3/30/2025

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).

RESUMEN

"Gastro & Hub" es una aplicación multiplataforma diseñada para optimizar la gestión operativa de bares y restaurantes. En un contexto donde la eficiencia y la coordinación en tiempo real son clave para el sector hostelero, este proyecto aborda el problema de la comunicación lenta y los procesos manuales entre camareros, cocineros y gerentes. La solución propuesta integra comandas en tiempo real mediante WebSocket, gestión básica de inventario y turnos, y un diseño escalable para adaptarse a distintos tipos de negocios. Desarrollada con tecnologías modernas como Spring Boot, Flutter y Docker, busca ofrecer una herramienta accesible y práctica para mejorar la experiencia operativa y la satisfacción del cliente.

INTRODUCCIÓN

Este proyecto se ha desarrollado como parte de la formación del CFGS en Desarrollo de Aplicaciones Multiplataforma, en el marco del Trabajo Final de Grado (TFG). Surge como una iniciativa para aplicar los conocimientos adquiridos durante el ciclo en un caso práctico del sector hostelero, un ámbito relevante en España y con necesidades tecnológicas en evolución.

MÓDULOS IMPLICADOS EN EL DESARROLLO DEL PROYECTO

- **Programación**: Desarrollo del backend con Java y Spring Boot, y del frontend con Dart/Flutter.
- **Bases de Datos**: Diseño y uso de PostgreSQL para gestionar datos de comandas, inventario y usuarios.
- **Sistemas Informáticos**: Configuración de Docker para despliegue local.
- **Desarrollo de Interfaces**: Creación de una interfaz intuitiva en Flutter para múltiples plataformas.
- **Gestión de Proyectos**: Planificación y documentación del TFG.

INTRODUCCIÓN A LA SOLUCIÓN OBJETO DE ESTE PROYECTO

"Gastro & Hub" es una aplicación que centraliza la gestión de bares y restaurantes en un "hub" tecnológico, permitiendo a los dueños y empleados coordinar tareas en tiempo real. Combina un backend robusto (Spring Boot con REST y WebSocket) con un frontend flexible (Flutter) y una base de datos eficiente (PostgreSQL), todo desplegado mediante Docker.

ESTADO DEL ARTE

El mercado hostelero cuenta con soluciones como Toast POS, OpenTable o Square, que ofrecen gestión de comandas, reservas o pagos. Sin embargo, muchas son sistemas propietarios caros, enfocados en grandes cadenas o con limitaciones en tiempo real y personalización. En España, los bares y restaurantes pequeños suelen depender de procesos manuales o software genérico, lo que genera retrasos y errores. "Gastro & Hub" busca llenar este hueco con una app gratuita, escalable y adaptada a las necesidades locales, usando tecnologías abiertas y modernas.

OBJETIVOS DEL PROYECTO

Desarrollar una aplicación multiplataforma para:

- ✓ Facilitar la gestión de comandas en tiempo real entre camareros y cocina.
- ✓ Controlar inventario básico y turnos del personal.
- ✓ Ofrecer una solución accesible y escalable para bares y restaurantes pequeños o medianos.
- ✓ Aplicar tecnologías actuales (Spring Boot, Flutter, WebSocket, Docker) en un caso práctico del CFGS.

¿QUÉ HARÁ LA APLICACIÓN?

"Gastro & Hub" será una herramienta integral para la operativa hostelera, con foco en tiempo real y facilidad de uso. Permitirá a bares y restaurantes gestionar comandas, inventario, turnos y menús de manera eficiente, centralizando la operativa en una solución tecnológica accesible y escalable.

REQUISITOS QUE DEBE CUMPLIR LA APLICACIÓN

REQUISITOS FUNCIONALES

GESTIÓN DE COMANDAS

RF-001: Registrar una comanda desde un dispositivo móvil en la interfaz del camarero.

- Descripción: El camarero selecciona ítems del menú, indica la mesa y añade notas opcionales en una pantalla específica, generando un registro con un ID único.
- Datos: id_comanda (auto-generado), id_usuario (FK), numero_mesa (entero 1-99), lista_items (array de id_item y cantidad), notas (texto, máx. 100 caracteres), timestamp (ISO 8601).
- Condición: La comanda se guarda localmente antes de enviarse.

RF-002: Enviar una comanda registrada a la cocina mediante WebSocket.

- Descripción: Una vez registrada (RF-001), la comanda se transmite instantáneamente a la cocina a través de un canal WebSocket específico.
- Datos: id_comanda, lista_items, numero_mesa, notas, timestamp.
- Condición: Requiere conexión activa; si falla, se reintenta cada 5 segundos.

RF-003: Notificar a la cocina mediante WebSocket sobre una nueva comanda enviada.

- Descripción: El sistema emite una notificación automática al canal WebSocket de la cocina al recibir una comanda (RF-002), mostrando un resumen en la interfaz del cocinero.
- Datos: id_comanda, lista_items (nombres), timestamp.
- Condición: La notificación es visible hasta que el cocinero la reconoce.

RF-004: Enviar una notificación push opcional al cocinero para nuevas comandas.

- Descripción: Además del WebSocket (RF-003), el cocinero recibe una notificación push en su dispositivo si está configurada, con un resumen breve.
- Datos: id_comanda, mensaje ("Nueva comanda #ID").
- Condición: Configurable por usuario en ajustes.

RF-005: Consultar comandas filtradas por estado desde la interfaz.

- Descripción: El usuario (camarero o cocinero) verá una lista de comandas filtradas por estado actual, con opción de filtrar por mesa o fecha.
- Datos: estado (pendiente/en_preparacion/completada/rechazada), numero_mesa (opcional), fecha (rango, YYYY-MM-DD).
- Condición: Devuelve hasta 50 comandas por consulta, ordenadas por timestamp descendente.

RF-006: Responder a consultas de comandas en menos de 1 segundo desde la interfaz.

- Descripción: Al consultar comandas filtradas por estado (RF-005), el sistema devuelve los resultados en la interfaz en menos de 1 segundo para hasta 1,000 comandas activas.
- Datos: estado, numero_mesa (opcional), fecha (opcional), lista_resultados.
- Condición: Medido en red local con latencia < 50ms.

RF-007: Añadir una nota personalizada a una comanda existente desde la interfaz del camarero.

- Descripción: El camarero edita una comanda pendiente para incluir o modificar una nota antes de que la cocina la procese.
- Datos: id_comanda, notas (texto, máx. 100 caracteres), timestamp_modificacion.
- Condición: Solo editable si el estado es "pendiente".

RF-008: Marcar una comanda como urgente desde la interfaz del camarero.

- Descripción: El camarero activa un indicador de prioridad al registrar o editar una comanda, resaltándola en la vista de la cocina.
- Datos: id_comanda, prioridad (booleano, predeterminado false).
- Condición: Máximo 5 comandas urgentes simultáneas por turno.

RF-009: Calcular y mostrar el tiempo estimado de preparación de una comanda.

- Descripción: Al registrar una comanda (RF-001), el sistema suma el tiempo predefinido de cada ítem (en minutos) y lo muestra al camarero.
- Datos: id_comanda, tiempo_estimado (entero en minutos, calculado).
- Ejemplo: "Hamburguesa (10 min) + Refresco (2 min) = 12 min".

RF-010: Actualizar el estado de una comanda a "en preparación" desde la interfaz del cocinero.

- Descripción: El cocinero selecciona una comanda pendiente y la marca como "en preparación" con un botón, actualizando la base de datos.
- Datos: id_comanda, estado (en_preparacion), timestamp_actualizacion.
- Condición: Solo comandas en "pendiente" pueden cambiarse.

RF-011: Actualizar el estado de una comanda a "completada" desde la interfaz del cocinero.

- Descripción: El cocinero marca una comanda como "completada" al finalizarla, actualizando la base de datos y notificando al camarero.
- Datos: id_comanda, estado (completada), timestamp_actualizacion.
- Condición: Solo comandas en "en preparación" pueden cambiarse.

RF-012: Cancelar una comanda desde la interfaz del camarero.

- Descripción: El camarero elimina una comanda, registrando un motivo y solicitando confirmación.
- Datos: id_comanda, motivo_cancelacion (texto, máx. 50 caracteres), timestamp_cancelacion.
- Condición: Requiere confirmación mediante botón "Sí/No".

RF-013: Rechazar una comanda desde la interfaz del cocinero.

- Descripción: El cocinero selecciona una comanda en estado "pendiente" o "en preparación" y la rechaza, registrando un motivo obligatorio en un campo de texto.
- Datos: id_comanda, motivo_rechazo (texto, máx. 50 caracteres), id_usuario (FK del cocinero), timestamp_rechazo (ISO 8601).
- Condición: Requiere confirmación mediante botón "Sí/No" antes de procesar el rechazo.

RF-014: Actualizar el estado de una comanda a "rechazada" en la base de datos.

- Descripción: Tras confirmar el rechazo (RF-013), el sistema cambia el estado de la comanda a "rechazada" y guarda los detalles del rechazo.
- Datos: id_comanda, estado (rechazada), motivo_rechazo, timestamp_rechazo.
- Condición: Solo comandas en "pendiente" o "en preparación" pueden ser rechazadas.

RF-015: Notificar al camarero mediante WebSocket sobre una comanda rechazada.

- Descripción: Una vez rechazada la comanda (RF-014), el sistema envía una notificación automática al canal WebSocket del camarero que la registró, mostrando el motivo del rechazo.
- Datos: id_comanda, motivo_rechazo, timestamp_rechazo.
- Condición: La notificación aparece en la interfaz del camarero hasta que se reconoce manualmente.

RF-016: Enviar una notificación push opcional al camarero sobre una comanda rechazada.

- Descripción: Además de la notificación WebSocket (RF-015), el camarero recibe una notificación push en su dispositivo si está configurada, con un resumen del rechazo.
- Datos: id_comanda, mensaje ("Comanda #ID rechazada: [motivo]").
- Condición: Configurable por usuario en ajustes, enviada en < 2 segundos tras el rechazo.

RF-017: Generar un informe diario de comandas completadas para el dueño.

- Descripción: El sistema compila un informe al final del día con estadísticas de comandas completadas, accesible desde la interfaz del dueño.
- Datos: fecha (YYYY-MM-DD), total_comandas (entero), items_mas_pedidos (lista top 5), ingresos_estimados (decimal).
- Condición: Solo muestra datos de comandas con estado "completada".

GESTIÓN DE INVENTARIO

RF-018: Crear un nuevo producto en el inventario desde la interfaz del dueño.

- Descripción: El dueño registra un producto nuevo con nombre, stock inicial y umbral mínimo en un formulario.
- Datos: id_producto (auto-generado), nombre (texto, máx. 50 caracteres), stock_inicial (entero positivo), umbral_minimo (entero positivo).
- Condición: Nombre único entre productos.

RF-019: Registrar manualmente una entrada de stock en el inventario desde la interfaz del dueño.

- Descripción: El dueño añade una cantidad de un producto al inventario, actualizando el stock actual.

- Datos: id_producto, cantidad (entero positivo), timestamp_entrada.
- Condición: Producto debe existir previamente.

RF-020: Modificar los datos de un producto existente en el inventario desde la interfaz del dueño.

- Descripción: El dueño edita nombre, stock o umbral de un producto ya creado, con confirmación.
- Datos: id_producto, nombre (nuevo), stock (nuevo), umbral_minimo (nuevo), timestamp_modificacion.
- Condición: Stock no puede ser negativo.

RF-021: Eliminar un producto del inventario desde la interfaz del dueño.

- Descripción: El dueño borra un producto del inventario, con confirmación explícita.
- Datos: id_producto, timestamp_eliminacion.
- Condición: Solo productos sin comandas activas asociadas.

RF-022: Consultar el estado actual del inventario desde una pantalla dedicada.

- Descripción: El dueño ve una lista de productos con su stock, filtrable por estado (todos, bajo, agotado) o nombre.
- Datos: id_producto, nombre, stock (entero), estado (calculado: bajo si < umbral, agotado si = 0).
- Condición: Lista paginada, 20 productos por página.

RF-023: Configurar un umbral mínimo de stock para un producto desde la interfaz del dueño.

- Descripción: El dueño define un nivel mínimo para cada producto, usado para alertas.
- Datos: id_producto, umbral_minimo (entero positivo).
- Condición: Umbral no puede superar el stock actual.

RF-024: Restar stock del inventario al completar una comanda.

- Descripción: Al marcar una comanda como "completada" (RF-011), el sistema resta automáticamente la cantidad de cada ítem usado del inventario.
- Datos: id_comanda, lista_items (id_item, cantidad), timestamp_resta.
- Condición: Si el stock llega a 0, se marca como "agotado".

RF-025: Notificar al dueño cuando el stock de un producto cae por debajo del umbral mínimo.

- Descripción: Tras restar stock (RF-024), si el nivel baja del umbral (RF-023), se muestra una alerta en la interfaz del dueño.
- Datos: id_producto, nombre, stock_actual, umbral_minimo.
- Condición: Alerta persistente hasta reabastecimiento o desactivación manual.

GESTIÓN DE TURNOS

RF-026: Crear un turno para un empleado desde la interfaz del gerente.

- Descripción: El gerente asigna un turno a un usuario en un calendario o formulario, especificando fecha y horas.
- Datos: id_turno (auto-generado), id_usuario (FK), fecha (YYYY-MM-DD), hora_inicio (HH:MM), hora_fin (HH:MM)**.**
- Condición: No permite solapamiento con otros turnos del mismo usuario.

RF-027: Modificar un turno existente desde la interfaz del gerente.

- Descripción: El gerente edita la fecha u horas de un turno ya creado, con confirmación.
- Datos: id_turno, fecha (nueva), hora_inicio (nueva), hora_fin (nueva), timestamp_modificacion**.**
- Condición: Solo turnos futuros son editables.

RF-028: Eliminar un turno existente desde la interfaz del gerente.

- Descripción: El gerente borra un turno, con confirmación explícita.
- Datos: id_turno, timestamp_eliminacion**.**
- Condición: Solo turnos futuros pueden eliminarse.

RF-029: Mostrar una vista semanal de turnos por empleado en la interfaz del gerente.

- Descripción: El sistema genera un calendario o lista de turnos asignados para una semana seleccionada, filtrable por usuario.
- Datos: semana (YYYY-WW), id_usuario (opcional), lista_turnos (id_turno, fecha, hora_inicio, hora_fin)**.**
- Condición: Muestra hasta 7 días, ordenados cronológicamente.

GESTIÓN DE MENÚ

RF-030: Crear una categoría para el menú desde la interfaz del dueño.

- Descripción: El dueño añade una categoría para organizar ítems.
- Datos: id_categoria (auto-generado), nombre (texto, máx. 30 caracteres).
- Condición: Nombre único entre categorías.

RF-031: Añadir un nuevo ítem al menú digital desde la interfaz del dueño.

- Descripción: El dueño crea un ítem con nombre, precio y categoría en un formulario.
- Datos: id_item (auto-generado), nombre (texto, máx. 50 caracteres), precio (decimal, 2 decimales), id_categoria (FK).
- Condición: Nombre único dentro de la categoría.

RF-032: Modificar un ítem existente del menú desde la interfaz del dueño.

- Descripción: El dueño edita nombre, precio o categoría de un ítem ya creado.
- Datos: id_item, nombre (nuevo), precio (nuevo), id_categoria (nueva), timestamp_modificacion.
- Condición: No permite duplicados de nombre en la misma categoría.

RF-033: Eliminar un ítem del menú desde la interfaz del dueño.

- Descripción: El dueño borra un ítem, con confirmación explícita.
- Datos: id_item, timestamp_eliminacion.
- Condición: Solo ítems sin comandas activas asociadas.

RF-034: Mostrar una lista desplegable de ítems organizados por categoría al registrar una comanda.

- Descripción: La interfaz del camarero muestra un desplegable con categorías y sus ítems activos para seleccionar al crear una comanda.
- Datos: lista_categorias (id_categoria, nombre), lista_items (id_item, nombre, precio).
- Condición: Solo ítems activos (no eliminados) son visibles.

RF-035: Generar un documento PDF del menú digital desde la API.

- Descripción: La API crea un PDF con todas las categorías e ítems activos del menú, incluyendo nombre, precio y descripción (si existe), al solicitarlo un empleado.
- Datos: fecha_generacion (YYYY-MM-DD), lista_categorias (id_categoria, nombre), lista_items (nombre, precio).
- Condición: Formato A4, fuente legible (ej. Arial 12), generado en < 5 segundos.

RF-036: Proporcionar un enlace público para descargar el PDF del menú desde la interfaz.

- Descripción: La interfaz muestra un enlace único y accesible (ej. URL corta) para descargar el PDF más reciente, visible para todos los empleados autenticados.
- Datos: url_pdf (texto, ej. "http://app.gastrohub/menu.pdf"), timestamp_generacion.
- Condición: Enlace válido por 24 horas o hasta nueva generación.

RF-037: Generar un código QR que apunte al enlace del PDF del menú.

- Descripción: La interfaz genera un QR escaneable que redirige al enlace del PDF (RF-036), accesible para todos los empleados autenticados.
- Datos: url_pdf, imagen_qr (formato PNG, 200x200 px).
- Condición: QR generado junto al PDF, descargable como imagen.

RF-038: Mostrar el enlace y el QR del menú en una pantalla accesible para todos los empleados.

- Descripción: Una sección dedicada en la interfaz (ej. "Compartir Menú") muestra el enlace y el QR actualizados para que cualquier empleado los use o comparta con clientes.
- Datos: url_pdf, imagen_qr.
- Condición: Visible tras login, sin restricciones de rol.

GESTIÓN DE USUARIOS

RF-039: Autenticar a un usuario mediante login desde la interfaz.

- Descripción: El usuario introduce credenciales en una pantalla de login para acceder según su rol.
- Datos: nombre_usuario (texto, máx. 20 caracteres), contraseña (hasheada, min. 8 caracteres).
- Condición: Bloquea tras 3 intentos fallidos por 5 minutos.

RF-040: Crear un nuevo usuario desde la interfaz del dueño.

- Descripción: El dueño registra un empleado con rol, nombre y credenciales iniciales.
- Datos: id_usuario (auto-generado), nombre (texto, máx. 50 caracteres), rol (dueño/camarero/cocinero), nombre_usuario, contraseña_inicial.
- Condición: Nombre de usuario único.

RF-041: Modificar los datos de un usuario existente desde la interfaz del dueño.

- Descripción: El dueño edita nombre, rol o contraseña de un usuario, con confirmación.
- Datos: id_usuario, nombre (nuevo), rol (nuevo), contraseña (nueva), timestamp_modificacion.
- Condición: No permite cambiar el propio rol del dueño.

RF-042: Eliminar un usuario existente desde la interfaz del dueño.

- Descripción: El dueño da de baja un usuario, con confirmación explícita.
- Datos: id_usuario, timestamp_eliminacion.
- Condición: No permite eliminar al propio usuario dueño.

GESTIÓN OPERATIVA

RF-043: Registrar automáticamente una acción clave en el historial de la base de datos.

- Descripción: Cada acción definida (crear comanda, cambiar estado, modificar inventario, etc.) genera un registro en la tabla de historial con detalles.

- Datos: id_accion (auto-generado), id_usuario (FK), tipo_accion (enum: crear_comanda, cambiar_estado, etc.), timestamp (ISO 8601), detalles (JSON con datos específicos, ej. {"id_comanda": 123}).
- Condición: Registro obligatorio para todas las acciones CRUD.

RF-044: Consultar el historial de acciones desde la interfaz del dueño.

- Descripción: El dueño ve una lista de acciones registradas, filtrable por usuario, tipo de acción o fecha.
- Datos: id_accion, id_usuario, tipo_accion, timestamp, detalles.
- Condición: Lista paginada, 50 acciones por página, ordenada por timestamp descendente.

REQUISITOS NO FUNCIONALES

RNF-001: Soportar ejecución en Android con Flutter.

- Descripción: La app debe compilar y ejecutarse sin errores en Android (mínimo API 21), con diseño responsive que ajuste elementos al tamaño de pantalla.
- Criterio: Probado en emulador Android Studio.

RNF-002: Soportar ejecución en iOS con Flutter.

- Descripción: La app debe compilar y ejecutarse sin errores en iOS (mínimo 12.0), con diseño responsive.
- Criterio: Probado en simulador Xcode.

RNF-003: Soportar ejecución en Windows con Flutter.

- Descripción: La app debe compilar y ejecutarse sin errores en Windows (mínimo 10), con diseño responsive.
- Criterio: Probado en Windows 10/11.

RNF-004: Soportar ejecución en Linux con Flutter.

- Descripción: La app debe compilar y ejecutarse sin errores en Linux (Ubuntu 20.04+), con diseño responsive.
- Criterio: Probado en Ubuntu 20.04+.

RNF-005: Garantizar tiempos de respuesta rápidos para consultas frecuentes.

- Descripción: El sistema debe garantizar que todas las operaciones críticas (registro de comandas, actualización de estados, consultas de inventario) tengan un tiempo de respuesta promedio inferior a 2 segundos bajo carga normal (100 usuarios concurrentes, 10,000 registros).
- Criterio: Medido con Postman en un entorno local (CPU 2GHz, 8GB RAM).

RNF-006: Almacenar contraseñas con cifrado seguro.

- Descripción: Las contraseñas de los usuarios deben almacenarse hasheadas utilizando BCrypt con una sal de 12 rondas para garantizar su seguridad.
- Criterio: No se exponen contraseñas en texto plano en la base de datos ni en logs.

RNF-007: Gestionar sesiones con tokens seguros.

- Descripción: Las sesiones de usuario deben usar tokens JWT con una expiración de 24 horas para autenticación segura en la aplicación.
- Criterio: Los tokens no exponen datos sensibles en respuestas API y son validados en cada solicitud.

RNF-008: Registrar logs de errores y acciones clave en el backend.

- Descripción: El backend genera logs en formato JSON para errores (excepciones) y acciones (login, envío de comanda) en un archivo app.log, con nivel DEBUG y rotación diaria.
- Criterio: Implementado con Logback en Spring Boot.

RNF-009: Garantizar una navegación intuitiva en la interfaz basada en ISO 9241-11.

- Descripción: Las acciones principales (registrar comanda, consultar inventario, ver turnos) deben ser accesibles en un máximo de 3 clics o toques desde la pantalla principal, con etiquetas claras y consistentes (ej. "Nueva Comanda", "Inventario").
- Criterio: Validado mediante pruebas con usuarios reales que completen tareas sin asistencia en < 1 minuto.

RNF-010: Proporcionar retroalimentación inmediata al usuario según ISO 9241-11.

- Descripción: Cada acción (enviar comanda, actualizar estado, guardar cambios) debe mostrar una confirmación visual o sonora (ej. "Comanda enviada") en menos de 0.5 segundos tras ejecutarse.
- Criterio: Verificado en pruebas de sistema con temporizador.

RNF-011: Cumplir con el nivel A de WCAG 2.1 para contraste de color (ISO/IEC 40500).

- Descripción: Los textos y elementos interactivos (botones, listas) deben tener un contraste mínimo de 4.5:1 contra el fondo (ej. texto negro sobre fondo blanco), asegurando legibilidad para usuarios con visión reducida.
- Criterio: Comprobado con WebAIM Contrast Checker.

RNF-012: Soportar navegación por teclado según WCAG 2.1 (ISO/IEC 40500).

- Descripción: Todas las funcionalidades (login, registro de comandas, gestión de menú) deben ser operables solo con teclado (teclas Tab, Enter, flechas), con un foco visible en los elementos activos.
- Criterio: Probado en Flutter con emuladores en cada SO soportado.

RNF-013: Incluir etiquetas descriptivas para accesibilidad conforme a WCAG 2.1 (ISO/IEC 40500).

- Descripción: Los campos de formulario (ej. usuario, contraseña) y botones deben tener etiquetas legibles por lectores de pantalla (ej. "Campo de usuario", "Botón Enviar"), compatibles con herramientas como TalkBack (Android) o VoiceOver (iOS).
- Criterio: Validado con un lector de pantalla en al menos dos plataformas.

HISTORIAS DE USUARIO

A continuación, se presentan las historias de usuario que describen las necesidades de los usuarios de "Gastro & Hub", vinculadas a los requisitos definidos.

HU	Descripción	RF	RNF
HU-001	"Como camarero, quiero registrar una comanda desde mi móvil y enviarla a la cocina rápidamente, para que el pedido se procese sin demoras."	RF-001, RF-002, RF-003, RF-004, RF-034	RNF-001, RNF-005, RNF-009, RNF-010
HU-002	"Como camarero, quiero añadir notas a una comanda y marcarla como urgente si es necesario, para especificar preferencias del cliente como 'sin sal' o atender pedidos prioritarios."	RF-007, RF-008	RNF-005, RNF-009, RNF-010
HU-003	"Como camarero, quiero ver el tiempo estimado de preparación de una comanda, para informar al cliente sobre el tiempo de espera."	RF-009	RNF-005, RNF-009, RNF-010
HU-004	"Como camarero, quiero consultar comandas activas filtradas por estado, para verificar su progreso y atender a los clientes."	RF-005, RF-006	RNF-005, RNF-009, RNF-010
HU-005	"Como camarero, quiero cancelar una comanda si hay un error, para corregirlo antes o durante su procesamiento."	RF-012	RNF-005, RNF-009, RNF-010
HU-006	"Como camarero, quiero recibir notificaciones cuando una comanda es rechazada por la cocina, para informar al cliente y tomar medidas."	RF-015, RF-016	RNF-005, RNF-009, RNF-010

HU-007	"Como cocinero, quiero ver las comandas pendientes y urgentes en una lista clara, para priorizar mi trabajo en la cocina."	RF-005, RF-006, RF-008	RNF-005, RNF-009, RNF-010
HU-008	"Como cocinero, quiero recibir notificaciones inmediatas de nuevas comandas, para no perderme ningún pedido incluso si estoy ocupado."	RF-003, RF-004	RNF-005, RNF-009, RNF-010
HU-009	"Como cocinero, quiero marcar una comanda como en preparación o completada, para que el camarero sepa su estado actual."	RF-010, RF-011	RNF-005, RNF-009, RNF-010
HU-010	"Como cocinero, quiero rechazar una comanda si no puedo prepararla, para notificar al camarero con un motivo claro."	RF-013, RF-014, RF-015, RF-016	RNF-005, RNF-009, RNF-010
HU-011	"Como gerente, quiero asignar y gestionar turnos del personal en un calendario, para organizar el horario semanal de forma eficiente."	RF-026, RF-027, RF-028, RF-029	RNF-005, RNF-009, RNF-010
HU-012	"Como gerente, quiero registrar y gestionar usuarios del equipo, para controlar el acceso al sistema según sus roles."	RF-039, RF-040, RF-041, RF-042	RNF-006, RNF-007, RNF-009, RNF-010
HU-013	"Como dueño, quiero crear y editar productos en el inventario, para mantener un control completo de los recursos disponibles."	RF-018, RF-019, RF-020, RF-021	RNF-005, RNF-009, RNF-010
HU-014	"Como dueño, quiero consultar el inventario y configurar alertas de stock bajo, para reabastecer a tiempo y evitar faltantes."	RF-022, RF-023, RF-024, RF-025	RNF-005, RNF-009, RNF-010
HU-015	"Como dueño, quiero editar el menú digital y organizarlo por categorías, para mantenerlo actualizado con los platos disponibles."	RF-030, RF-031, RF-032, RF-033	RNF-005, RNF-009, RNF-010
HU-016	"Como dueño, quiero generar un PDF del menú y compartirlo con un enlace o QR, para que los empleados lo distribuyan a los clientes fácilmente."	RF-035, RF-036, RF-037, RF-038	RNF-005, RNF-009, RNF-010

HU-017	"Como dueño, quiero ver un resumen diario de comandas completadas, para analizar el rendimiento del restaurante."	RF-017	RNF-005, RNF-009, RNF-010
HU-018	"Como dueño, quiero consultar el historial de acciones del sistema, para auditar cambios y detectar problemas."	RF-043, RF-044	RNF-005, RNF-008, RNF-009, RNF-010

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 FRONTEND

Flutter

- *Descripción*: Framework de Google para aplicaciones multiplataforma con una sola base de código, usando Dart.
- *Justificación*: Cumple RNF-001 (Android), RNF-003 (Windows) y RNF-004 (Linux), con diseño responsive. Soporta RF-001 (registro de comandas desde móvil) y RF-034 (lista desplegable del menú).
- *Uso*: Interfaz para camareros, cocineros, gerente y dueño (HU-001 a HU-018).

Dart

- *Descripción*: Lenguaje optimizado para UI, usado por Flutter.
- *Justificación*: Implementa lógica frontend como RF-009 (cálculo de tiempo estimado) y RF-037 (generación de QR), asegurando RNF-009 (navegación intuitiva).
- *Uso*: Lógica y validaciones en el cliente.

DESARROLLO BACKEND

Spring Boot

- *Descripción*: Framework de Java para aplicaciones backend robustas con configuración mínima.
- *Justificación*: Soporta RF-002 (WebSocket), RF-043 (historial) y RNF-008 (logs con Logback). Facilita RNF-006 (BCrypt) y RNF-007 (JWT).

- *Uso*: API REST y WebSocket para comandas, inventario, turnos, menú y usuarios.

Java

- *Descripción*: Lenguaje robusto y portátil.
- *Justificación*: Base de Spring Boot, ideal para RF-017 (informe diario) y RF-035 (PDF), compatible con RNF-005 (respuesta rápida).
- *Uso*: Backend y reglas de negocio.

BASE DE DATOS

PostgreSQL

- *Descripción*: Base de datos relacional de código abierto con soporte JSON.
- *Justificación*: Almacena comandas (RF-001), inventario (RF-018), turnos (RF-026), menú (RF-030) y usuarios (RF-039). Soporta RF-043 (historial JSON) y RNF-006 (contraseñas hasheadas).
- *Uso*: Persistencia de datos.

COMUNICACIONES

WebSocket (Spring WebSocket)

- *Descripción*: Protocolo y librería para comunicación bidireccional en tiempo real.
- *Justificación*: Cumple RF-002, RF-003 (notificaciones a cocina) y RF-015 (notificaciones de rechazo), asegurando RNF-005.
- *Uso*: Notificaciones entre camareros y cocineros.

REST

- *Descripción*: Arquitectura para APIs basadas en HTTP.
- *Justificación*: Soporta RF-035 (PDF del menú) y consultas como RF-005, alineado con RNF-005.
- *Uso*: Endpoints de consulta y gestión.

GENERACIÓN DE DOCUMENTOS Y CÓDIGOS

iTextPDF

- *Descripción*: Librería Java para generar PDFs dinámicos.
- *Justificación*: Implementa RF-035 (PDF del menú) con formato A4, vinculado a HU-016.
- *Uso*: Creación del PDF del menú.

ZXing

- *Descripción*: Librería para generar y leer códigos QR.
- *Justificación*: Soporta RF-037 (QR del menú) en PNG, usado en HU-016.
- *Uso*: Generación de QR descargable.

PRUEBAS

JUnit

- *Descripción*: Framework de pruebas unitarias para Java.
- *Justificación*: Verifica RF-009 (cálculo de tiempo) y RF-014 (estado rechazada).
- *Uso*: Pruebas unitarias del backend.

Flutter Test

- *Descripción*: Framework de pruebas para Flutter.
- *Justificación*: Valida RF-005 (consulta de comandas) y RNF-009 (navegación intuitiva).
- *Uso*: Pruebas unitarias y de widgets en frontend.

Postman

- *Descripción*: Herramienta para probar APIs REST y WebSocket.
- *Justificación*: Mide RNF-005 (< 2s) y prueba RF-002, RF-035.
- *Uso*: Pruebas manuales de APIs.

JMeter

- *Descripción*: Herramienta de Apache para pruebas de carga y rendimiento.
- *Justificación*: Valida RNF-005 (respuesta < 2s bajo carga de 100 usuarios) y RF-006 (< 1s), simulando uso concurrente en RF-002 (WebSocket).
- *Uso*: Pruebas de estrés y rendimiento del sistema.

Testcontainers

- *Descripción*: Librería para pruebas de integración con contenedores Docker.
- *Justificación*: Verifica integración de RF-001 (registro), RF-002 (envío) y RF-043 (historial) con PostgreSQL en un entorno realista.
- *Uso*: Pruebas automatizadas de flujos completos.

CONTROL DE VERSIONES Y COLABORACIÓN

Git

- *Descripción*: Sistema de control de versiones distribuido.
- *Justificación*: Gestiona el código y documentación del equipo.
- *Uso*: Repositorio del proyecto.

GitHub

- *Descripción*: Plataforma para alojar repositorios Git.
- *Justificación*: Hospeda el código.
- *Uso*: Almacenamiento y documentación.

ENTORNO DE DESARROLLO

Visual Studio Code (VS Code)

- *Descripción*: Editor ligero y personalizable para múltiples lenguajes.
- *Justificación*: Soporta Dart/Flutter y Java con extensiones, útil para edición general y depuración junto a RF-001 y RF-002.
- *Uso*: Editor principal para código y documentación.

IntelliJ IDEA Ultimate

- *Descripción*: IDE avanzado para Java y Spring Boot.
- *Justificación*: Desarrollo backend (RF-002, RF-043) y configuración de RNF-008 (logs).
- *Uso*: Desarrollo y depuración del backend.

Android Studio

- *Descripción*: IDE para desarrollo Android y Flutter.
- *Justificación*: Emulador para RNF-001 (Android) y herramientas para RF-001, RF-034.
- *Uso*: Desarrollo y pruebas del frontend en Android.

DESPLIEGUE

Docker

- *Descripción*: Plataforma para contenedores que empaqueta aplicaciones.
- *Justificación*: Consistencia para PostgreSQL y Spring Boot, soportando RNF-005.
- *Uso*: Empaquetado del backend y base de datos.

Docker Compose

- *Descripción*: Herramienta para definir y ejecutar aplicaciones multicontenedor.
- *Justificación*: Orquesta backend, base de datos y comunicaciones en local/production, alineado con RF-002 y RNF-005.
- *Uso*: Configuración de despliegue local.

Servicios en la Nube (AWS, GCP, Azure, etc.)

- *Descripción*: Plataformas para hospedaje en producción.
- *Justificación*: Posible escalado futuro para RF-002 (WebSocket) y RNF-005 (respuesta rápida).
- *Uso*: Despliegue en producción (opcional).

ACCESIBILIDAD Y USABILIDAD

WebAIM Contrast Checker

- *Descripción*: Herramienta para verificar contraste de colores.
- *Justificación*: Cumple RNF-011 (WCAG 2.1 nivel A).
- *Uso*: Validación de diseño accesible.

TalkBack

- *Descripción*: Lector de pantalla nativo de Android.
- *Justificación*: Verifica RNF-013 (etiquetas descriptivas) en Flutter.
- *Uso*: Pruebas de accesibilidad en Android.

DOCUMENTACIÓN Y GESTIÓN

Draw.io

- *Descripción*: Herramienta online para crear diagramas.
- *Justificación*: Documentación visual de arquitectura, flujos y modelos de datos (RF-001 a RF-044).
- *Uso*: Diagramas UML y de Bases de Datos.

Microsoft Word

- *Descripción*: Procesador de texto para documentación formal.
- *Justificación*: Redacción de informes y documentación externa vinculada a HU-017 (resumen diario).
- *Uso*: Documentos oficiales.

Trello

- *Descripción*: Herramienta de gestión de tareas con tableros Kanban.
- *Justificación*: Organiza HU-001 a HU-018 y seguimiento de requisitos.
- *Uso*: Planificación y seguimiento.

HARDWARE

PC con Windows

- *Descripción*: Equipo de desarrollo con mínimo 8GB RAM y procesador decente (ej. 2GHz).
- *Justificación*: Ejecuta Docker, IntelliJ, Android Studio y VS Code, compatible con RNF-003 (Windows).
- *Uso*: Desarrollo, pruebas y despliegue local.

7008503 2F ABEL MORO
Digitally signed by 70085032F ABEL MORO
Date: 2025.03.30 22:19:46 +02'00'