).

RF-049: Modificar un producto del menú

- El dueño cambia el nombre, precio o categoría de un producto en el menú para ajustarlo.
- Condición: El producto debe estar en el menú (RF-048).

RF-050: Eliminar un producto del menú

- El dueño quita un producto del menú cuando ya no se ofrece a los clientes.
- Condición: El producto debe estar en el menú (RF-048) y no estar en comandas activas (RF-060 o RF-069).

RF-051: Consultar el menú para comandas

- Los camareros ven una lista organizada por categorías con todos los productos del menú al registrar comandas.
- Condición: Debe haber productos en el menú (RF- 048).

RF-052: Crear una oferta especial

- El dueño define una oferta especial, como un menú del día, combinando productos del inventario con un precio único.
- Condición: Debe haber productos en el inventario (RF-035).

RF-053: Generar un documento del menú

- El dueño crea un documento con todas las categorías y productos del menú para compartirlo con los clientes.
- Condición: Debe haber productos en el menú (RF-048).

RF-054: Generar un código QR del menú

- El dueño crea un código QR que lleva al documento del menú, para que los clientes lo vean fácilmente.
- Condición: El documento del menú debe estar generado (RF-053).

RF-055: Ver el documento y código del menú

- Todos los empleados consultan el documento del menú y su código QR en la aplicación para compartirlo con los clientes.
- Condición: El código QR debe estar creado (RF-054).

RF-056: Asignar Atributos a Productos del Menú

- El sistema asigna automáticamente atributos a los productos del menú según los atributos de sus ingredientes, con opción de asignación manual por el dueño.
- Condición: Deben existir productos en el menú (RF-048) y atributos asignados a ingredientes (RF-046).

RF-057: Crear y Gestionar Menús Contextuales

- El sistema permite al dueño definir menús para contextos específicos (ej. "cena"), asignando ítems con precios opcionales.
- Condición: Deben existir productos en el menú (RF-048).

RF-058: Crear y Gestionar Ofertas Especiales

- El sistema permite al dueño crear promociones (ej. "combo familiar"), asociando productos y definiendo descuentos.
- Condición: Deben existir productos en el inventario (RF-035).

RF-059: Cambiar la disponibilidad de ítem de menú

- El gerente cambia la disponibilidad de un ítem de menú manualmente.
- Condición: El ítem de menú debe estar definido (RF-048).

7. REGISTRO DE COMANDAS

Con los menús listos, los camareros pueden registrar las comandas de los clientes desde sus móviles, seleccionando ítems del menú y asignándolos a una mesa específica del plano. Las comandas se envían a la cocina en tiempo real, iniciando el proceso de preparación y coordinación entre sala y cocina.

RF-060: Registrar una nueva comanda

- El camarero crea una comanda para una mesa, seleccionando productos del menú y enviándolo a la cocina en tiempo real.
- Condición: El plano de mesas debe estar creado (RF-024) y el menú debe tener productos (RF-048).

RF-061: Añadir una nota a una comanda

- El camarero incluye una nota en una comanda, como "para llevar", para indicar preferencias del cliente antes de prepararlo.
- Condición: La comanda debe estar registrada (RF-060 o RF-069) y no estar en estado 'en preparación'.

RF-062: Marcar una comanda como urgente

- El camarero señala una comanda como prioritario para que la cocina lo prepare antes que otros.

- Condición: La comanda debe estar registrado (RF-060 o RF-069).

RF-063: Mostrar el tiempo estimado de una comanda

- El camarero visualiza tres estimaciones aproximadas del tiempo de preparación (optimista, media, pesimista), calculadas según la carga actual y los tiempos de preparación.
- Condición: La comanda debe estar registrado (RF-060 o RF-069).

RF-064: Consultar las comandas activos

- Los camareros y cocineros ven una lista de comandas actuales, filtrada por estado, mesa o fecha, para seguir su progreso.
- Condición: Debe haber comandas registrados (RF-060 o RF-069).

RF-065: Cancelar una comanda

- El camarero elimina una comanda antes o durante su preparación, indicando un motivo como "error del cliente".
- Condición: La comanda debe estar registrado (RF-060 o RF-069) y requiere confirmación.

RF-066: Añadir ítem a una comanda en curso

- El camarero añade un ítem a una comanda en curso.
- Condición: La comanda debe estar registrada (RF-060 o RF-069) y estar en estado 'pendiente' o 'en preparación' (RF-071).

RF-067: Añadir una nota a un ítem de comanda

- El camarero incluye una nota en un ítem de comanda, como "sin sal", para indicar preferencias del cliente antes de prepararlo.
- Condición: La comanda debe estar registrada (RF-060 o RF-069) y no estar en estado 'en preparación'.

RF-068: Cancelar ítem de una comanda en curso

- El camarero cancela un ítem de una comanda en curso y selecciona un motivo.
- Condición: La comanda debe estar registrada (RF-060 o RF-069) y estar en estado 'pendiente' o 'en preparación' (RF-071).

RF-069: Crear comanda en la barra

- El camarero puede crear una comanda para la barra, sin asociarla a una mesa de la zona, para gestionar comandas rápidos o de clientes presentes en la barra.
- Condición: El restaurante debe estar activado (RF-006) y el usuario debe tener rol adecuado (RF-012).

RF-070: Gestionar múltiples comandas activas

- El sistema permite que tanto las mesas como la barra tengan múltiples comandas activas.
- Condición: Deben existir comandas activas (RF-060 o RF-069).

8. PREPARACIÓN DE COMANDAS

Este módulo permite a los cocineros recibir las comandas y marcar cada plato o ítem individual dentro de una comanda como listo a medida que lo terminan. Una vez que todos los ítems están preparados, el cocinero indica que la comanda está completa, notificando al camarero que puede proceder con el servicio o el pago.

RF-071: Empezar la preparación de una comanda

- El cocinero marca una comanda como "en preparación" cuando comienza a trabajar en él, indicando que ya no se puede cambiar.
- Condición: La comanda debe estar registrado (RF-060 o RF-069) y pendiente.

RF-072: Marcar un ítem como listo

- El cocinero indica que un ítem dentro de una comanda, como un plato o bebida, está preparado y listo para servir.
- Condición: La comanda debe estar en preparación (RF-071).

RF-073: Marcar un ítem como rechazado

- El cocinero rechaza un ítem de la comanda si no puede prepararlo, indicando un motivo como "sin ingredientes".
- Condición: La comanda debe estar registrado (RF-060 o RF-069) y pendiente o en preparación, con confirmación.

RF-074: Marcar una comanda como servido

- El camarero marca una comanda como servido lo que activa el acceso a el flujo de pago de este.
- Condición: Todos los ítems de la comanda deben estar marcados como listos (RF-072) o rechazados (RF-073).

RF-075: Completar una comanda

- El sistema marca una comanda como "completado" cuando todos sus ítems están listos, avisando al camarero.
- Condición: Todos los ítems de una comandan deben estar marcados como listos (RF-072) o rechazados (RF-073).

9. GESTIÓN DE NOTIFICACIONES

Este módulo proporciona alertas instantáneas al equipo sobre eventos importantes, como nuevos comandas enviados a la cocina, ítems listos para servir, existencias bajas o mesas reservadas próximas a llegar. Mejora la comunicación interna, asegurando que todos estén informados y puedan actuar a tiempo.

RF-076: Alertar a la cocina de un nuevo comanda

- La cocina recibe una alerta inmediata cuando el camarero registra una comanda, mostrando los detalles para prepararlo.
- Condición: La comanda debe estar registrado (RF-060 o RF-069).

RF-077: Alertar al camarero de un ítem listo

- El camarero recibe una alerta cuando un ítem de una comanda está preparado, para organizar el servicio.
- Condición: El ítem debe estar marcado como listo (RF-072).

RF-078: Alertar al camarero de una comanda completado

- El camarero recibe una alerta cuando una comanda está listo para servirse o pagarse.
- Condición: La comanda debe estar completada (RF-075).

RF-079: Alertar al camarero de una comanda rechazado

- El camarero recibe una alerta cuando la cocina rechaza una comanda, mostrando el motivo para actuar.
- Condición: Un ítem de la comanda debe estar rechazado (RF-073).

RF-080: Alertar al dueño de existencias bajas

- El dueño recibe una alerta cuando un ingrediente o producto preparado cae por debajo de su nivel mínimo.
- Condición: El nivel mínimo debe estar definido (RF-041).

RF-081: Alertar al equipo de una reserva próxima

- El equipo recibe una alerta cuando una reserva está a punto de llegar, indicando la mesa y la hora.
- Condición: La reserva debe estar registrada (RF-027 o RF-028).

10. GESTIÓN DE PAGOS

Una vez completados las comandas, este módulo permite a los clientes pagar sus cuentas directamente desde la aplicación, usando métodos digitales como códigos QR, mientras el dueño puede registrar ingresos en efectivo si los hay. Cada pago se asocia a una comanda, preparando el terreno para el cierre financiero del día.

RF-082: Pagar una comanda digitalmente

- El cliente paga una comanda completa usando un método digital, como un código QR mostrado por el camarero.
- Condición: La comanda debe estar completado (RF-075).

RF-083: Registrar un pago en efectivo

- El camarero o dueño registra un pago en efectivo para una comanda, indicando la cantidad recibida.
- Condición: La comanda debe estar completado (RF-075).

RF-084: Confirmar el pago de una comanda

- La aplicación marca una comanda como "pagada" tras recibir el pago, ya sea digital o en efectivo.
- Condición: El pago debe estar registrado (RF-082 o RF-083).

11. CIERRE DE CAJA

Al final del día, este módulo genera un resumen claro de todos los ingresos, desglosando los pagos digitales y en efectivo, para que el dueño pueda cerrar la caja de forma sencilla. Proporciona una visión general de las finanzas diarias, conectando las comandas completados con el dinero recibido.

RF-085: Generar el resumen diario de caja

- El dueño ve un resumen de todos los pagos recibidos en el día, dividido entre digitales y en efectivo.
- Condición: Debe haber comandas pagadas (RF-084).

RF-086: Ajustar el resumen diario

- El dueño modifica el resumen diario, añadiendo o corrigiendo ingresos como propinas no registradas.
- Condición: El resumen diario debe estar generado (RF-085).

12. GESTIÓN DE INFORMES

Este módulo genera resúmenes detallados para el dueño sobre el rendimiento del negocio, incluyendo comandas completados, consumo de inventario, reservas atendidas y tendencias de ventas. Con estas predicciones y datos, se pueden tomar decisiones informadas, como ajustar menús, planificar compras o redistribuir el personal.

RF-087: Consultar un informe de comandas

- El dueño ve un resumen de las comandas del día, incluyendo completados, rechazados y pendientes.

- Condición: Debe haber comandas registrados (RF-060 o RF-069).

RF-088: Consultar un informe de inventario

- El dueño ve un resumen del consumo de ingredientes, mostrando uso y existencias.
- Condición: Debe haber ingredientes en el inventario (RF-031).

RF-089: Consultar un informe de reservas

- El dueño ve un resumen de reservas atendidas y pendientes para planificar el uso de mesas.
- Condición: Debe haber ingredientes o productos en el inventario (RF-033, RF-034 o RF-035).

RF-090: Consultar un informe de tendencias

- El dueño ve un análisis de ventas y consumo con predicciones sobre compras y ajustes al menú.
- Condición: Debe haber comandas completados (RF-075).

REQUISITOS NO FUNCIONALES

PRESTACIONES

RNF-001: Soportar ejecución en múltiples plataformas

- La aplicación debe ejecutarse sin errores en sistemas operativos móviles y de escritorio, con un diseño responsive que se adapte a pantallas de 320x480 píxeles hasta 1920x1080 píxeles.
- Criterio: Probado en entornos móviles y de escritorio con diferentes resoluciones, asegurando funcionalidad completa.

RNF-002: Garantizar tiempos de respuesta rápidos

- Las operaciones críticas (registro de comandas, actualización de estados, consultas de inventario) deben tener un tiempo de respuesta promedio inferior a 2000 ms bajo carga de 100 usuarios concurrentes y 10,000 registros.
- Criterio: Verificado en pruebas de rendimiento con carga simulada, manteniendo latencia máxima de 2000 ms.

RNF-003: Gestionar alta concurrencia de usuarios

- La aplicación debe soportar hasta 100 usuarios simultáneos ejecutando operaciones críticas con un tiempo de respuesta máximo de 2500 ms.
- Criterio: Evaluado en pruebas con 100 usuarios activos, asegurando estabilidad y respuesta dentro del límite.

RNF-004: Procesar notificaciones en tiempo real

- Las alertas (nuevos comandas, ítems listos, existencias bajas) deben entregarse a los usuarios en menos de 1000 ms tras el evento que las genera.
- Criterio: Medido en pruebas de eventos simulados, confirmando entrega en menos de 1000 ms.

SEGURIDAD

RNF-005: Almacenar contraseñas con cifrado seguro

- Las contraseñas deben almacenarse usando un algoritmo de hash con sal (mínimo 12 rondas), garantizando que no sean recuperables en texto claro.
- Criterio: Comprobado que las contraseñas no se puedan descifrar ni exponer en el sistema.

RNF-006: Gestionar sesiones con autenticación segura

- Las sesiones deben usar tokens seguros con expiración de 24 horas, renovables manualmente, para proteger el acceso de los empleados.
- Criterio: Validado en pruebas de autenticación, asegurando que los tokens expiren tras 24 horas.

RNF-007: Proteger contra accesos no autorizados

- La aplicación debe bloquear el acceso tras 3 intentos fallidos consecutivos, con un tiempo de bloqueo mínimo de 300 segundos.
- Criterio: Confirmado en pruebas de intentos fallidos, verificando activación del bloqueo.

RNF-008: Asegurar la integridad de los pagos digitales

- Las transacciones digitales deben procesarse con cifrado de extremo a extremo, garantizando que los datos no sean interceptados ni alterados.
- Criterio: Evaluado en pruebas de pago simulado, asegurando integridad de los datos.

MANTENIBILIDAD

RNF-009: Registrar eventos del sistema

- La aplicación debe guardar un historial en formato estructurado de errores y acciones clave (login, cambios, pagos), con rotación diaria y retención mínima de 7 días.
- Criterio: Revisado que el historial sea completo, legible y accesible por al menos 7 días.

RNF-010: Facilitar actualizaciones continuas

- La aplicación debe soportar actualizaciones sin interrupción del servicio, con un tiempo de despliegue máximo de 300 segundos.
- Criterio: Probado aplicando cambios en un entorno activo, manteniendo disponibilidad.

USABILIDAD

RNF-011: Garantizar navegación intuitiva según ISO 9241-11

- Las acciones principales (registrar comanda, consultar inventario, gestionar reservas) deben ser accesibles en un máximo de 3 interacciones desde la pantalla inicial.
- Criterio: Validado con usuarios completando tareas en menos de 60 segundos sin asistencia.

RNF-012: Proporcionar retroalimentación inmediata según ISO 9241-11

- Cada acción (enviar comanda, completar comanda) debe mostrar una confirmación visual o sonora en menos de 500 ms tras ejecutarse.
- Criterio: Medido en pruebas, asegurando respuesta en menos de 500 ms.

ACCESIBILIDAD

RNF-013: Cumplir con contraste mínimo según WCAG 2.1 nivel A

- Los textos y elementos interactivos deben tener un contraste mínimo de 4.5:1 contra el fondo para garantizar legibilidad.
- Criterio: Verificado en pruebas de contraste, cumpliendo el estándar WCAG 2.1 nivel A.

RNF-014: Soportar navegación por teclado según WCAG 2.1

- Todas las funciones deben ser operables con teclado (Tab, Enter, flechas), con foco visible en elementos activos.
- Criterio: Evaluado en pruebas manuales, asegurando navegación completa con teclado.

RNF-015: Incluir etiquetas descriptivas según WCAG 2.1

- Los campos y botones deben tener etiquetas legibles por lectores de pantalla, con descripciones claras de su propósito.
- Criterio: Validado en pruebas con lectores de pantalla, confirmando compatibilidad.

ESCALABILIDAD

RNF-016: Adaptarse a diferentes tamaños de negocio

- La aplicación debe soportar desde 5 mesas y 2 empleados hasta 50 mesas y 20 empleados sin degradación del rendimiento.
- Criterio: Probado en configuraciones pequeñas y medianas, manteniendo tiempos de respuesta estables.

RNF-017: Soportar integración con sistemas externos

- La aplicación debe permitir conexiones futuras con sistemas externos mediante una interfaz estándar, procesando hasta 1000 solicitudes por minuto.
- Criterio: Evaluado con una simulación de integración, asegurando capacidad de respuesta.

ROBUSTEZ

RNF-018: Mantener estabilidad bajo fallos parciales

- La aplicación debe seguir funcionando si falla una operación (ej. pago digital), permitiendo uso continuo de otras funciones.
- Criterio: Probado simulando fallos, verificando que el sistema no se detenga completamente.

DESPLIEGUE

RNF-019: Desplegar la API en un entorno cloud accesible públicamente

- La API REST desarrollada con Spring Boot debe estar desplegada en una plataforma de alojamiento en la nube como Render, AWS, Railway o similar, garantizando disponibilidad pública, estabilidad y soporte para HTTPS.
- Criterio: Confirmado el funcionamiento correcto del back-end accediendo desde un cliente remoto mediante HTTPS y realizando operaciones básicas (registro de comanda, consulta de inventario, etc.).

RNF-020: Publicar la app cliente en Google Play Store

- La aplicación móvil desarrollada en Flutter debe ser compilada y publicada en Google Play Store, cumpliendo con los requisitos de distribución de la plataforma y permitiendo su descarga desde dispositivos Android.
- Criterio: Verificado que la app está disponible públicamente en Google Play, accesible desde al menos un dispositivo Android compatible, incluyendo recursos gráficos (icono, capturas, descripción).

RNF-021: Publicar la app cliente para dispositivos iOS (App Store o distribución alternativa)

- La aplicación debe estar disponible para su instalación en dispositivos iOS, ya sea mediante publicación en App Store o utilizando mecanismos alternativos como TestFlight o distribución empresarial, respetando las políticas de Apple.
- Criterio: Verificado el correcto funcionamiento de la app en al menos un dispositivo iOS compatible, incluyendo instalación, acceso y ejecución de funciones clave (pedidos, notificaciones, pagos).

RNF-022: Proporcionar versión ejecutable para sistemas Windows

- La aplicación debe ofrecer un instalador o ejecutable para sistemas operativos Windows (Windows 10 o superior), permitiendo su uso en equipos de escritorio o portátiles.
- Criterio: Confirmado que la versión para Windows se ejecuta correctamente, con interfaz adaptada a pantallas grandes y sin errores críticos en funciones principales.

RNF-023: Proporcionar versión compatible con macOS

- La aplicación debe compilarse y distribuirse para sistemas macOS (mínimo macOS Catalina 10.15), ya sea como app empaquetada (.app) o mediante distribución por notarización/autofirma.
- Criterio: Verificada la ejecución funcional en un entorno macOS compatible, validando estabilidad, diseño responsive y acceso completo a las funcionalidades.

RNF-024: Proporcionar versión instalable para sistemas Linux

- La aplicación debe estar disponible para entornos Linux (preferentemente en formatos .AppImage, .deb o .tar.gz), facilitando la instalación y ejecución en distribuciones comunes como Ubuntu o Fedora.
- Criterio: Confirmado que la aplicación se ejecuta correctamente en un entorno Linux, sin necesidad de configuraciones avanzadas y con acceso total a la interfaz y funcionalidades principales.

RECURSOS Y TECNOLOGÍAS NECESARIAS

A continuación, se detallan las herramientas y tecnologías necesarias para el desarrollo, implementación y mantenimiento de "Gastro & Hub", justificadas según los requisitos funcionales (RF) y no funcionales (RNF) definidos.

DESARROLLO FRONT-END

Flutter

Descripción: Framework de Google para aplicaciones multiplataforma con una sola base de código, usando Dart.

Uso: Interfaz para camareros, cocineros, gerentes y dueño en todos los módulos funcionales.

Dart

Descripción: Lenguaje optimizado para UI, usado por Flutter.

Uso: Lógica y validaciones en la interfaz del cliente.

DESARROLLO BACK-END

Spring Boot

Descripción: Framework de Java para aplicaciones back-end robustas con configuración mínima.

Justificación: Soporta RF-054, RF-009, RF-063, RF-068, RF-060, RNF-009, RNF-006, RNF-008.

Uso: API REST, WebSocket y cálculos predictivos para todos los módulos funcionales.

Java

Descripción: Lenguaje robusto y portátil.

Uso: Desarrollo de reglas de negocio, predicción y procesamiento de datos.

BASE DE DATOS

PostgreSQL

Descripción: Base de datos relacional de código abierto con soporte JSON.

Uso: Persistencia de toda la información del sistema.

COMUNICACIONES

WebSocket (Spring WebSocket)

Descripción: Protocolo y librería para comunicación bidireccional en tiempo real.

Uso: Notificaciones entre camareros, cocineros y dueño.

REST (Spring REST)

Descripción: Arquitectura para APIs basadas en HTTP, implementada en Spring Boot.

Uso: Endpoints de consulta y gestión.

Google API (Google Sign-In)

Descripción: API para autenticación de usuarios mediante cuentas de Google.

Uso: Autenticación simplificada para usuarios y empleados mediante cuentas de Google.

GESTIÓN DE PAGOS

Stripe API

Descripción: API para procesamiento de pagos digitales con soporte para transacciones seguras y códigos QR.

Uso: Integración de pagos digitales desde la aplicación.

GENERACIÓN DE DOCUMENTOS Y CÓDIGOS

iTextPDF

Descripción: Librería Java para generar PDFs dinámicos.

Uso: Creación de documentos de menú compartible.

ZXing

Descripción: Librería para generar y leer códigos QR.

Uso: Generación de códigos QR descargables.

PRUEBAS

JUnit

Descripción: Framework de pruebas unitarias para Java.

Uso: Pruebas unitarias del back-end.

Flutter Test

Descripción: Framework de pruebas integrado en Flutter.

Uso: Pruebas unitarias y de interfaz en front-end.

JMeter

Descripción: Herramienta de Apache para pruebas de carga y rendimiento.

Uso: Pruebas de estrés y rendimiento del sistema.

Testcontainers

Descripción: Librería para pruebas de integración con contenedores Docker.

Uso: Pruebas automatizadas de flujos completos.

CONTROL DE VERSIONES Y COLABORACIÓN

Git

Descripción: Sistema de control de versiones distribuido.

Uso: Repositorio del proyecto.

GitHub

Descripción: Plataforma para alojar repositorios Git.

Uso: Almacenamiento y gestión del proyecto.

GESTIÓN DE PROYECTO

ProjectLibre

Descripción: Software de gestión de proyectos de código abierto con soporte para Gantt charts, planificación Agile y DevOps.

Uso: Planificación, seguimiento y simulación de sprints y entregas del proyecto.

Excel

Descripción: Hoja de cálculo para análisis y seguimiento de métricas de proyecto además de definición de plan de pruebas.

Uso: Gestión de métricas como sprint burnout, planificación de sprints y análisis de datos.

ENTORNO DE DESARROLLO

Visual Studio Code (VS Code)

Descripción: Editor ligero y personalizable para múltiples lenguajes, editor recomendado por Google para el desarrollo con Flutter.

Uso: Editor principal para front-end y back-end.

IntelliJ IDEA Ultimate

Descripción: IDE avanzado para Java y Spring Boot.

Uso: Desarrollo y depuración del back-end.

Android Studio

Descripción: IDE para desarrollo Android y Flutter pero principalmente para la gestión de JDKs y emuladores Android.

Uso: Desarrollo, pruebas del front-end en móviles, emulador y gestor de JDKs.

DESPLIEGUE

Docker

Descripción: Plataforma para contenedores que empaqueta aplicaciones.

Uso: Empaquetado del back-end y base de datos.

Docker Compose

Descripción: Herramienta para definir y ejecutar aplicaciones multicontenedor.

Uso: Configuración de despliegue local.

Servicios en la Nube (AWS, GCP, Azure, Render, Railway)

Descripción: Plataformas para hospedaje de back-end en producción.

Uso: Despliegue del back-end y la base de datos.

Tiendas digitales (Google Play)

Descripción: Plataformas para hospedaje de front-end en producción.

Uso: Despliegue del cliente Android como Android App Bundle.

HARDWARE

PC con Windows

Descripción: Equipo de desarrollo con sistema operativo Windows, con mínimo 8 GB de RAM y procesador de 2 GHz o superior.

Justificación: Compatible con RNF-001 (multiplataforma), RNF-002 (rendimiento), RNF-010 (actualizaciones sin interrupción).

Uso: Desarrollo principal, ejecución de backend, pruebas con Docker, emulador de Android y cliente de escritorio (Windows).

Mac con macOS

Descripción: Equipo físico o entorno virtualizado con macOS Catalina 10.15 o superior, con al menos 8 GB de RAM.

Justificación: Requisito para la compilación y despliegue de aplicaciones iOS (RNF-021, RNF-023).

Uso: Compilación de versiones para iOS y macOS, pruebas nativas en dispositivos Apple, firma y despliegue en App Store o TestFlight.

Dispositivo Android físico

Descripción: Smartphone o tablet con Android 10 o superior.

Justificación: Pruebas en entorno real de cliente móvil, validación de RNF-012 (retroalimentación), RNF-013 (contraste), RNF-014 (teclado).

Uso: Pruebas de experiencia de usuario, rendimiento, notificaciones y pagos.

Dispositivo iOS físico

Descripción: iPhone o iPad con iOS 14 o superior.

Justificación: Validación de compatibilidad real con dispositivos iOS (RNF-021).

Uso: Pruebas funcionales y visuales en entorno nativo iOS.

Servidor cloud

Descripción: Instancia en la nube con mínimo 1 CPU, 1 GB de RAM y 5 GB de almacenamiento.

Justificación: Requisito para el despliegue del backend y base de datos en producción (RNF-019).

Uso: Hospedaje de backend en producción, acceso remoto desde cliente.

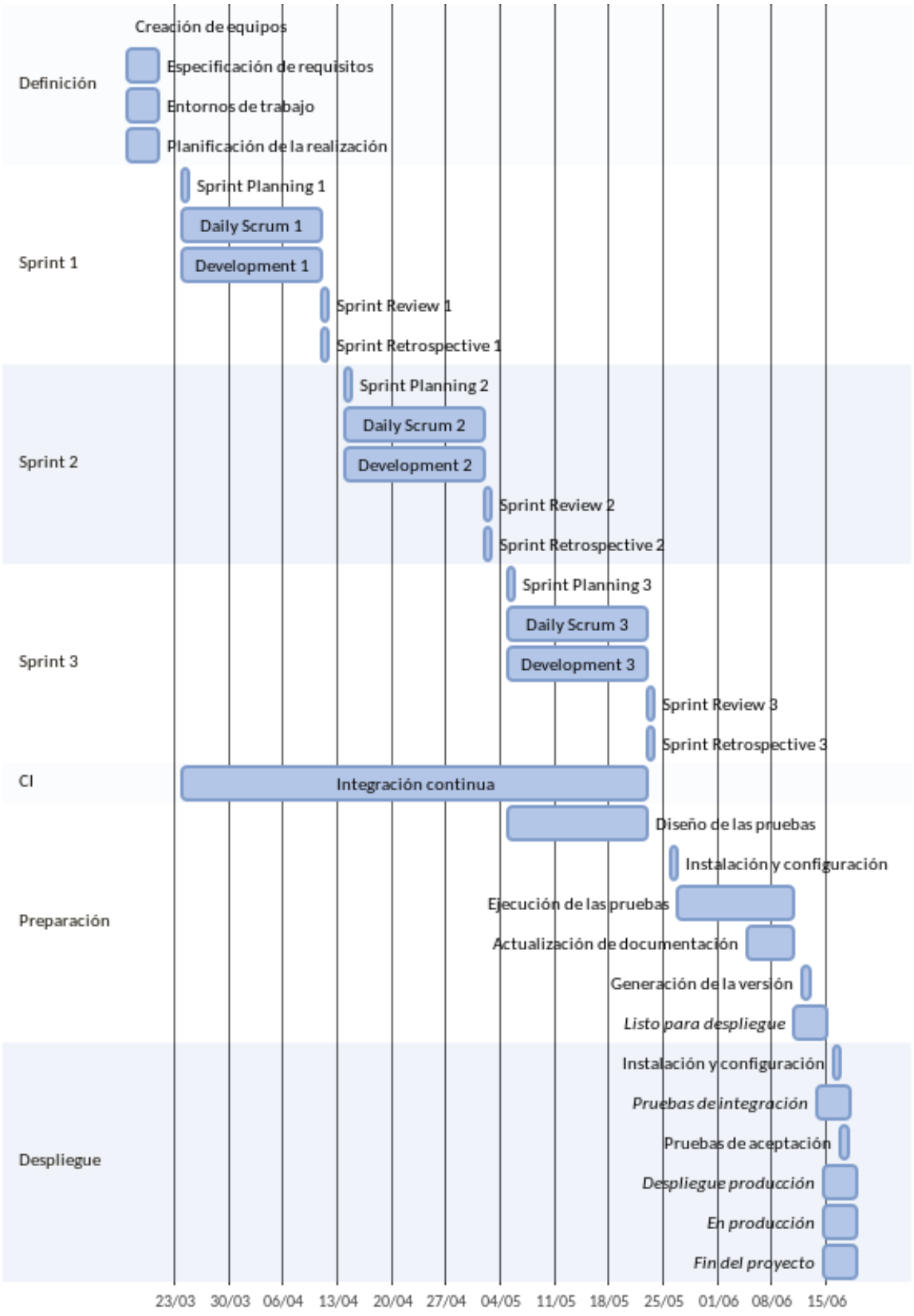
Equipo con Linux

Descripción: Sistema Linux con entorno gráfico, 2 GHz CPU, 4 GB RAM.

Justificación: Validación de RNF-024 (soporte en Linux), pruebas de cliente de escritorio en ese entorno.

Uso: Pruebas de instalación y ejecución de la app cliente en Linux.

PLANIFICACIÓN DE REALIZACIÓN DEL PROYECTO



5. DISEÑO

ARQUITECTURA DEL SISTEMA

La arquitectura de "Gastro & Hub" sigue un modelo cliente-servidor con tres capas (presentación, lógica de negocio, datos), especificando los entornos de desarrollo/pruebas y producción.

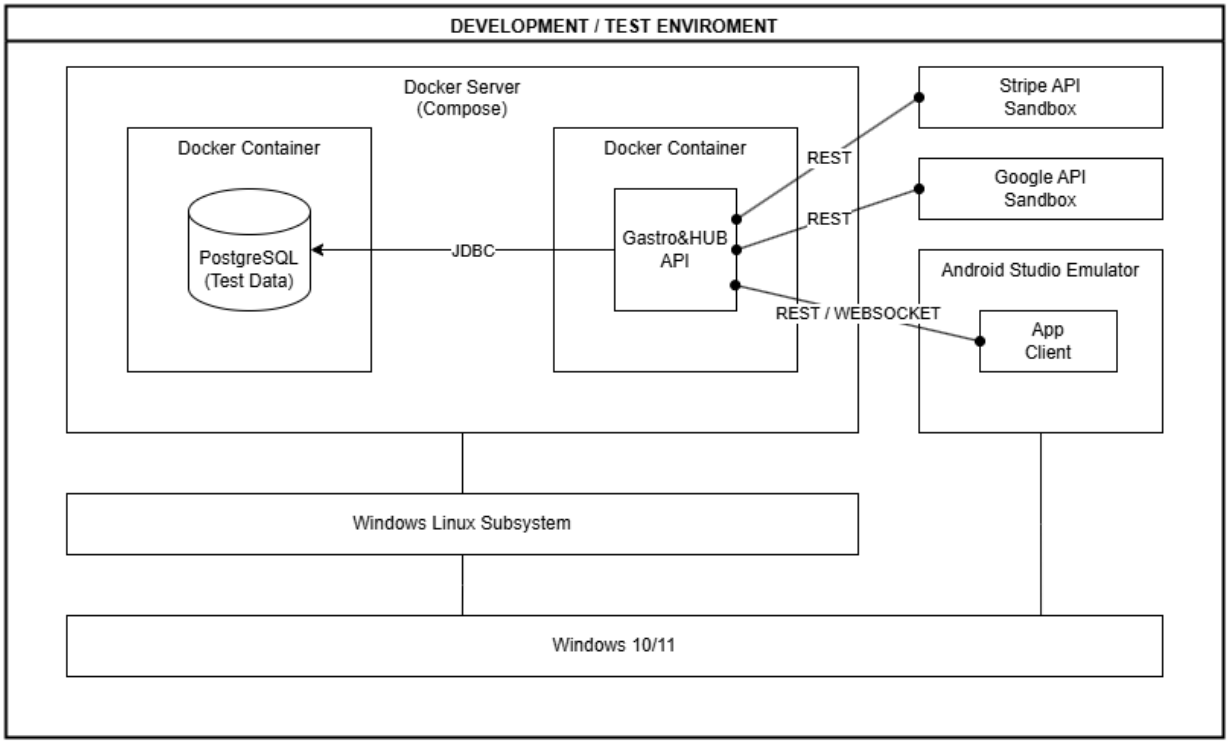
ENTORNO DE DESARROLLO Y PRUEBAS:

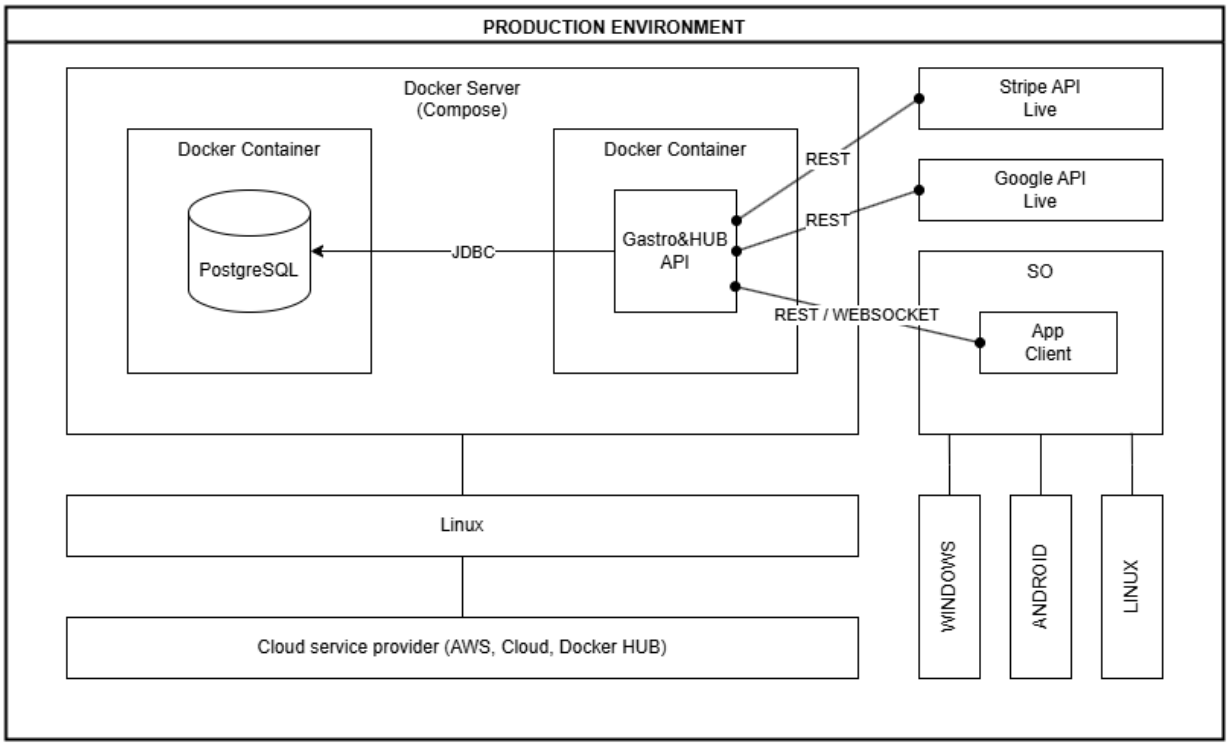
- **Hardware:** PC con Windows, mínimo 8GB RAM y procesador 2GHz, soporta herramientas de desarrollo (RNF-001).
- Capa de presentación: Flutter y Dart ejecutados en emuladores de Android Studio sobre Windows.
- **Capa de lógica de negocio:** Spring Boot y Java en un contenedor Docker, corriendo en un servidor local sobre WSL.
- **Capa de datos:** PostgreSQL en un contenedor Docker, corriendo en el mismo servidor local sobre WSL.
- **Comunicaciones externas:** Stripe API y Google API en modo sandbox, integradas desde el back-end.
- **Despliegue:** Docker y Docker Compose orquestan los contenedores en el servidor local.

ENTORNO DE PRODUCCIÓN:

- **Hardware:** Instancias Linux en servicios en la nube (AWS, GCP o Azure), escalables para 100 usuarios simultáneos (RNF-003, RNF-016).
- **Capa de presentación:** Flutter y Dart ejecutados en sistemas Android/iOS reales, para uso operativo (RF-044, RF-060, RNF-001).
- **Capa de lógica de negocio:** Spring Boot y Java en un contenedor Docker, corriendo en un servidor cloud sobre Linux, gestiona operaciones en vivo.
- **Capa de datos:** PostgreSQL en alta disponibilidad en un contenedor Docker, corriendo en el mismo servidor cloud sobre Linux.
- **Comunicaciones externas:** Stripe API y Google API en modo live, integradas desde el back-end.
- **Despliegue:** Docker y Docker Compose orquestan los contenedores en la nube.

DIAGRAMAS DE ARQUITECTURA SOFTWARE





ARQUITECTURA DE BASE DE DATOS

DIAGRAMA ENTIDAD RELACIÓN (COMPLETO)

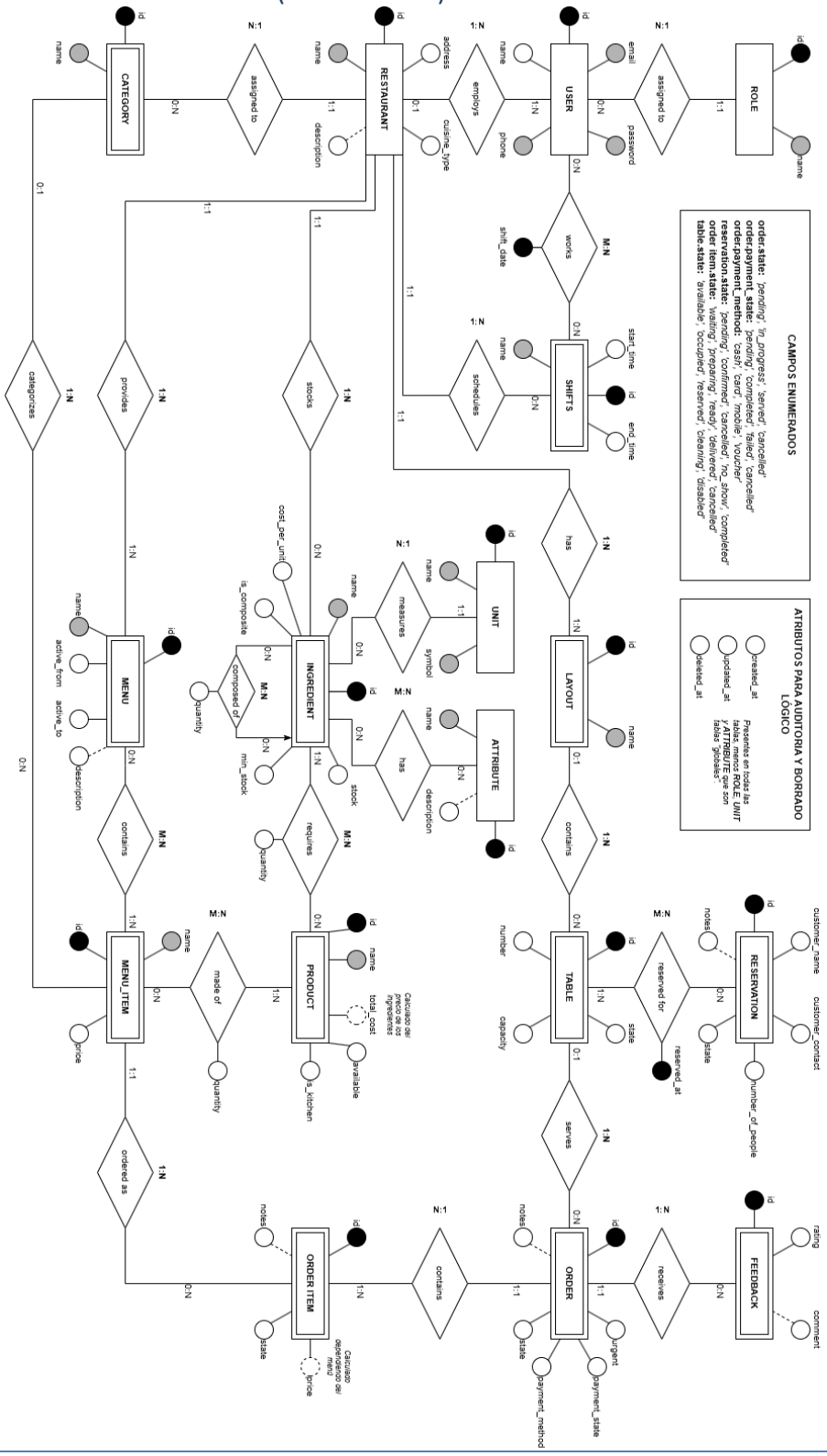


DIAGRAMA ENTIDAD RELACIÓN (IMPLEMENTADO)

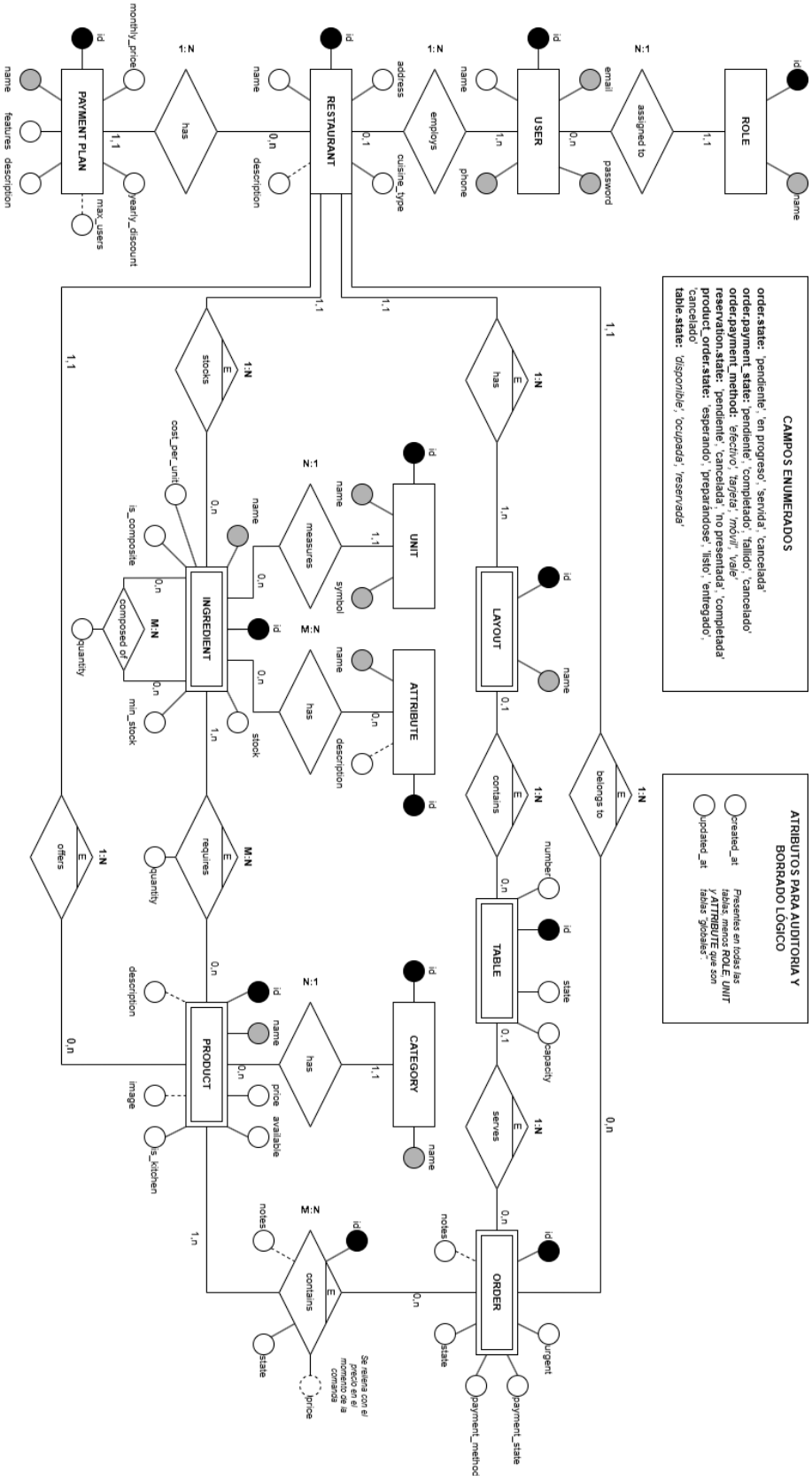


DIAGRAMA LÓGICO RELACIONAL

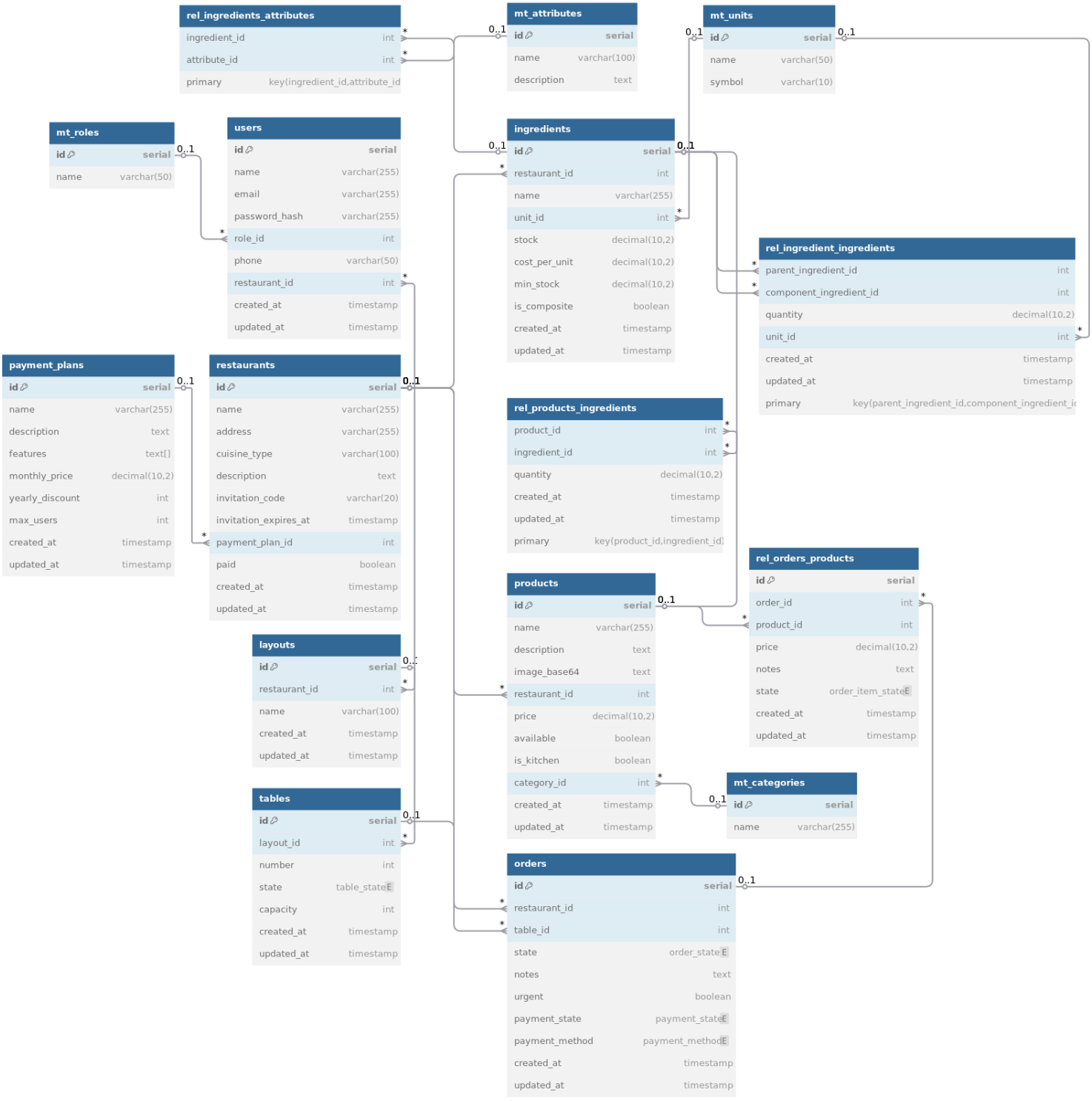


DIAGRAMA DE CLASES

6.IMPLEMENTACIÓN Y PRUEBAS

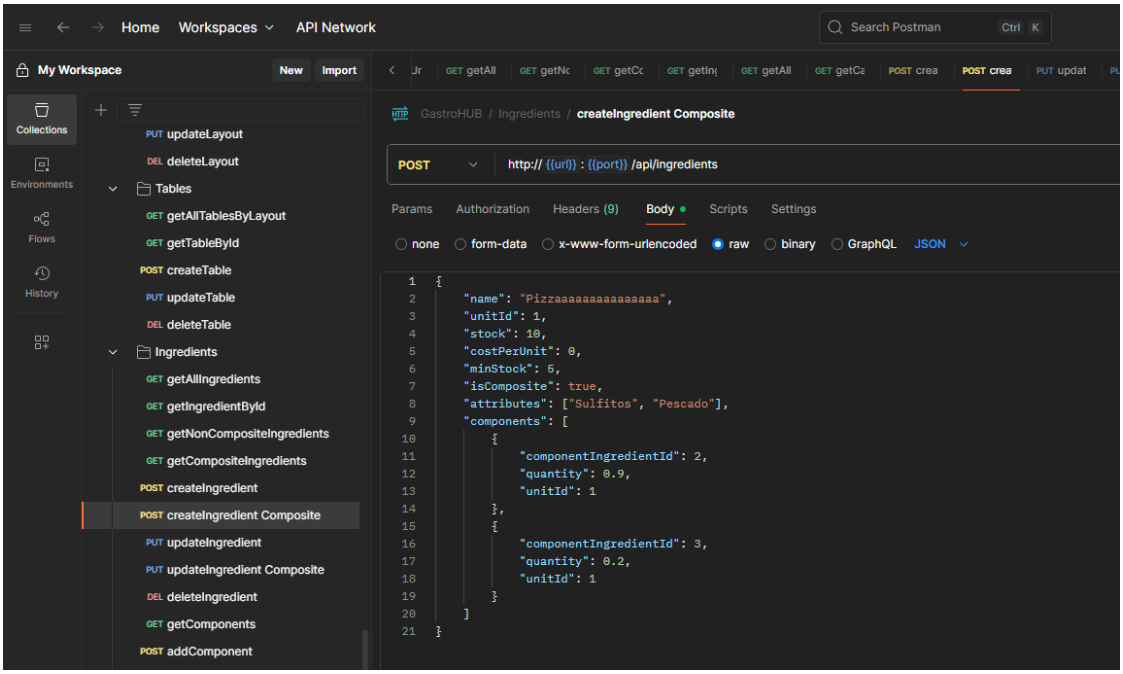
Esta sección describe el proceso de implementación y las estrategias de prueba de "Gastro & Hub", cubriendo el backend y frontend, así como las herramientas empleadas para asegurar su funcionalidad.

PRUEBAS DEL BACKEND

El backend, desarrollado con Spring Boot, fue sometido a pruebas unitarias exhaustivas usando Postman para validar los endpoints de la API REST.

PRUEBAS DEL FRONTEND

El frontend, implementado en Flutter, se probó conectándolo a una API local mediante un emulador Android en Android Studio. Lo que permitió validar la integración cliente-servidor y iterar rápidamente, testeando funciones en tiempo real.



CASOS DE PRUEBA

Los casos de prueba, detallados en el anexo fueron ejecutados exclusivamente en Android, priorizando esta plataforma para estabilidad y rendimiento.

1. Verificar que la pantalla de OnBoarding se muestra al iniciar la aplicación
2. Verificar que un usuario puede crear una cuenta con nombre, email y contraseña válidos

3. Verificar que no se puede registrar un usuario con un email ya en uso
4. Verificar que el sistema rechaza contraseñas que no cumplen con los requisitos de seguridad
5. Verificar que un usuario registrado puede crear un restaurante con un nombre único
6. Verificar que no se puede crear un restaurante con un nombre ya en uso
7. Verificar que el dueño puede seleccionar un plan de pago disponible
8. Verificar que el espacio del restaurante se activa tras confirmar el pago
9. Verificar que el dueño puede ver el código de invitación en texto plano
10. Verificar que el dueño puede ver el código de invitación como QR
11. Verificar que el dueño puede regenerar el código de invitación
12. Verificar que un empleado puede unirse al restaurante con un código válido
13. Verificar que no se puede unir al restaurante con un código inválido
14. Verificar que un empleado puede unirse al restaurante escaneando un QR válido
15. Verificar que no se puede unir al restaurante con un QR inválido
16. Verificar que el dueño puede ver la lista de empleados con sus datos
17. Verificar que el dueño puede asignar un rol a un empleado
18. Verificar que el dueño puede cambiar el rol de un empleado
19. Verificar que el dueño puede expulsar a un empleado
20. Verificar que un empleado puede ver sus propios datos y rol
21. Verificar que un empleado puede editar sus datos personales
22. Verificar que un empleado puede decidir abandonar el restaurante
23. Verificar que el dueño o gerente puede crear una zona
24. Verificar que el dueño o gerente puede editar el nombre de una zona
25. Verificar que el dueño o gerente puede eliminar una zona sin mesas
26. Verificar que no se puede eliminar una zona con mesas activas
27. Verificar que el dueño o gerente puede añadir una mesa con número y capacidad
28. Verificar que el dueño o gerente puede editar el número y capacidad de una mesa
29. Verificar que los empleados pueden ver el plano con el estado de las mesas
30. Verificar que el equipo puede consultar el estado de las mesas
31. Verificar que el dueño puede añadir un ingrediente con nombre, unidad, cantidad, etc.
32. Verificar que el dueño puede añadir un ingrediente compuesto con sus componentes
33. Verificar que el dueño puede añadir un producto con nombre, precio, ingredientes, etc.
34. Verificar que el dueño puede aumentar la cantidad de un ingrediente
35. Verificar que el dueño puede editar los datos de un ingrediente
36. Verificar que el dueño puede editar los datos de un producto
37. Verificar que el dueño puede definir un nivel mínimo para un ingrediente
38. Verificar que el dueño o gerente puede ver la lista de ingredientes con cantidades y estados
39. Verificar que los usuarios pueden acceder a la lista de alérgenos y atributos
40. Verificar que el dueño o gerente puede asociar atributos a un ingrediente
41. Verificar que los camareros pueden ver el menú organizado por categorías
42. Verificar que el sistema asigna atributos a productos según sus ingredientes
43. Verificar que el dueño puede asignar atributos manualmente a un producto
44. Verificar que el gerente puede cambiar la disponibilidad de un ítem
45. Verificar que el camarero puede crear una comanda seleccionando productos y asignándola a una mesa
46. Verificar que no se puede registrar una nueva comanda para una mesa ya ocupada
47. Verificar que el camarero puede añadir una nota a una comanda pendiente
48. Verificar que el camarero puede marcar una comanda como urgente

49. Verificar que los camareros y cocineros pueden ver las comandas actuales
50. Verificar que el camarero puede cancelar una comanda pendiente
51. Verificar que el camarero puede añadir un ítem a una comanda en estado "pendiente" o "en preparación"
52. Verificar que el camarero puede añadir una nota a un ítem específico de una comanda
53. Verificar que el camarero puede cancelar un ítem de una comanda en estado "pendiente" o "en preparación"
54. Verificar que el camarero puede crear una comanda sin asignarla a una mesa
55. Verificar que una mesa puede tener múltiples comandas activas
56. Verificar que el cocinero puede marcar una comanda como "en preparación"
57. Verificar que el cocinero puede marcar un ítem de una comanda como listo
58. Verificar que el cocinero puede rechazar un ítem de una comanda
59. Verificar que el camarero puede marcar una comanda como servida
60. Verificar que el sistema marca una comanda como "completada" cuando todos los ítems están listos o rechazados
61. Verificar que el camarero o dueño puede registrar un pago en efectivo
62. Verificar que la aplicación marca una comanda como "pagada" tras registrar el pago

En resumen, el backend se testeó con Postman, el frontend con un emulador Android conectado a la API local, y los casos de prueba, enfocados en Android, están en el anexo.

7.IMPLANTACIÓN

En este apartado se proporcionan instrucciones paso a paso para desplegar la aplicación "Gastro & Hub" en un entorno local, abarcando tanto el desarrollo como la configuración para pruebas. Abra una terminal y ejecute el siguiente comando para clonar el repositorio: ***git clone https://github.com/AbelMoroEducaMadrid/gastrohub-app***

CONFIGURAR EL FRONTEND (FLUTTER)

INSTALAR VISUAL STUDIO CODE

- Descargue e instale VS Code desde su sitio oficial si aún no lo tiene.

INSTALAR FLUTTER SDK

- Siga las instrucciones de la documentación oficial de Flutter según su sistema operativo (Windows, macOS o Linux). Descargue el SDK y descomprímalo en una ubicación.
- Verifique la instalación ejecutando el comando: ***flutter doctor***
- Resuelva cualquier problema reportado (como dependencias faltantes) siguiendo las sugerencias de flutter doctor.

CONFIGURAR FLUTTER EN VS CODE

- Abra VS Code e instale las extensiones de ***Flutter*** y ***Dart*** después reinicie VS Code para aplicar los cambios.

INSTALAR ANDROID STUDIO Y CONFIGURAR EL EMULADOR

- Descargue e instale Android Studio desde su sitio oficial, asegúrese de incluir el Android SDK y el Android Virtual Device (AVD).
- Abra Android Studio y vaya a Tools > Device Manager y haz clic en Create Virtual Device.
- Seleccione un dispositivo, elija una versión de Android y complete la configuración. Inicie el emulador para verificar que funciona. Gastro & HUB usa la 35.

ABRIR EL PROYECTO EN VS CODE

- Navege a la carpeta frontend del proyecto y ábrala con VS Code, detectará automáticamente el proyecto Flutter y cargará las dependencias.

CONFIGURAR VARIABLES DE ENTORNO

- El frontend requiere variables de entorno para conectarse a la base de datos y otras configuraciones. Copie el archivo ***.env*** como ***.env.local*** en la carpeta frontend y configure las variables de entorno locales.

EJECUTAR LA APLICACIÓN FLUTTER

- Asegúrese de que el emulador de Android esté en ejecución. Abra la terminal integrada en VS Code y ejecute ***flutter pub get*** para instalar las dependencias del proyecto
- Ejecute ***flutter run,*** flutter compilará el código y desplegará la app en el emulador o dispositivo.

CONFIGURAR EL BACKEND (JAVA CON SPRING BOOT)

INSTALAR INTELLIJ IDEA

- Descargue e instale IntelliJ IDEA desde su sitio oficial. La versión Ultimate es ideal por su soporte avanzado para Spring Boot.

INSTALAR JAVA 17

- Descargue **Java 17** e instale el JDK siguiendo las instrucciones para su sistema operativo.

ABRIR EL PROYECTO EN INTELLIJ

- Abra IntelliJ IDEA. File > Open y navegue a la carpeta backend dentro del repositorio clonado (gastro-hub/backend).
- IntelliJ importará el proyecto como un proyecto **Maven** y descargará las dependencias automáticamente.

CONFIGURAR VARIABLES DE ENTORNO

- El backend requiere variables de entorno para conectarse a la base de datos y otras configuraciones. Configure ***aplication.properties*** con las variables locales.

CONSTRUIR Y EJECUTAR EL BACKEND

- Abra la clase principal del proyecto (GastroHubApplication.java) y haga clic en el botón "Run" (triángulo verde) junto a la clase o use Run > Run 'GastroHubApplication'.
- IntelliJ compilará el proyecto y lo ejecutará. Una vez iniciado, el backend estará disponible en http://localhost:8080.

CONFIGURAR LA BASE DE DATOS (POSTGRESQL)

INSTALAR POSTGRESQL

- Descarga postgresql desde su sitio oficial y sigue las instrucciones según tu sistema operativo.

INSTALAR Y ABRIR PGADMIN

- Una vez instalado postgresql, instala pgadmin si no se incluyó en el paquete anterior.

CREAR CONEXIÓN A UN SERVIDOR

- En el panel izquierdo, haz clic derecho sobre "servers" y selecciona "create > server". en la pestaña general, asigna un nombre como "gastrohub db" y rellena los valores restantes.

EJECUTAR LOS SCRIPTS DE BASE DE DATOS

- Los scripts sql necesarios para crear las tablas están disponibles en la carpeta backend/database del repositorio clonado. abre el archivo y ejecútalo desde el editor de consultas de pgadmin o bien desde la terminal con el cliente psql, asegurándote de apuntar a la base de datos gastrohub.

8. RESULTADOS Y DISCUSIÓN

El desarrollo de "Gastro & Hub" se ejecutó siguiendo una metodología ágil basada en sprints, con el objetivo de entregar una aplicación multiplataforma funcional para la gestión hostelera. Aunque la planificación inicial fue ambiciosa, se enfrentaron desafíos técnicos y temporales que requirieron ajustes en el cronograma y las prioridades. A continuación, se detalla el proceso, los retos superados, los logros alcanzados y las oportunidades de mejora identificadas.

TEMPORALIZACIÓN Y DESARROLLO POR SPRINTS

La planificación inicial, documentada en el apartado "Planificación de Realización del Proyecto", estimó duraciones optimistas para cada sprint. Sin embargo, la realidad del desarrollo reveló complejidades imprevistas lo que me llevó al final a una simplificación de la Base de Datos.

Estos retrasos reflejan la necesidad de equilibrar ambición con realismo en la planificación, una lección clave para futuros proyectos.

DIFICULTADES TÉCNICAS Y SOLUCIONES

El desarrollo enfrentó varios obstáculos técnicos que moldearon el resultado final:

- Complejidad de la Base de Datos
- Integración de WebSocket para Notificaciones en Tiempo Real
- Compatibilidad Multiplataforma con Flutter

LOGROS ALCANZADOS

A pesar de los desafíos, "Gastro & Hub" cumplió con sus objetivos principales:

DESPLIEGUE DEL BACKEND EN RENDER

La API REST, desarrollada con Spring Boot, se desplegó exitosamente en Render (RNF-019), garantizando disponibilidad pública y seguridad mediante HTTPS.

FRONTEND MULTIPLATAFORMA PARA ANDROID

El frontend en Flutter se completó para Android, con una interfaz intuitiva, también es relativamente funcional para Windows y Linux.

LANZAMIENTO EN GOOGLE PLAY STORE

Aunque no estaba contemplado inicialmente en los requisitos del proyecto, se logró publicar una versión de prueba interna de "Gastro & Hub" en Google Play Store, compilada como un

Android App Bundle. Esta experiencia enriqueció el proceso de desarrollo al requerir la navegación de la API de Google Play Console, la configuración de requisitos de distribución (e.g., iconos, capturas de pantalla, descripción), y la validación de compatibilidad con dispositivos Android (RNF-020). Este hito, más allá de su carácter exploratorio, demostró la capacidad de llevar la aplicación a un entorno de distribución real, sentando las bases para un lanzamiento público futuro y destacando el aprendizaje en la gestión de plataformas de publicación.

AMPLIACIONES FUTURAS

En el futuro se podrían implementar todos los requisitos que faltan y después pulir la aplicación para sacarla al mercado. Asegurando requisitos de seguridad, escalabilidad, finanzas, etc que la aplicación pueda necesitar en un futuro.

9. CONCLUSIONES

"Gastro & Hub" ha sido un proyecto transformador, tanto en términos técnicos como personales, consolidando las competencias adquiridas en el CFGS de Desarrollo de Aplicaciones Multiplataforma (DAM) y demostrando su aplicabilidad en un problema real del sector hostelero.

APRENDIZAJES TÉCNICOS

DOMINIO DE TECNOLOGÍAS

Spring Boot: Se profundizó en la creación de APIs REST y WebSocket, manejando endpoints complejos como la actualización de orders.state y notificaciones (RF-076).

Flutter: Se optimizó la interfaz para multiplataforma, resolviendo problemas como el renderizado en Linux (RNF-024).

PostgreSQL: El diseño de db-schema.sql aplicó normalización (e.g., tablas maestras mt_*) y tipos enumerados (order_state) para garantizar integridad.

Docker facilitó el despliegue local y en la nube (RNF-019), asegurando consistencia entre entornos. Por ejemplo, los contenedores de Spring Boot y PostgreSQL se configuraron con Docker Compose (página 30), simplificando las pruebas.

IMPACTO EN LA FORMACIÓN

APLICACIÓN DE MÓDULOS DEL DAM

Módulos de programación: La lógica de negocio en Java/Dart aplicó conceptos de estructuras de datos y algoritmos.

Módulos de bases de datos: La normalización en db-schema.sql reflejó principios teóricos, como la eliminación de redundancias.

Módulos de interfaces: La navegación intuitiva en Flutter (RNF-011) integró lecciones de diseño de UI/UX.

CRECIMIENTO PERSONAL

Autonomía
Trabajar independientemente en un proyecto completo, desde el diseño (db-schema.sql) hasta el despliegue (Render), incrementó la confianza y la autogestión.

Preparación Profesional
La experiencia con tecnologías demandadas (Spring Boot, Flutter) y la gestión de un ciclo de desarrollo completo preparan para roles como desarrollador full-stack en entornos empresariales.

10. BIBLIOGRAFÍA Y REFERENCIAS

A continuación, se listan las fuentes utilizadas en el desarrollo de "Gastro & Hub," asegurando que sean relevantes y estén correctamente citadas. Se eliminó la referencia ficticia y se agregaron recursos específicos:

- Flutter Dev Team. (2024). *Flutter Documentation*. Recuperado de **https://flutter.dev/docs**
- Spring Boot. Recuperado de **spring.io/projects/spring-boot**
- PostgreSQL Global Development Group. (2024). *PostgreSQL 16 Documentation*. Recuperado de **https://www.postgresql.org/docs/16/**
- Stripe Inc. (2024). *Stripe API Reference*. Recuperado de **https://stripe.com/docs/api**
- Docker Inc. (2024). *Docker Documentation*. Recuperado de **https://docs.docker.com/**