PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

PLAN DE TRABAJO DE TITULACIÓN

FACULTAD:
HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

CARRERA:
INGENIERIA EN SISTEMAS DE LA INFORMACIÓN

TEMA:
DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB PARA OPTIMIZAR LA COMUNICACIÓN ENTRE CAJA Y COCINA AL RECIBIR PEDIDOS PARA LLEVAR EN EL RESTAURANTE SANTA GULA

AUTOR:
DIEGO JAVIER ROSALES AÑAZCO

FECHA:
QUITO DM, ENERO DE 2026

Dedicatoria

Dedico mi trabajo en primer lugar a Dios, a mis padres y mi hermano quienes con su amor han sido un pilar fundamental de apoyo y motivación constante e incondicional a lo largo de toda mi formación académica y de mi vida, gracias a su esfuerzo comprensión y confianza me ha sido posible superar cada etapa de mi vida y mantener la determinación necesaria para alcanzar mis metas.

Así mismo a toda mi familia quienes han representado un apoyo permanente y un impulso para avanzar en cada logro alcanzado a lo largo de mi vida.

Agradecimiento

A mi familia, quienes han sido el pilar fundamental a lo largo de todo este proceso. Su comprensión y acompañamiento incondicional han sido esenciales para superar cada reto académico y personal. Por la motivación diaria, la confianza depositada en mí y por estar siempre presentes en cada etapa de mi formación.

Además, Agradezco a todos los profesores que formaron parte de mi formación académica por transmitir sus conocimientos y experiencia, especialmente a mi tutor por la orientación constante y el acompañamiento dado durante el desarrollo del trabajo.

Igualmente, agradezco a mis amigos por el apoyo ofrecido y por las experiencias compartidas a lo largo de este camino.

Resumen

El objetivo del proyecto es crear un prototipo de aplicación web con el fin de identificar una solución tecnológica que mejore la gestión interna de los pedidos para llevar en el restaurante Santa Gula. Se evalúa la factibilidad técnica de implementar un sistema que no solo organice adecuadamente los pedidos, sino que también optimice el intercambio de información entre el personal.

Santa Gula maneja sus procesos de forma tradicional, lo cual suele derivar en errores de comunicación, retrasos en la cocina y cierta desorganización al atender las solicitudes. Ante esta realidad, se necesita plantear una herramienta que maneje el flujo de pedidos de manera más eficiente y ordenada, procurando no alterar la dinámica de trabajo del restaurante.

Para el desarrollo del prototipo se utilizó la metodología Scrum, lo que facilitó organizar el trabajo en sprints y avanzar paso a paso desde la planificación hasta la validación del sistema. Se utilizó Node.js para el backend, SQL Server para la base de datos y Angular en el frontend, todo bajo una arquitectura Modelo-Vista-Controlador. Tras realizar las pruebas de interfaz y de APIs. Se concluye que el prototipo cumple su objetivo, demuestra la viabilidad técnica de la propuesta y deja una base funcional sólida para futuras modificaciones.

Índice de Contenidos

Índice de Tablas

Índice de Ilustraciones

CAPÍTULO 1 - PLANTEAMIENTO

1. TEMA

Desarrollo de un prototipo de aplicación web para optimizar la comunicación entre caja y cocina al recibir pedidos para llevar en el restaurante Santa Gula.

2. JUSTIFICACIÓN

El restaurante Santa Gula destaca por su dedicación al servir comida de alta calidad. Como la mayoría de los restaurantes, tiene dificultades para optimizar la eficiencia de sus operaciones internas, sobre todo en la comunicación entre caja y cocina. Actualmente, la información de los pedidos para llevar se transmite manualmente, mediante notas en papel o boca a boca. Si bien ambas opciones han sido útiles, presentan numerosos inconvenientes que afectan el flujo de servicio y, por ende, la satisfacción en el cliente. Las notas en papel pueden ser ilegibles y la comunicación verbal se presta a malentendidos, lo que resulta ocasionalmente en errores en los pedidos y retrasos en la preparación.

El sistema en línea garantizará que todos los pedidos se registren en línea desde el momento de la compra hasta que llegan a la cocina. El uso del sistema evitará los problemas ocasionados por el manejo manual de la información. Este sistema optimizará el uso del tiempo y garantizará una adecuada coordinación de las actividades en el restaurante.

La automatización de los procesos internos del negocio aumentará la calidad del servicio, la eficiencia, y directamente a la satisfacción del cliente, lo que contribuirá al éxito general del restaurante Santa Gula.

3. PLANTEAMIENTO DEL PROBLEMA

En el restaurante Santa Gula, la comunicación deficiente entre caja y el personal de cocina es un problema importante de eficiencia en la prestación del servicio. Actualmente, los pedidos para llevar se reciben manualmente y se pasan a cocina mediante notas impresas o de boca a boca. Si bien esto era eficiente en sus inicios, se ha vuelto ineficaz debido a su crecimiento, especialmente en horas punta. Las notas manuscritas pueden no ser claras y, con frecuencia, resultan en errores al transmitir los pedidos, lo que resulta en elaborar platos incorrectos o mal preparados, un factor que afecta en la experiencia del cliente y el prestigio del negocio. La comunicación verbal también puede ser engañosa y provocar malas interpretaciones, lo que agrava la falta de coordinación entre el personal de caja y el de cocina.

Además de los errores en los pedidos para llevar, este sistema manual provoca graves retrasos en la preparación de los pedidos. Los retrasos en el servicio pueden generar insatisfacción, lo que perjudica la reputación y la fidelización del cliente.

Esto también demuestra la falta de un sistema tecnológico mediante el cual la comunicación dentro de la empresa puede ser tratada de una manera ágil y precisa, lo que produce una gran necesidad de mejoramiento en los procesos de coordinaciones entre la caja y la cocina.

Este trabajo no pretende un alcance de automatización de todo el negocio, sino únicamente se centra en los pedidos para llevar en las áreas de caja y cocina.

4. OBJETIVOS

4.1. OBJETIVO GENERAL

Desarrollar un prototipo de aplicación web que mejore la comunicación entre el área de caja y la cocina en el restaurante Santa Gula con el fin de optimizar el proceso de administración de los pedidos para llevar.

4.2. OBJETIVOS ESPECÍFICOS

- Identificar los problemas que afectan la transmisión de pedidos entre caja y cocina.
- Diseñar la estructura funcional del prototipo que permita registrar y transmitir pedidos en tiempo real.
- Implementar una interfaz intuitiva que facilite el uso del sistema por parte del personal del restaurante.
- Aportar al entorno productivo a través de la creación de una solución tecnológica que mejore los procesos internos del restaurante Santa Gula y ayude a una mejor calidad de atención a sus clientes.
- Aplicar la metodología ágil Scrum durante el desarrollo del prototipo, promoviendo la planificación iterativa, la retroalimentación continua y la mejora progresiva del sistema.

5. MARCO TEÓRICO Y CONCEPTUAL

5.1 **Aplicación web:** Programa informático en navegadores web y que facilita la comunicación entre el sistema web y el operador del sistema mediante interfaces gráficas. No requiere instalación y permite el acceso a través de diferentes dispositivos conectados en red (García y Pérez, 2022).

5.2 **Comunicación interna:** Define cómo fluye la información dentro de una organización, entre sus diferentes áreas o departamentos. La comunicación mejora

la coordinación, evita malentendidos y genera una mayor eficiencia operativa (Fundación Telefónica, 2021).

5.3 **Pedidos para llevar:** Pedidos realizados por clientes para llevar el pedido del restaurante a otro lugar. Este servicio requiere una transmisión de datos eficiente y fiable desde la caja registradora hasta la cocina para evitar errores y retrasos.

5.4 **Prototipo:** Modelo operativo inicial de un sistema donde se prueban y validan las características principales antes de la implementación final. Los prototipos permiten la entrada de datos del usuario y el perfeccionamiento continuo del producto (Pressman, 2014).

5.5 **Metodología Ágil:** Las metodologías ágiles se entienden como un enfoque flexible de gestión de proyectos que promueve la colaboración, la adaptación constante y la entrega continua de valor al cliente a través de ciclos iterativos denominados sprints (Asana, 2025).

5.6 **Diagramas UML:** Los diagramas UML son representaciones visuales que permiten modelar el comportamiento, la estructura y los límites de nuestro sistema. Funcionan como una especie de planos, que describen cómo se organiza y funciona el software, facilitando el análisis y diseño orientado a objetos, aunque no son un lenguaje de programación (Lucidchart, 2025).

5.7 **Scrum:** Es un marco de trabajo pensado para organizar el desarrollo de proyectos de forma incremental e iterativa. Busca dividir el trabajo en “sprints” que

generalmente se dividen en ciclos fijos de 1 a 4 semanas. Estos tienen una duración fija, sin posibilidad de alargar (Deemer et al., 2009).

5.8 **Arquitectura MVC:** Es una arquitectura por capas utilizada en aplicaciones web dividiendo el desarrollo en tres; Modelo, Vista y Controlador (MVC). El Modelo se encarga del manejo de la información, la Vista presenta la interfaz al actor del sistema, y el Controlador, actúa como un intermediario para manipular el Modelo y presentar la información en la vista (Garzón, C., 2019).

5.9 **Requerimientos:** Son las descripciones que debe ofrecer el sistema y las condiciones en las que debe operar. Se dividen en requerimientos funcionales y no funcionales, que indican qué funciones realizará el sistema y que establecen las restricciones y criterios de calidad que deben cumplirse respectivamente (Sommerville, I., 2005).

5.10 **Base de datos:** Corresponde a un conjunto de datos que se encuentra almacenado en un medio externo o local, organizada mediante estructuras específicas. Su diseño responde a las necesidades de una empresa u organización y puede clasificarse en bases de datos relacionales o no relacionales (Marqués, M., 2009).

5.11 **PlantUML:** Es una herramienta versátil que permite crear de manera rápida, intuitiva y sencilla distintos tipos de diagramas. Gracias a su lenguaje simple e intuitivo, los usuarios pueden definir diagramas con facilidad (PlantUML, 2025).

5.12 **Power Designer:** Es la principal herramienta utilizada para el modelado de base de datos. Ayuda a visualizar, analizar y gestionar modelos de datos (Power Designer, 2024).

5.13 **Node.js:** Node.js es un entorno de ejecución de JavaScript de código abierto y compatible con múltiples plataformas, que permite la ejecución de código JavaScript fuera del navegador web. Su arquitectura está orientada al desarrollo de aplicaciones escalables gracias a su modelo asíncrono y basado en eventos (Node JS, 2015).

5.14 **Angular:** Es un framework web desarrollado por Google que permite crear aplicaciones rápidas, estables y escalables. Proporciona herramientas, APIs y bibliotecas que facilitan y optimizan el proceso de desarrollo (Angular, 2025).

5.15 **SQL Server:** QL Server es un sistema gestor de bases de datos relacional creado por Microsoft, que permite el almacenamiento, administración y consultas de datos utilizando Transact-SQL y ofrece herramientas como SQL Server Management Studio para facilitar su administración (Actian, 2025).

6. ALCANCE

El propósito de este proyecto consiste en la construcción de un prototipo funcional tipo aplicación web para el restaurante Santa Gula. El prototipo posibilitará la administración digital y en tiempo real de los pedidos para llevar, lo que hará más fácil el contacto entre la cocina y la caja. El sistema se concentrará en mejorar el flujo de registro, envío, preparación y seguimiento de pedidos.

El desarrollo se enfocará únicamente en los procesos internos asociados con la administración de pedidos, dejando fuera las funciones administrativas, de control de inventario, facturación o servicio en mesa. De igual manera, el sistema estará hecho para funcionar en una red local, lo que asegura que su funcionamiento sea estable y controlado según los requerimientos del restaurante.

7. METODOLOGÍA

El desarrollo del proyecto se basará en un enfoque ágil, empleando la metodología Scrum combinada con la técnica de prototipado evolutivo y la arquitectura Modelo-Vista-Controlador (MVC).

Scrum permitirá gestionar el proyecto mediante iteraciones cortas (sprints), entregando avances funcionales del prototipo y recibiendo retroalimentación continua del personal del restaurante Santa Gula. El prototipado garantizará que las interfaces sean intuitivas y validadas tempranamente por los usuarios finales.

La arquitectura MVC se aplicará durante la fase de desarrollo para mantener una estructura ordenada y escalable del sistema:

- Modelo (Model): gestionará las operaciones de los pedidos, la base de datos y la lógica del negocio.
- Vista (View): representará la interfaz web con la que interactúan los usuarios de caja y cocina.
- Controlador (Controller): coordinará el intercambio de información entre la Vista y el Modelo, procesando las solicitudes del usuario y actualizando la información.

Las fases del proyecto serán:

- Levantamiento de requerimientos: identificación de procesos actuales, necesidades y flujo de comunicación.
- Análisis y diseño del sistema: definición de diagramas UML, estructura de datos, arquitectura MVC e interfaces gráficas.
- Elegir herramientas para desarrollar.
- Desarrollo del prototipo: implementación por módulos bajo Scrum y MVC, utilizando tecnologías web.
- Pruebas y validación: ejecución de pruebas funcionales del prototipo con el personal del restaurante, con el fin de validar la precisión en el registro y envío de pedidos, así como la usabilidad del sistema desarrollado.
- Documentación y ajustes finales: corrección de errores, mejoras visuales y documentación técnica del sistema.

Este enfoque integrado permitirá obtener un prototipo funcional, escalable y alineado a buenas prácticas de ingeniería de software, asegurando eficiencia, mantenimiento y facilidad de uso.

8. PRESUPUESTO Y FINANCIAMIENTO

Para las herramientas de desarrollo no se requiere una inversión, ya que se utilizarán herramientas gratuitas. Sin embargo, el proyecto sí contempla costos asociados al tiempo de desarrollo, infraestructura, conectividad y materiales de prueba, los cuales conforman un presupuesto estimado que se detalla a continuación:

Recurso / Actividad	Descripción	Costo estimado (USD)
Equipos de cómputo	Uso de computadoras del restaurante y desarrollador.	125
Software de desarrollo	Herramientas gratuitas.	0
Tiempo de desarrollo	Trabajo del desarrollador, análisis, programación y documentación del sistema	430
Internet y servicios	Conectividad, energía eléctrica y costos generales del desarrollo	80
Material de pruebas	Documentación, formularios y hojas de validación	20
Total estimado		**655**

Tabla 1: Financiamiento

Todos los gastos asociados al desarrollo y validación del prototipo serán financiados por el restaurante Santa Gula, quien proporcionará los recursos necesarios para la implementación, pruebas y mantenimiento inicial del sistema. No se requiere financiamiento externo ni apoyo institucional adicional.

CAPÍTULO 2 - ANÁLISIS

9. REQUERIMIENTOS FUNCIONALES

Para identificar y definir los requerimientos funcionales del sistema se utilizó la técnica de casos de uso, la cual permite describir con claridad la relación entre el sistema y los actores. Esta metodología facilita la comprensión de la forma en que el sistema debe actuar desde el punto de vista del usuario, permitiendo representar las

funcionalidades principales de forma estructurada y comprensible. Esto sirve como punto de partida para el análisis y posterior diseño de la solución propuesta.

RF0. Actores y contexto

- Actores: Usuario Caja, Usuario Cocina.
- Ámbito: Gestión digital y en tiempo real de pedidos para llevar entre Caja y Cocina, sin inventario ni administración.

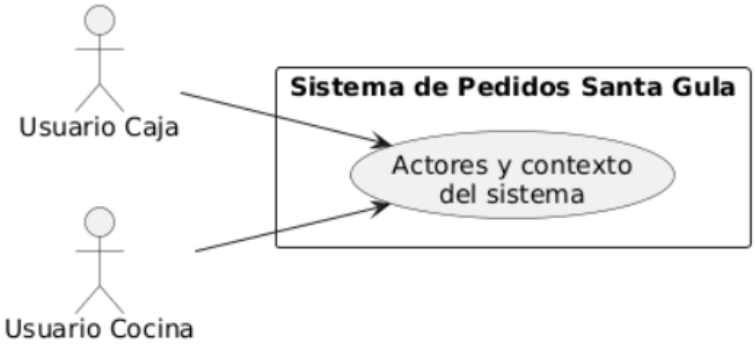


Ilustración 1: Diagrama Caso de Uso RF0

RF1. Catálogo y armado de pedido (Caja)

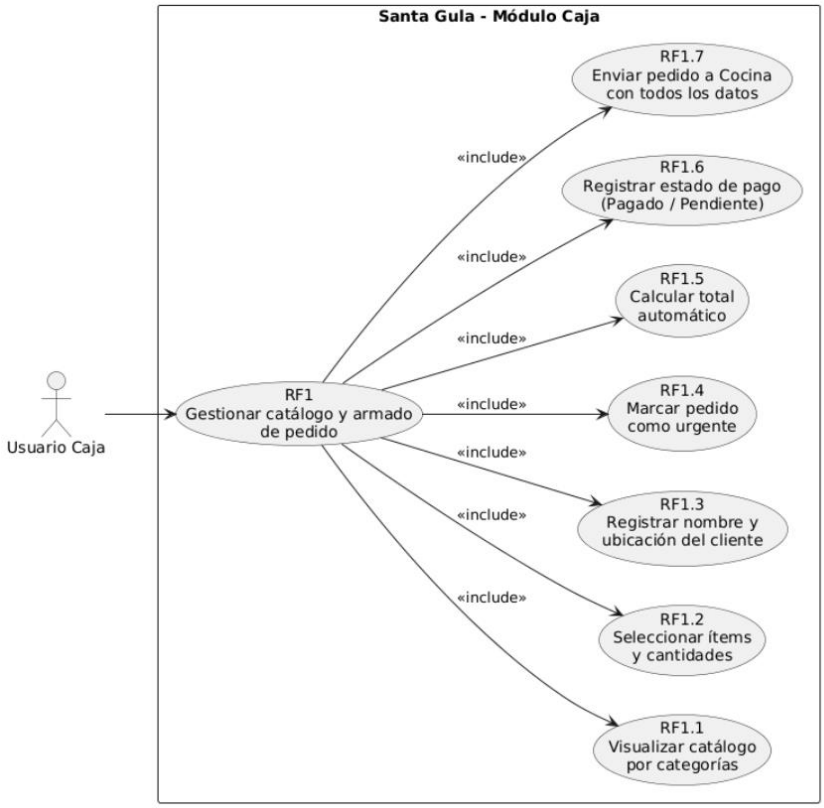


Ilustración 2: Diagrama Casos de Uso RF1

- RF1.1 Visualizar catálogo por categorías: Almuerzos completos (Opción 1, Opción 2), Segundos (Opción 1, Opción 2), Porciones (arroz, ensalada, papas, patacones), Extras (proteína, sopa, jugo, postre).

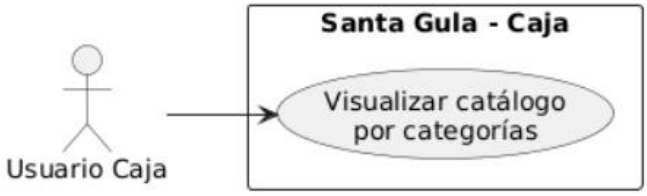


Ilustración 3: Diagrama Caso de Uso RF1.1

- RF1.2 Seleccionar múltiples ítems y cantidades por pedido.

Ilustración 4: Diagrama Caso de Uso RF1.2

- RF1.3 Registrar nombre del cliente y referencia de ubicación (ej.: "Diego Rosales – PUCE, Ingeniería piso 5").

Ilustración 5: Diagrama Caso de Uso RF1.3

- RF1.4 Marcar el pedido como urgente (booleano).

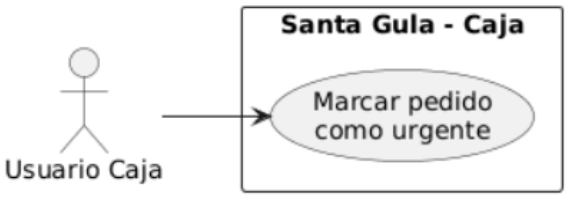


Ilustración 6: Diagrama Caso de Uso RF1.4

- RF1.5 Calcular total automático según selección y precios.

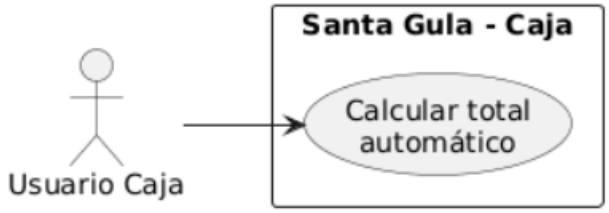


Ilustración 7: Diagrama Caso de Uso RF1.5

- RF1.6 Registrar estado de pago: “Pagado” o “Pendiente (efectivo)”.

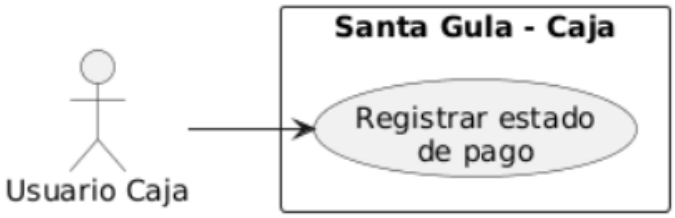


Ilustración 8: Diagrama Caso de Uso RF1.6

- RF1.7 Enviar pedido a Cocina con todos los datos (cliente/ubicación, detalle, urgencia, pago, total, fecha/hora).

Ilustración 9: Diagrama Caso de Uso RF1.7

El sistema deberá generar el ticket automáticamente utilizando los datos enviados desde Caja, sin intervención del usuario de Cocina.

Los requisitos que debe cumplir el ticket son:

- Nombre del cliente y la referencia de ubicación.
- Incluir el detalle de productos del pedido (nombre, cantidad y total).
- El ticket con el estado del pago (pagado o el monto pendiente).

RF2. Bandeja de pedidos (Cocina)

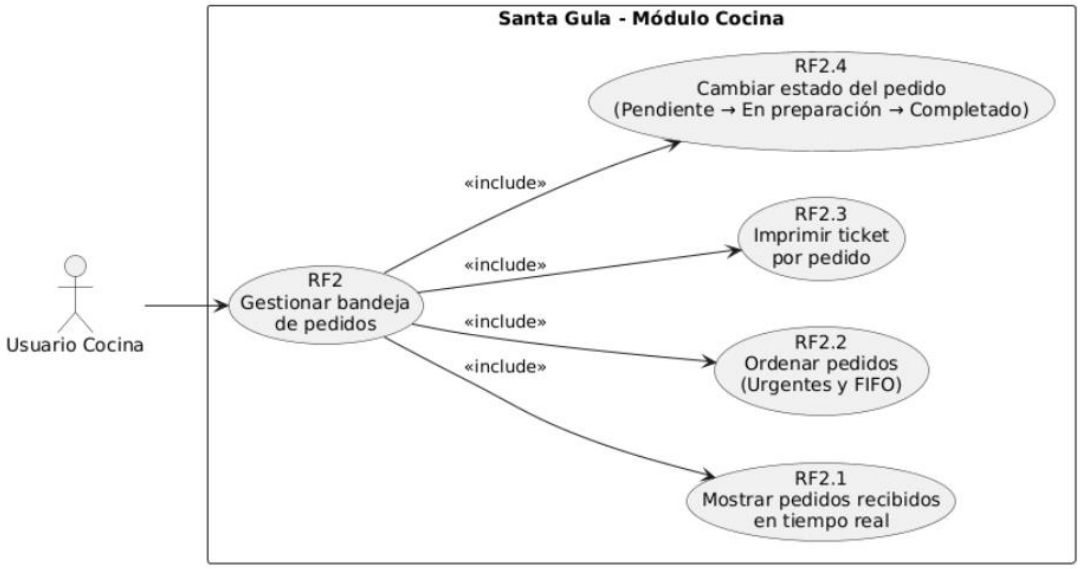


Ilustración 10: Diagrama Caso de Uso RF2

- RF2.1 Mostrar pedidos recibidos en tiempo real.

Ilustración 11: Diagrama Caso de Uso RF2.1

- RF2.2 Ordenar por regla de cola: primero Urgentes (al inicio con alerta), luego por orden de llegada (FIFO).

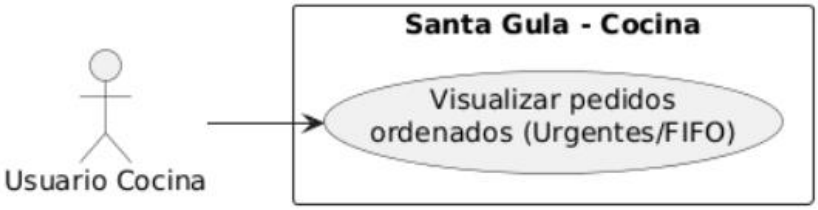


Ilustración 12: Diagrama Caso de Uso RF2.2

- RF2.3 Imprimir ticket por pedido.

Ilustración 13: Diagrama Caso de Uso RF2.3

- RF2.4 Cambiar estado del pedido: "Pendiente" → "En preparación" → "Completado"; mantener visible hasta completar. (Trazabilidad del flujo).

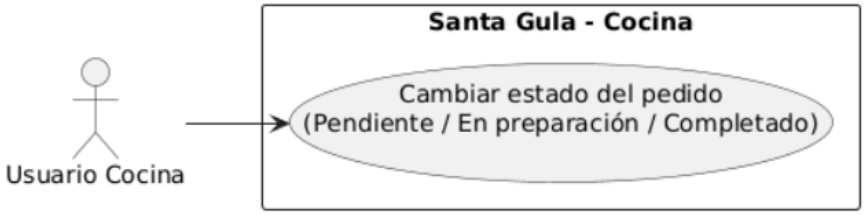


Ilustración 14: Diagrama Caso de Uso RF2.4

RF3. Persistencia de datos

- RF3.1 Guardar en BD: datos del cliente (nombre, ubicación), detalle de productos, total, estado de pago, timestamp de registro.

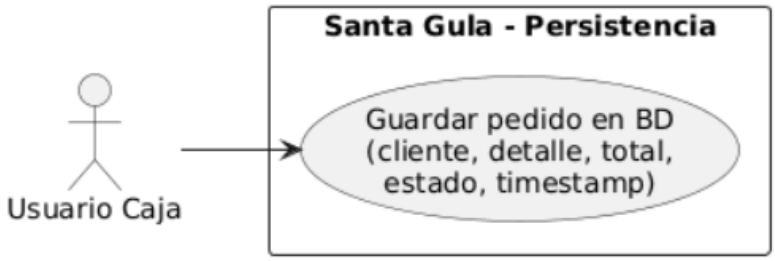


Ilustración 15: Diagrama Caso de Uso RF3

RF4. Gestión básica operativa

- RF4.1 Refresco de cola en Cocina sin recargar la página.

Ilustración 16: Diagrama Caso de Uso RF4

RF5. Configuración de precios y productos

- RF5.1 Cargar catálogo y precios iniciales en BD para: almuerzos completos (Opc. 1 / Opc. 2), segundos (Opc. 1 / Opc. 2), porciones (arroz, ensalada, papas, patacones), extras (proteína, sopa, jugo, postre).

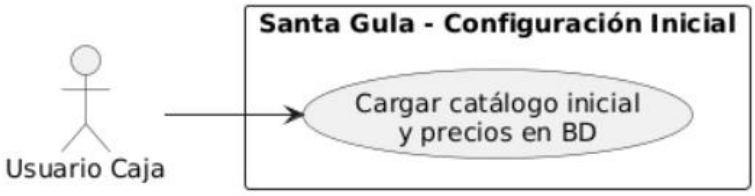


Ilustración 17: Diagrama Caso de Uso RF5

10.DIAGRAMA ENTIDAD RELACIÓN

Un modelo entidad relación es fundamental para definir cómo se estructuran los datos dentro del sistema. Gracias a él podemos organizar toda la información que interviene en la gestión de los pedidos, desde que se registran en la caja hasta que llegan a la cocina. Con base en los requisitos funcionales que se identificaron, se establecen las entidades, sus atributos y las relaciones necesarias para que los pedidos puedan crearse, enviarse, procesarse y monitorearse a lo largo de todo el flujo. Este modelo ayuda a mantener la información ordenada, consistente y confiable, además sirve como punto de partida para diseñar la base de datos que la aplicación utilizará más adelante.

La siguiente tabla muestra las entidades del modelo que se ven involucradas en cada requerimiento funcional del sistema.

Requerimiento funcional	Entidades / Tablas implicadas
RF0. Actores y contexto	(No aplica)
RF1. Catálogo y armado	Usuario, Pedido, DetallePedido, Producto, Categoria
RF2. Bandeja de pedidos	Pedido, DetallePedido, Producto
RF3. Persistencia	Pedido, DetallePedido
RF4. Refresco de cola	Pedido, DetallePedido
RF5. Configuración de catálogo	Producto, Categoria

Tabla 2: Entidades Modelo E-R

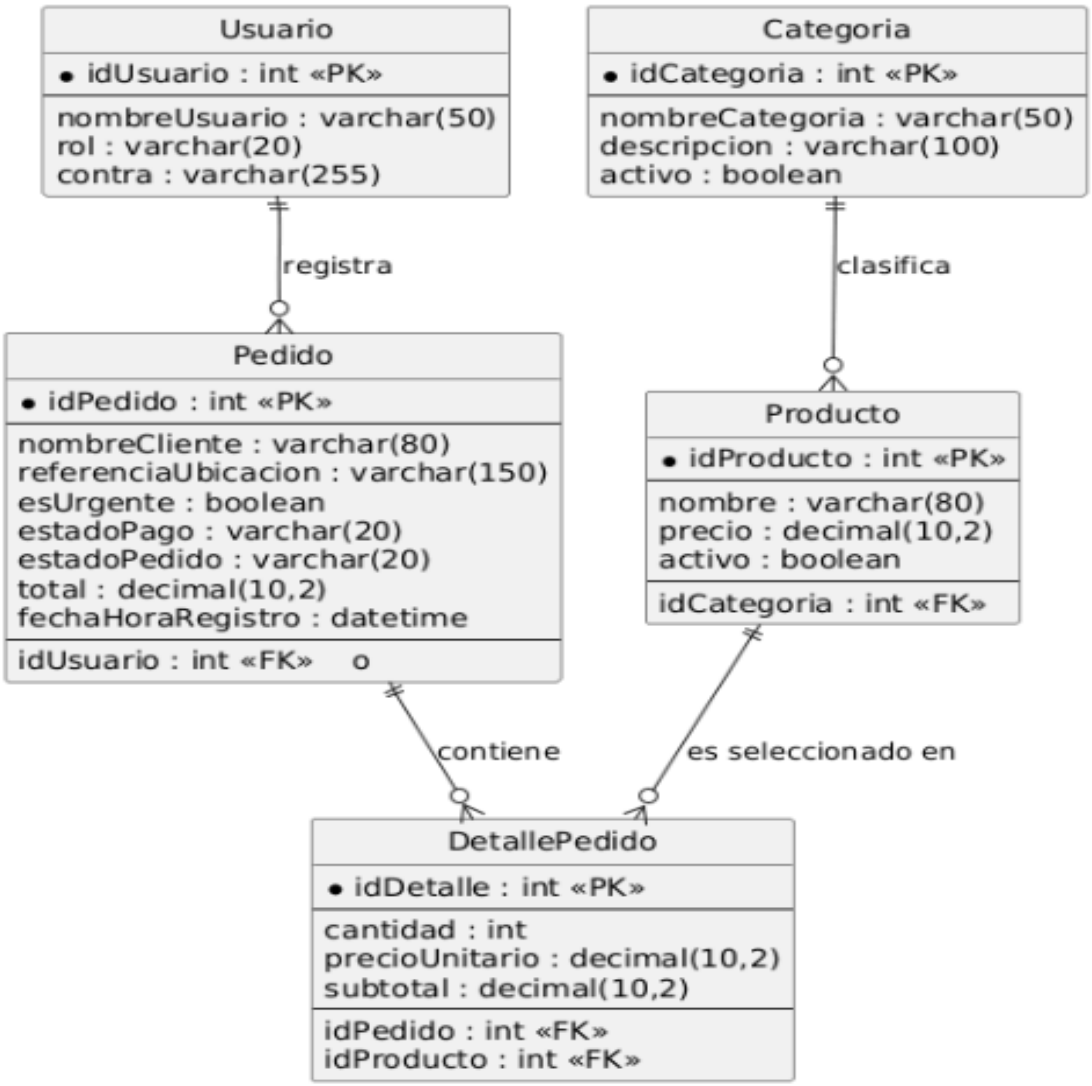


Ilustración 18: Diagrama Entidad-Relación

11.DIAGRAMA DE FLUJO DE DATOS

11.1. N0

Ilustración 19: Diagrama FD N0

Mediante el Diagrama de Flujo de Datos de Nivel 0, se visualiza el sistema bajo el concepto de 'caja negra', priorizando la interacción externa antes que la lógica interna. Aquí se observa cómo el Sistema de Gestión de Pedidos centraliza la comunicación entre la caja y la cocina. Los flujos clave identificados incluyen el ingreso de datos del cliente desde caja, la transmisión de prioridades de preparación hacia la cocina y las notificaciones de actualización de estado. Este nivel es fundamental para delimitar el alcance del proyecto sin profundizar aún en la persistencia de datos o procesos secundarios.

11.2. N1

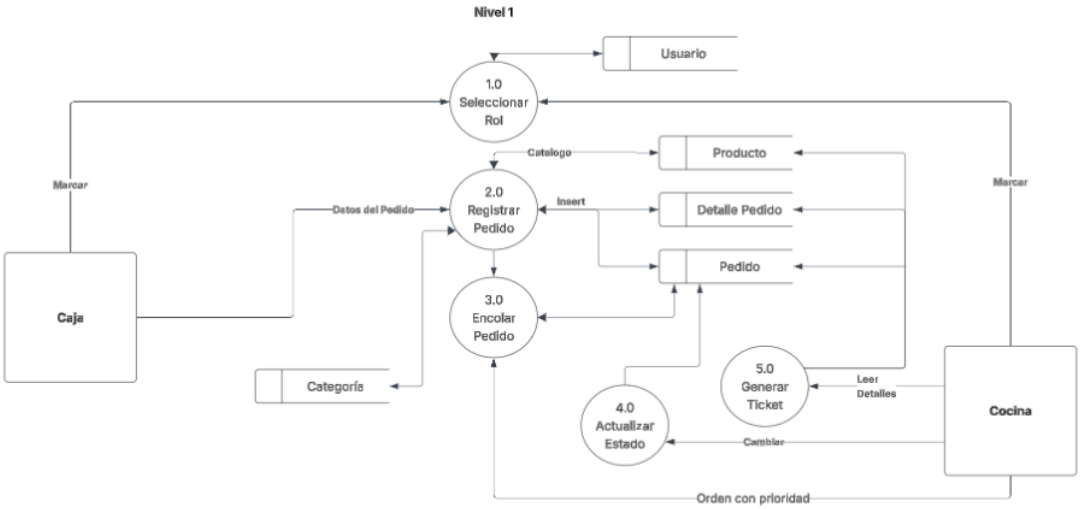


Ilustración 20: Diagrama FD N1

Al profundizar en el Nivel 1, el sistema deja de verse como una unidad para dividirse en sus funciones operativas fundamentales. En esta etapa, es posible observar el ciclo de vida detallado de una orden, partiendo desde el registro inicial en el catálogo de productos, pasando por la gestión inteligente de colas según la prioridad del pedido, hasta llegar a la actualización de estados y la impresión final del ticket. Este desglose es vital porque permite mapear con precisión cómo cada interacción de los usuarios afecta a las tablas de la base de datos, asegurando que el flujo de información cumpla estrictamente con las reglas de negocio establecidas para el restaurante.

11.3. **N2** (Registrar Pedido)

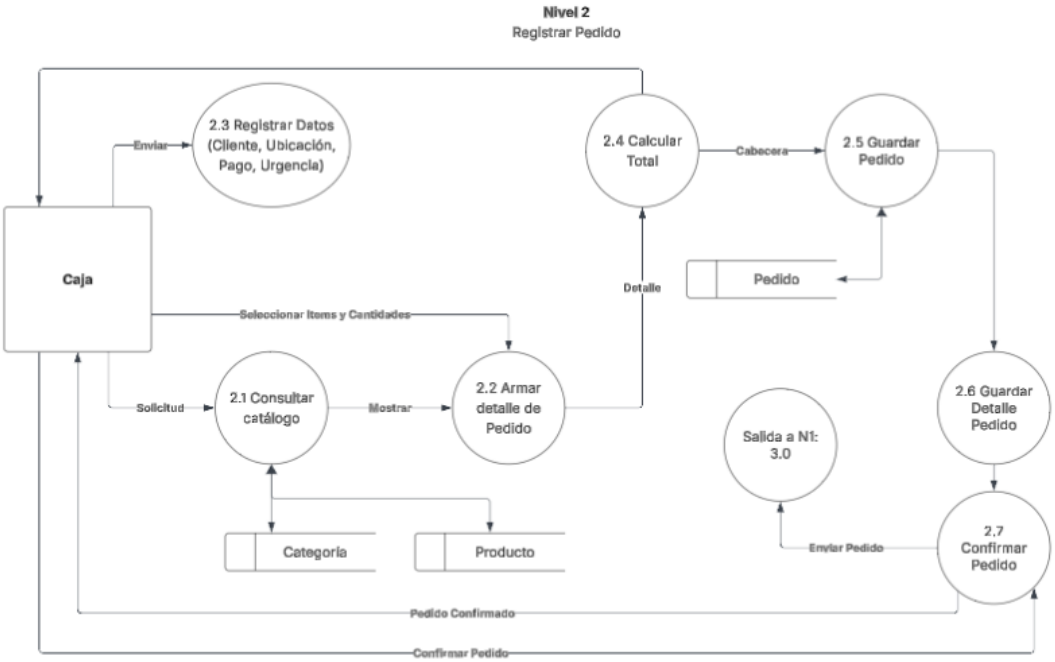


Ilustración 21: Diagrama FD Nivel2 (Registrar Pedido)

En el Nivel 2, se profundiza en el proceso de Registrar Pedido, que es básicamente el corazón del prototipo. Lo que buscamos aquí es desglosar la complejidad de manejar tantas variables al mismo tiempo. Al dividirlo en tareas como navegar por el catálogo, capturar datos del cliente o validar el pago se logra que el flujo sea mucho más limpio. Además, esto permite ver cómo los datos se separan correctamente entre la cabecera y el detalle de la orden, garantizando que cada pedido llegue a cocina con información exacta y bien respaldada.

El uso de los diagramas de flujo en sus niveles 0, 1 y 2 nos ofrece una “hoja de ruta” clara del sistema. Esta estructura garantiza que lo que el negocio necesita coincida perfectamente con la lógica de programación y cómo se guardan los datos, manteniendo la coherencia en todo momento.

12.DIAGRAMA DE PROCESOS

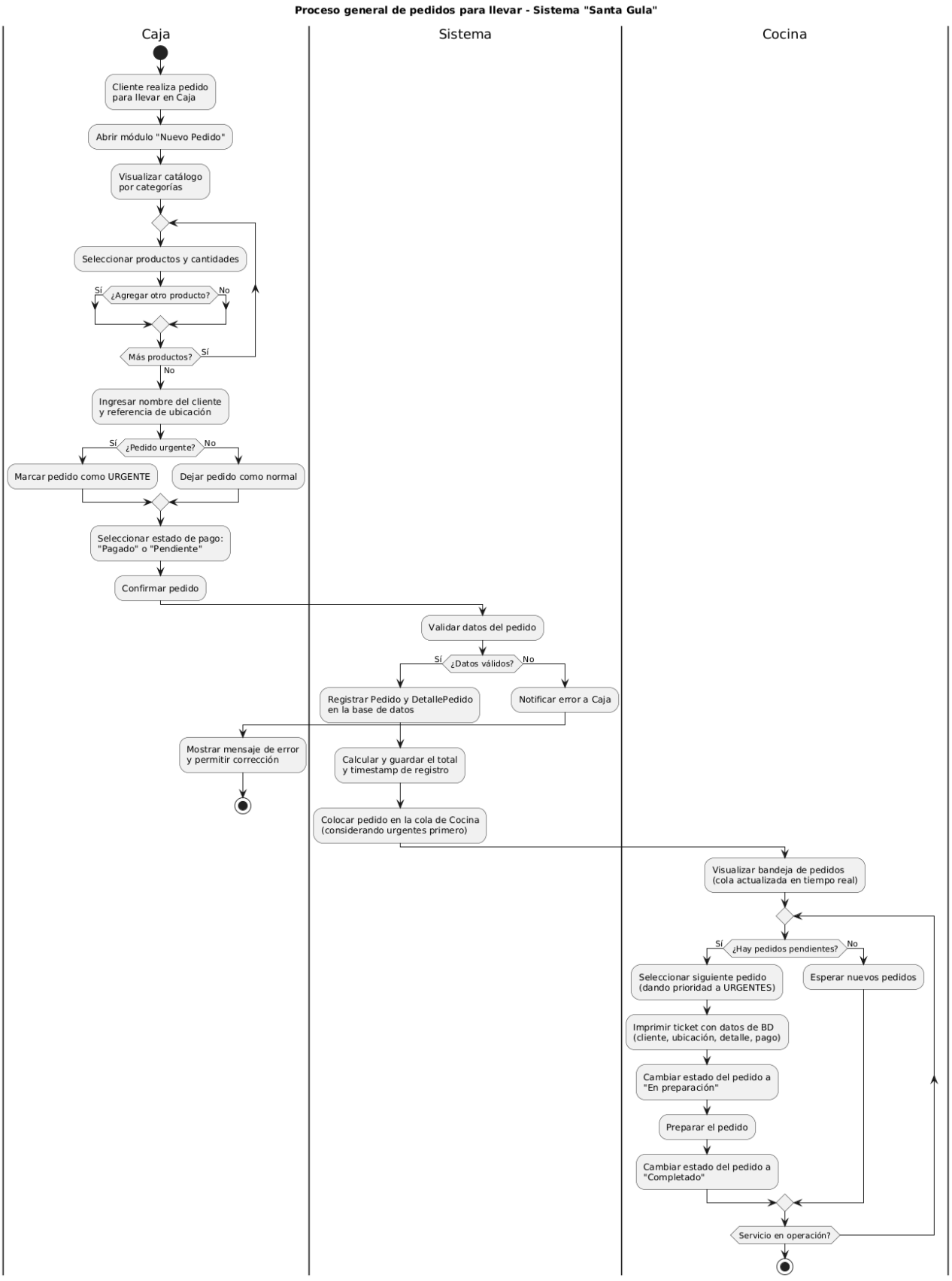


Ilustración 22: Diagrama de Procesos

Para el desarrollo de los diagramas realizados, como el de casos de uso, el modelo entidad relación, los diagramas de flujo y el diagrama de procesos, se utilizaron herramientas de inteligencia artificial junto con la plataforma en línea PlantUML. La incorporación de estas tecnologías permitió automatizar el proceso de modelado, esto agilizó la generación de versiones de los diagramas. Durante el desarrollo del proyecto, los modelos se revisaron y ajustaron en varias ocasiones, comparándolos con los requisitos funcionales y las necesidades específicas del proyecto Santa Gula, hasta garantizar que todos los diagramas fueran coherentes, completos y se ajustaran al alcance definido del sistema.

13.DIAGRAMAS DE FLUJO

13.1. RF1 – Catálogo y armado de pedido (Caja)

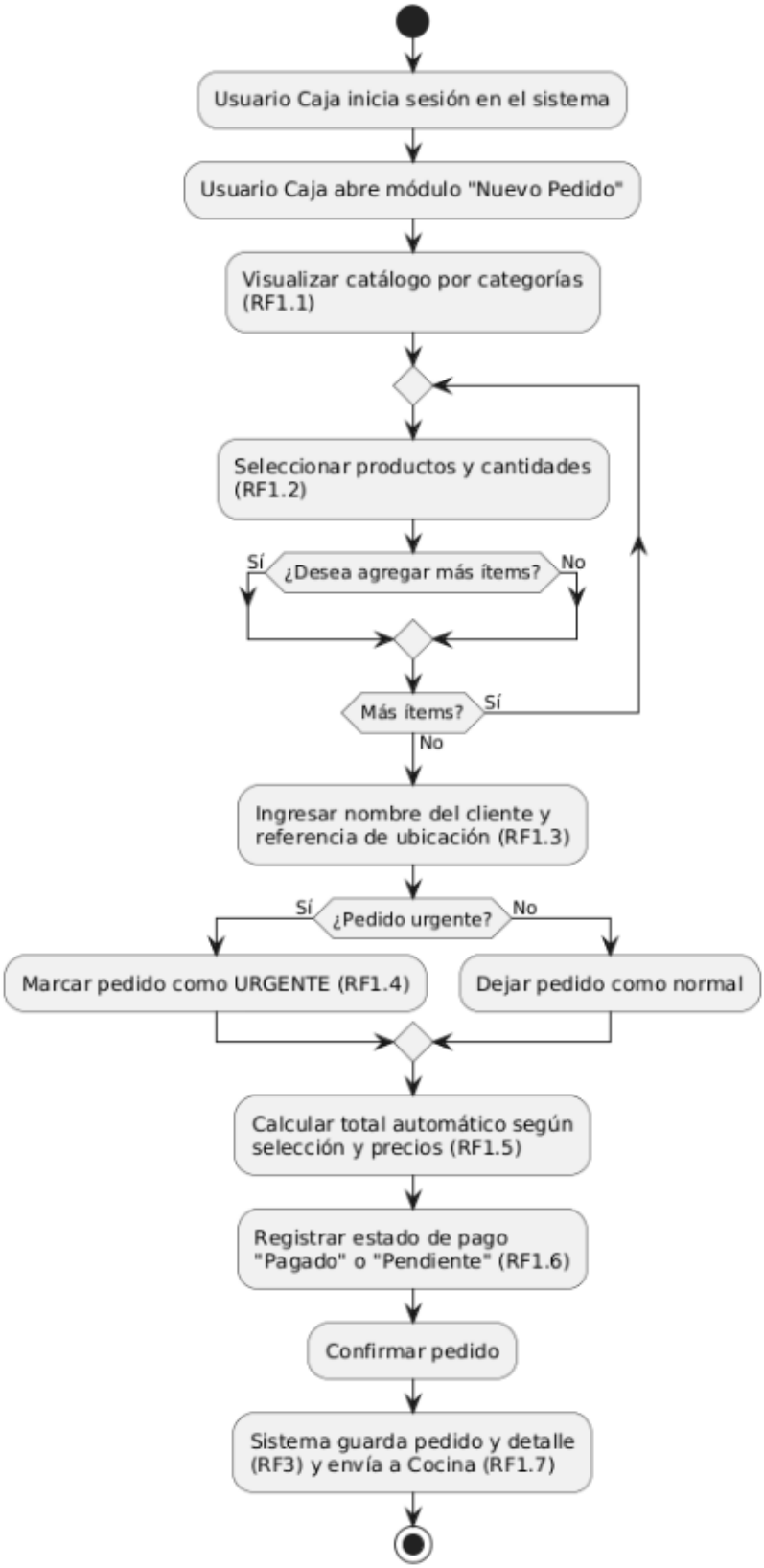


Ilustración 23: Diagrama de flujo RF1

El diagrama de flujo representa el procedimiento que sigue el usuario Caja para el registro de un pedido dentro del sistema. El proceso inicia con el acceso al módulo de nuevo pedido y la consulta del catálogo organizado por categorías, lo que permite seleccionar los productos y sus cantidades. A continuación, se ingresan los datos del cliente y se define la prioridad del pedido, considerando si es urgente o normal. El sistema realiza automáticamente el cálculo del total, registra el estado de pago correspondiente y, tras la confirmación, almacena la información del pedido junto con su detalle, enviándolo finalmente a Cocina para su preparación.

13.2. RF2 – Bandeja de pedidos (Cocina)

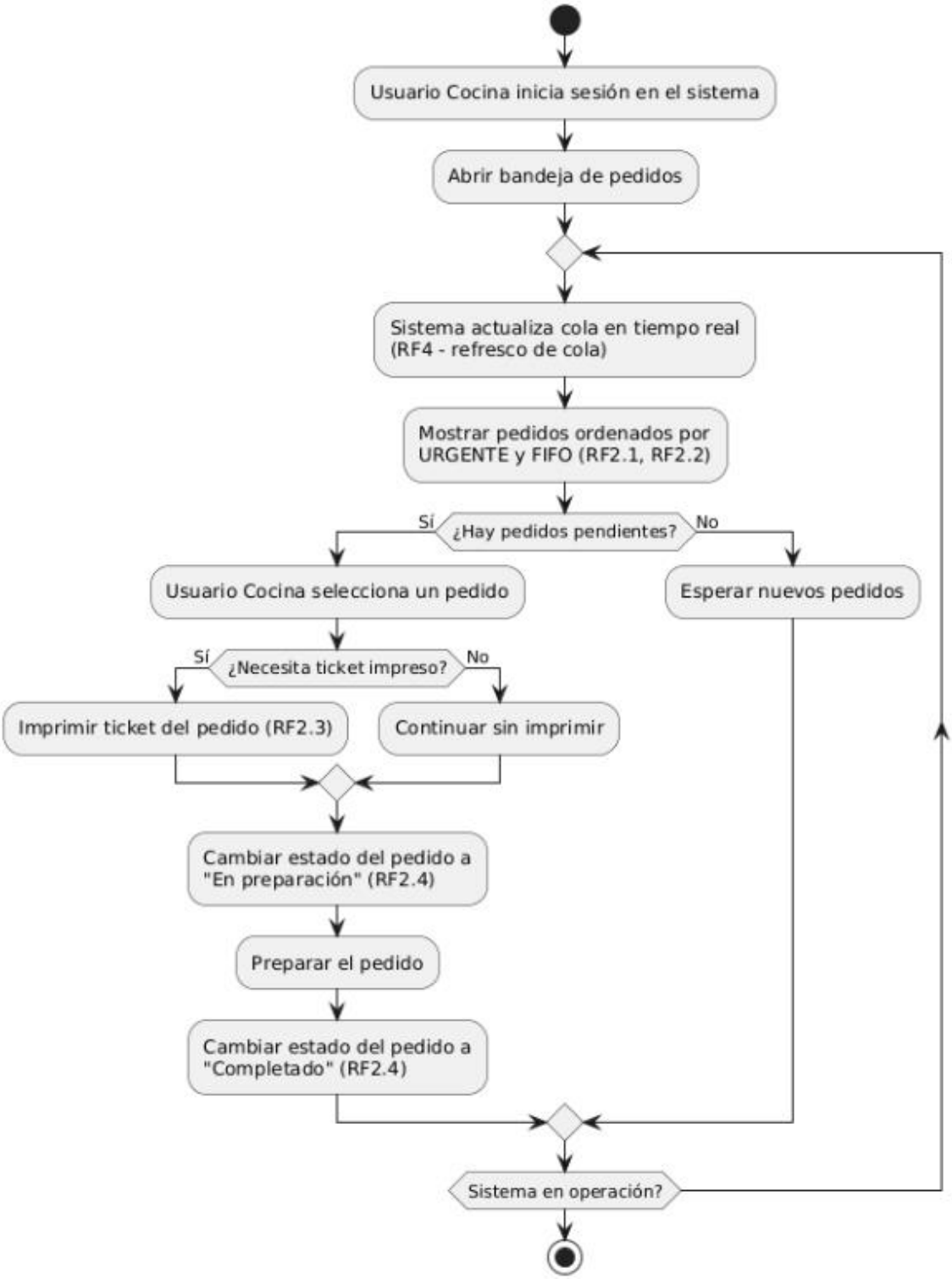


Ilustración 24: Diagrama de flujo RF2

El diagrama de flujo describe el comportamiento del usuario Cocina durante la gestión y preparación de los pedidos. El proceso inicia con el acceso a la bandeja de pedidos, donde el sistema mantiene actualizada la cola en tiempo real y muestra los pedidos ordenados según prioridad de urgencia y criterio FIFO. Cuando existen pedidos pendientes, el usuario selecciona uno, decide si requiere la impresión del ticket y actualiza su estado a “En preparación”. Tras completar la elaboración, el pedido es marcado como “Completado”, permitiendo que el sistema continúe en operación y a la espera de nuevos pedidos, garantizando un flujo continuo y ordenado en la atención.

13.3. RF3 – Persistencia de datos

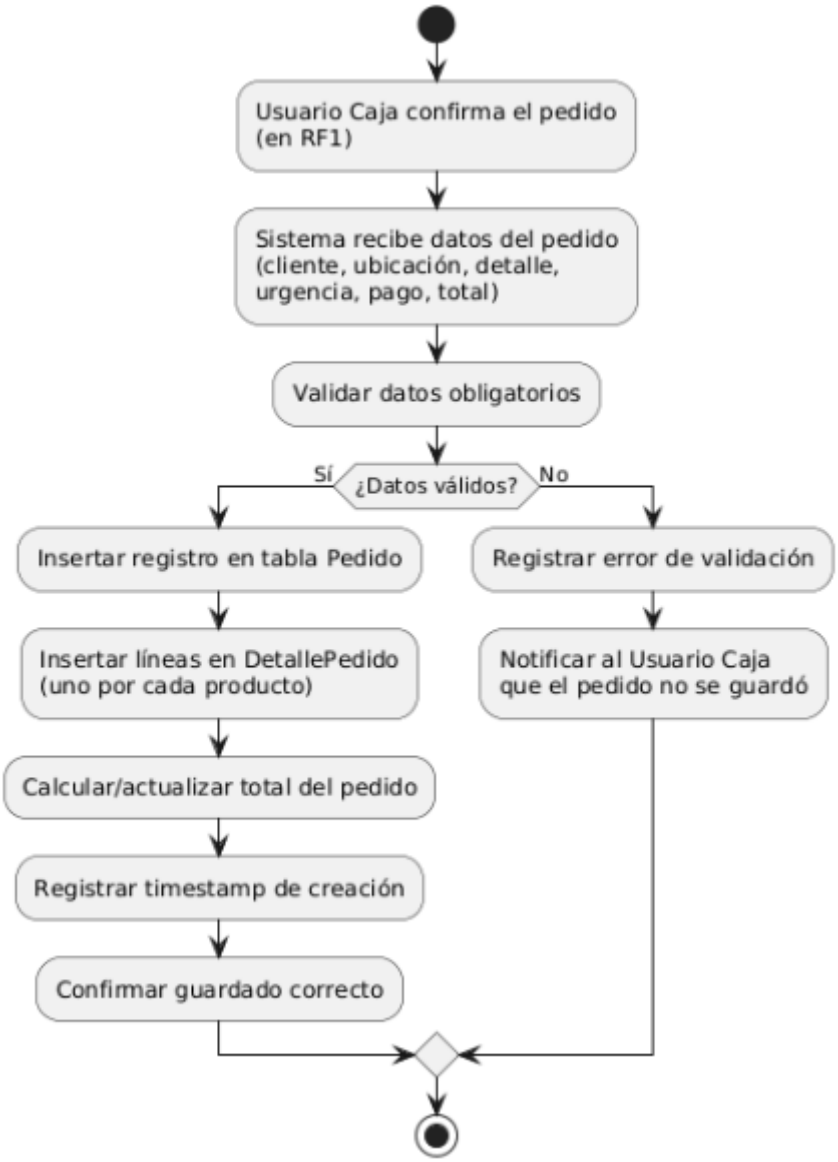


Ilustración 25: Diagrama de flujo RF3

Este diagrama de flujo representa el proceso interno que ejecuta el sistema al momento de confirmar y registrar un pedido. El flujo inicia cuando el usuario Caja confirma el pedido, tras lo cual el sistema recibe y valida los datos obligatorios, incluyendo información del cliente, detalle del pedido, prioridad y estado de pago. Si la validación es exitosa, se procede al registro de la cabecera del pedido y de sus líneas de detalle en la base de datos, se calcula y actualiza el total y se registra la fecha y la hora

de creación. En caso de que los datos no sean válidos, el sistema genera un mensaje de error y notifica al usuario que el pedido no fue almacenado.

13.4. RF4 – Gestión básica operativa

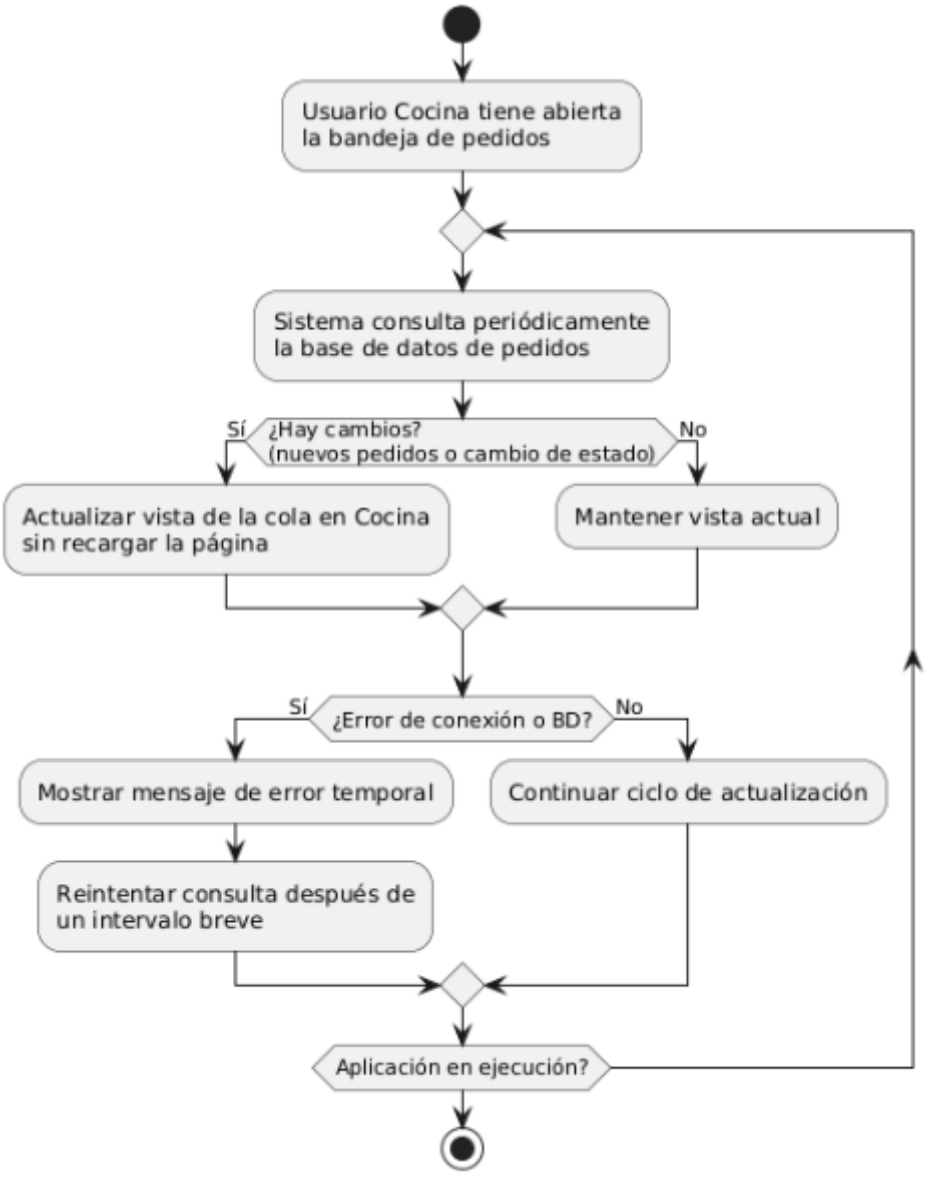


Ilustración 26: Diagrama de flujo RF4

El diagrama de flujo describe el mecanismo de actualización continua de la bandeja de pedidos en el módulo de Cocina. El proceso se mantiene activo mientras la aplicación está en ejecución, realizando consultas periódicas a la base de datos para detectar nuevos pedidos o cambios en su estado. Cuando se identifican modificaciones, la vista se actualiza de forma inmediata sin requerir la recarga de la página, caso contrario, se mantiene la información actual. Adicionalmente, el flujo contempla el manejo de errores de conexión, mostrando mensajes temporales y reintentando la consulta tras un intervalo breve, garantizando la disponibilidad y estabilidad del sistema durante su operación.

13.5. RF5 – Configuración de precios y productos iniciales

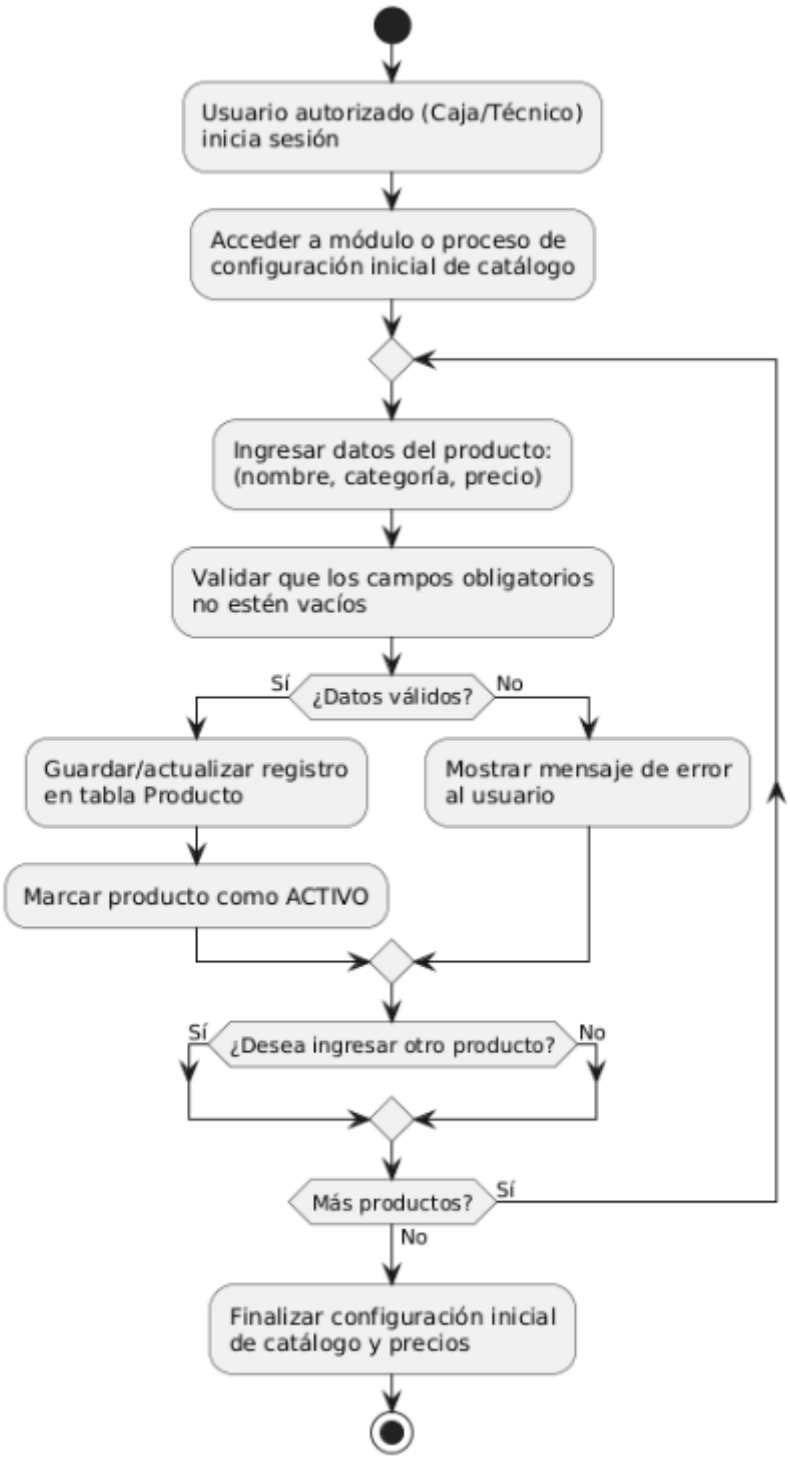


Ilustración 27: Diagrama de flujo RF5

Este diagrama de flujo describe el proceso de configuración inicial del catálogo de productos, el cual es realizado por un usuario autorizado. El flujo inicia con el acceso al módulo de configuración, donde se ingresan los datos básicos del producto, como nombre, categoría y precio. El sistema valida que los campos obligatorios estén completos antes de permitir el registro o actualización del producto en la base de datos, marcándolo posteriormente como activo. En caso de existir errores de validación, se notifica al usuario para su corrección. El proceso permite el ingreso de múltiples productos de forma consecutiva y finaliza cuando se completa la configuración inicial del catálogo y los precios del sistema.

14.REQUERIMIENTOS NO FUNCIONALES

RNF1. Rendimiento y Concurrencia

- RNF1.1 La creación de pedido debe completar en ≤ 3 s en condiciones normales de red local.
- RNF1.2 La publicación/visibilidad del pedido en Cocina debe reflejarse en ≤ 5 s.
- RNF1.3 La cola debe reordenarse inmediatamente cuando un pedido sea marcado como Urgente.

RNF2. Disponibilidad y Operación

- RNF2.1 Funcionamiento en red local con tolerancia a cortes breves.
- RNF2.2 Si se pierde la conexión temporalmente, el sistema reintenta y alerta al usuario, no debe perderse el pedido.

RNF3. Usabilidad (Caja y Cocina)

- RNF3.1 Interfaz intuitiva, con flujo en 3 pasos en la Caja: (1) Selección de productos, (2) Datos cliente, ubicación y pago, (3) Confirmar y enviar.
- RNF3.2 En Cocina, vista tipo tablero con prioridad de urgencia y filtros por estado.
- RNF3.3 Tamaños de fuente y contraste aptos para lectura rápida en cocina.

RNF4. Seguridad y Datos

- RNF4.1 Autenticación mínima por rol (Caja, Cocina).
- RNF4.2 Validación en cliente y servidor (campos obligatorios, tipos, rangos).

RNF5. Calidad, Mantenibilidad y Arquitectura

- RNF5.1 Patrón MVC (separación de responsabilidades).
- RNF5.2 Código con estándares (linting) y pruebas funcionales básicas (creación de pedido, urgencia, impresión, cambio de estado).
- RNF5.3 Documentación breve de endpoints y modelo de datos.

RNF6. Tecnologías objetivo

Backend: Node.js, BD: SQL Server, Frontend: Angular.

- RNF6.1 Uso de ORM/driver estable para SQL Server.
- RNF6.2 Impresión de ticket compatible con impresoras térmicas.

CAPÍTULO 3 – TECNOLOGÍA

15.HERRAMIENTAS A UTILIZAR

La selección de las herramientas tecnológicas para el desarrollo del prototipo del sistema Santa Gula se realizó considerando criterios de rendimiento, seguridad de la información y simplicidad de implementación, acordes al alcance definido del proyecto y a su operación en un entorno controlado. El objetivo principal fue contar con tecnología adecuada para garantizar estabilidad y rapidez en la respuesta del sistema.

15.1. Backend

Node.js fue elegido como tecnología para el desarrollo del backend gracias a su diseño basado en eventos junto con un modelo de entrada y salida no bloqueante, lo que posibilita la gestión de múltiples solicitudes de manera eficiente.

15.1.1. Fortaleza

- Capaz de manejar múltiples peticiones concurrentes sin afectar el rendimiento del sistema.

15.1.2. Justificación

Para el desarrollo de Santa Gula, el backend debe procesar pedidos enviados desde el módulo de caja y transmitirlos a cocina en tiempo real. Node.js permite mantener un flujo de comunicación ágil entre estos módulos, asegurando respuestas rápidas.

15.2. Gestión de Datos

Para el almacenamiento de la información del sistema se utilizó SQL Server como administrador de base de datos relacional, encargado de administrar los datos correspondientes a usuarios, pedidos, productos y sus respectivos detalles.

15.2.1. Fortaleza

- Alta robustez e integridad referencial en el manejo de datos estructurados.

15.2.2. Justificación

SQL Server asegura la integridad de la información, permitiendo mantener relaciones claras entre las entidades del sistema. Resulta completamente adecuado para gestionar los registros operativos del sistema, asegurando un correcto control y validación de los pedidos dentro del entorno del proyecto.

15.3. Frontend

La interfaz de usuario será desarrollada utilizando Angular, un framework orientado a la creación de aplicaciones web, que facilita la construcción de interfaces dinámicas y estructuradas.

15.3.1. Fortaleza

- Arquitectura modular y enlace de datos bidireccional, que permite sincronizar automáticamente la información entre la vista y la lógica de la aplicación.

15.3.2. Justificación

Angular permite ofrecer una experiencia de uso fluida para los usuarios del sistema, tanto en el módulo de caja como en el de cocina. Las acciones realizadas en la interfaz, como el registro de pedidos o la visualización de estados, se reflejan de forma inmediata sin necesidad de recargar la página, mejorando la eficiencia operativa del sistema.

CAPÍTULO 4 - DISEÑO

16.MODELO CONCEPTUAL

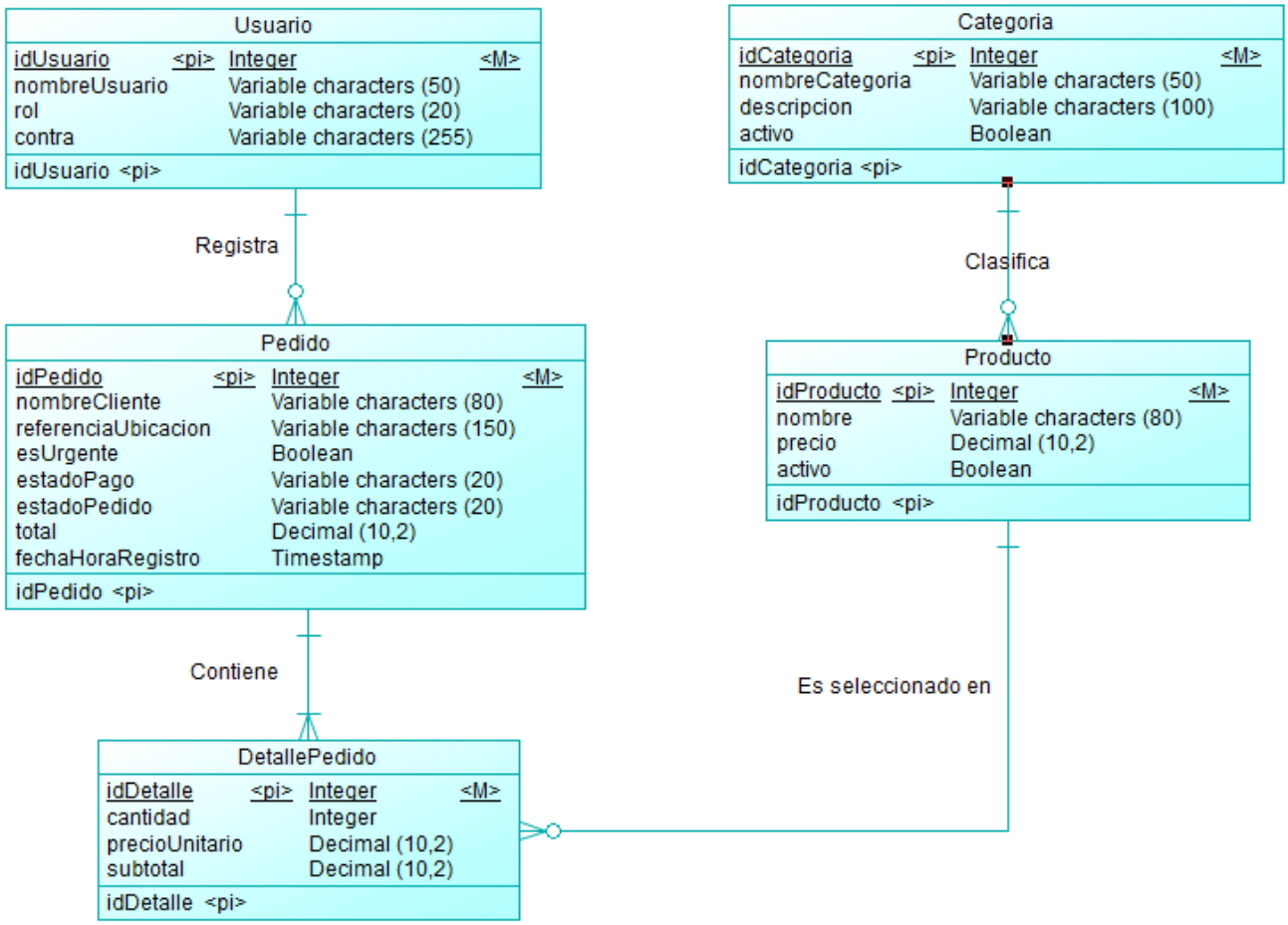


Ilustración 28: Modelo Conceptual

17.MODELO LÓGICO

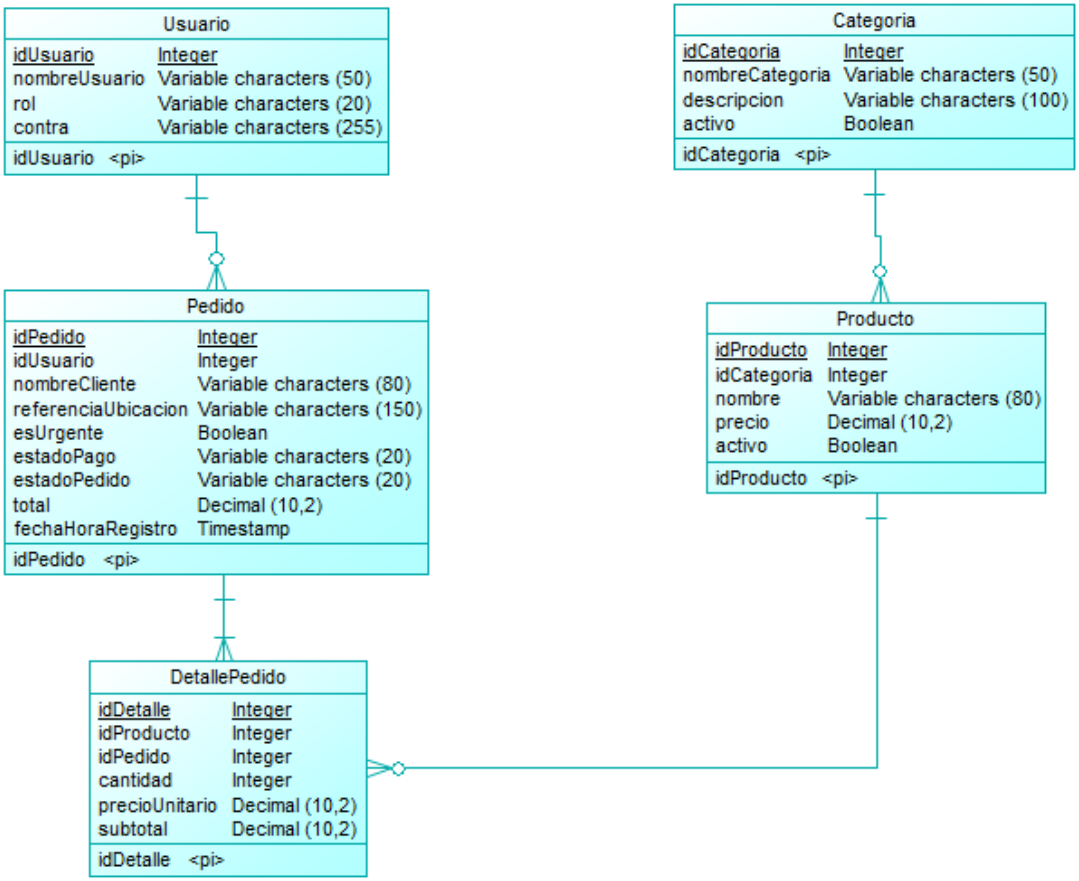


Ilustración 29: Modelo Lógico

18.MODELO FÍSICO

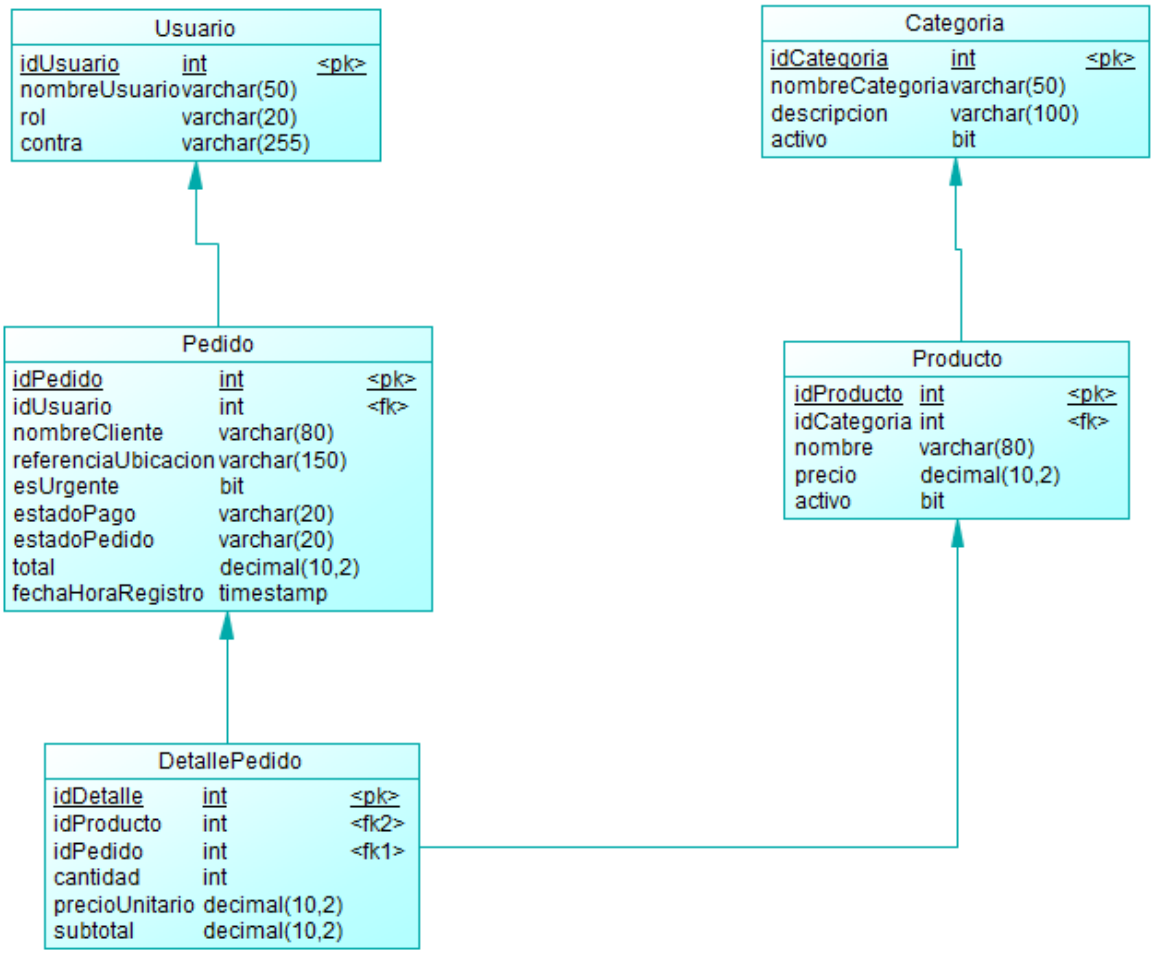


Ilustración 30: Modelo Físico

El diseño de los modelos de datos se llevó a cabo utilizando la herramienta Power Designer, integrando los fundamentos teóricos y metodológicos adquiridos durante la formación académica. Este proceso incluyó una fase de revisión exhaustiva y ajuste técnico, con el fin de alinear la arquitectura de la base de datos a los requerimientos específicos y procesos operativos de la empresa Santa Gula.

19.DISEÑO DEL FRONT END (PROTOTIPO)

Antes de desarrollar el sistema se llevó a cabo la definición del diseño de las pantallas de la aplicación con el propósito de establecer una visión clara y ordenada del producto a construir. Esto permitió anticipar la forma en que el usuario interactuaría con el sistema, así como identificar la disposición de los elementos visuales y la secuencia de uso de las funcionalidades planteadas. El planteamiento previo de las pantallas ayudó a comprender la estructura general del sistema y a anticipar el comportamiento esperado en cada módulo.

19.1 Pantalla de Inicio de Sesión

- Debe permitir el acceso al sistema según el rol del usuario
- Es fundamental porque respalda los requisitos de seguridad y gestión de accesos.

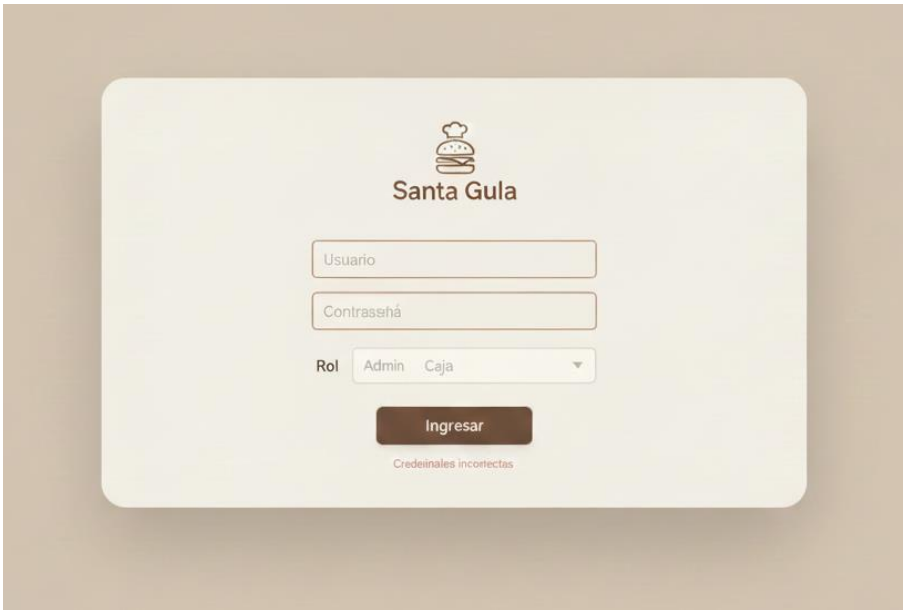


Ilustración 31: Pantalla de Login

19.2 Pantalla principal del módulo Caja

- Es la vista inicial tras el login del usuario Caja
- Funciona como punto de entrada al flujo operativo de pedidos

Ilustración 32: Pantalla Principal (CAJA)

19.3 Pantalla de catálogo y armado del pedido (Caja)

- Esta es la pantalla más importante del prototipo, ya que implementa el RF1 completo

Ilustración 33: Pantalla Pedidos (CAJA)

19.4 Pantalla de datos del cliente y confirmación del pedido

- Continúa el flujo del armado del pedido

Ilustración 34: Pantalla de Confirmación de Pedido

19.5 Pantalla de ticket del pedido (Caja)

- Representa visualmente el comprobante generado

Ilustración 35: Ticket de Pedido

19.6 Pantalla de ticket del pedido (Caja)

- Representa visualmente el comprobante generado

Ilustración 36: Pantalla Principal Pedido (COCINA)

19.7 Pantalla de gestión del pedido (Cocina)

- Vista enfocada en un pedido específico

Ilustración 37: Gestión Pedido (COCINA)

Para el diseño del prototipo del frontend, se utilizaron herramientas de inteligencia artificial para generar las pantallas del sistema de forma rápida y eficiente. Estas herramientas permitieron una visualización clara de la estructura de la interfaz y el flujo de navegación general antes de la implementación. Este proceso se automatizó porque

el diseño del prototipo visual no era un elemento crítico dentro del alcance del proyecto. La prioridad era el desarrollo funcional del sistema. Utilizar inteligencia artificial optimizó el tiempo de trabajo y centró los esfuerzos en los aspectos técnicos del prototipo, demostrando además el uso adecuado de las tecnologías actuales para respaldar el diseño de sistemas de información.

CAPÍTULO 5 - SCRUM

20.DESARROLLO METODOLOGÍA SCRUM

Se adoptó la metodología ágil Scrum, integrando sus principios en la gestión técnica del sistema. Realizar el desarrollo con SCRUM permitió abordar el software de manera iterativa e incremental, organizando el flujo en sprints que facilitaron la entrega continua de módulos funcionales. Gracias a esta estructura, fue posible adaptar el diseño a nuevos requerimientos de manera ágil y mantener un control continuo sobre el avance de cada etapa del proyecto.

20.1. Roles

La metodología Scrum se estructura en tres roles: Propietario del Producto (Product Owner), Maestro Scrum (Scrum Master) y Equipo de Desarrollo (Development Team), los cuales trabajan juntos para garantizar la entrega a través de ciclos de trabajo cortos. Cada rol cumple funciones específicas, como la definición y priorización de requerimientos, la organización del proceso de desarrollo y la implementación técnica del producto. Sin embargo, el desarrollo fue realizado por un único integrante, por lo que estos roles fueron asumidos por una sola persona.

Roles en Scrum	
Scrum Master	Diego Rosales
Dueño del Producto (Product Owner)	Diego Rosales / Santa Gula
Equipo de Desarrollo (Development Team)	Diego Rosales

Tabla 3: Roles

20.2. Fortalezas de Scrum en el proyecto

La aplicación de Scrum aportó diversas fortalezas al desarrollo del proyecto, entre las cuales destacan:

- Organización clara de tareas.
- Entregas incrementales del desarrollo.
- Priorización de funcionalidades críticas.
- Mayor control en los avances del desarrollo.
- Facilidad para detectar y corregir errores oportunamente.

20.3. Horarios de trabajo

El desarrollo del proyecto se realizó de acuerdo con una planificación semanal flexible, considerando la carga académica del desarrollador. Se establecieron sesiones de trabajo distribuidas principalmente en horarios vespertinos y fines de semana, permitiendo avanzar de manera constante en cada sprint sin afectar otras actividades académicas.

Período	Horario	Descripción
Lunes a viernes	16:00 – 21:00	Desarrollo y pruebas del sistema
Sábado	10:00 – 14:00	Avance de sprints y documentación

Tabla 4: Horario

20.4. Priorización de requerimientos

Escala de prioridades		
1	Prioridad Alta	Funcionalidades críticas
2	Prioridad Media	Funcionalidades importantes
3	Prioridad Baja	Funcionalidades complementarias
4	Prioridad Muy Baja	Mejoras opcionales

Tabla 5: Prioridades

Para organizar el desarrollo, clasificamos los requerimientos del sistema en una escala del 1 al 4. Se basa principalmente en qué tanto afectaban la operación diaria y los objetivos del negocio, así, el nivel 1 se asignó para las funciones críticas, mientras que el 4 quedó para aquellas menos urgentes.

20.5. Planificación de los sprints

El desarrollo del prototipo se realizó a través de cuatro ciclos de desarrollo o sprints, lo que permitió avanzar de manera progresiva desde un diseño de baja fidelidad hasta un prototipo funcional. Esta organización hizo posible la verificación de la viabilidad técnica del sistema y del flujo de trabajo. Se enfocó principalmente en validar el correcto funcionamiento del prototipo.

20.5.1. Sprint 1: Planificación y base del prototipo

20.5.1.1. Sprint Planning (S1)

En este sprint se planificó el alcance del prototipo, se definió el modelo de datos y se configuró la estructura base del backend, estableciendo los cimientos técnicos necesarios para su posterior desarrollo.

20.5.1.2. Épicas (S1)

Épica	Descripción	Prioridad
E1	Definición del modelo de base de datos	1
E2	Configuración del entorno de desarrollo	1
E3	Definición de arquitectura del sistema	2

Tabla 6: Épicas Sprint 1

20.5.1.3. Sprint Backlog (S1)

Tarea		
Nro	Nombre	Descripción
1	Creación de la base de datos	Configuración del entorno de almacenamiento.
2	Definición de tablas y relaciones	Modelado de la estructura de datos, llaves primarias y relaciones entre las entidades del sistema.
3	Configuración inicial del backend	Selección e instalación de dependencias básicas
4	Estructuración del proyecto bajo MVC	Organización de los archivos y carpetas diferenciando la lógica de negocio, la gestión de datos y la capa de presentación.

Tabla 7: Sprint Backlog Sprint 1

20.5.1.4. Sprint Review (S1)

Al finalizar el sprint se obtuvo la estructura base del prototipo, con la base de datos definida y el backend preparado para la implementación de funcionalidades.

20.5.1.5. Conclusión del Sprint (S1)

Este sprint permitió reducir la incertidumbre técnica y sentar una base sólida para el desarrollo del prototipo.

20.5.2. Sprint 2: Desarrollo de funcionalidades principales

20.5.2.1. Sprint Planning (S2)

Durante este sprint se planificó el desarrollo de las funcionalidades principales del prototipo, enfocándose en la implementación de la lógica de negocio y las APIs necesarias.

20.5.2.2. Épicas (S2)

Épica	Descripción	Prioridad
E4	Gestión de pedidos	1
E5	Gestión de productos y categorías	2
E6	Autenticación y roles	1

Tabla 8: Épicas Sprint 2

20.5.2.3. Sprint Backlog (S2)

Tarea		
Nro	Nombre	Descripción
5	Desarrollo de controladores	Programación de la lógica que gestiona las peticiones del usuario y coordina las respuestas del sistema.
6	Implementación de endpoints	Creación de las rutas de acceso y puntos de para la transferencia de información entre cliente y servidor.
7	Validación de datos	Establecimiento de reglas y filtros para asegurar que la información ingresada sea correcta y segura.
8	Pruebas de funcionamiento con Postman	Verificación de los servicios mediante el envío de solicitudes para confirmar que el backend responde según lo esperado.

Tabla 9: Sprint Backlog Sprint 2

20.5.2.4. Sprint Review (S2)

Como resultado del sprint, se logró un prototipo funcional del backend, permitiendo registrar y consultar pedidos mediante las APIs desarrolladas.

20.5.2.5. Conclusión del Sprint (S2)

Este sprint consolidó la lógica principal del prototipo, permitiendo validar el flujo básico de operación del sistema.

20.5.3. Sprint 3: Desarrollo de interfaz y pruebas del prototipo

20.5.3.1. Sprint Planning (S3)

En este sprint se planificó el desarrollo de la interfaz de usuario y la realización de pruebas funcionales orientadas a validar el comportamiento del prototipo.

20.5.3.2. Épicas (S3)

Épica	Descripción	Prioridad
E7	Interfaz de usuario (Caja y Cocina)	1
E8	Pruebas funcionales del prototipo	2

Tabla 10: Épicas Sprint 3

20.5.3.3. Sprint Backlog (S3)

Tarea		
Nro	Nombre	Descripción
9	Desarrollo de la interfaz de usuario	Construcción de las vistas del prototipo orientadas a los módulos de caja y cocina, permitiendo la visualización y registro de pedidos.
10	Simulación del flujo de pedidos	Ejecución de escenarios de uso para comprobar el recorrido del pedido dentro del prototipo.
11	Verificación de estados del pedido	Comprobación del correcto cambio de estados del pedido durante la simulación del flujo operativo.
12	Ajustes de usabilidad	Realización de mejoras menores en la interfaz orientada a optimizar la interacción del usuario con el sistema.

Tabla 11: Sprint Backlog Sprint 3

20.5.3.4. Sprint Review (S3)

Al finalizar el sprint, el prototipo permitió simular el registro y visualización de pedidos entre los módulos definidos.

20.5.3.5. Conclusión del Sprint (S3)

Este sprint permitió comprobar el comportamiento del prototipo desde el punto de vista del usuario.

20.5.4. Sprint 4: Ajustes finales y documentación

20.5.4.1. Sprint Planning (S4)

El último sprint se enfocó en realizar ajustes finales al prototipo y elaborar la documentación correspondiente al desarrollo realizado.

20.5.4.2. Épicas (S4)

Épica	Descripción	Prioridad
E9	Ajustes finales del prototipo	1
E10	Documentación técnica	1

Tabla 12: Épicas Sprint 4

20.5.4.3. Sprint Backlog (S4)

Tarea		
Nro	**Nombre**	**Descripción**
13	Corrección de errores menores	Identificación y corrección de fallos detectados durante las pruebas del prototipo.
14	Validación final del prototipo	Revisión general del funcionamiento del sistema con el fin de garantizar el logro de los objetivos establecidos.
15	Elaboración de documentación técnica	Redacción de la documentación correspondiente al desarrollo, estructura y funcionamiento del prototipo.
16	Preparación para presentación	Organización del proyecto y sus evidencias.

Tabla 13: Sprint Backlog Sprint 4

20.5.4.4. Sprint Review (S4)

El prototipo alcanzó un estado funcional estable, permitiendo evidenciar el funcionamiento del sistema manejo de pedidos propuesto.

20.5.4.5. Conclusión del Sprint

Este sprint permitió cerrar el desarrollo del prototipo de forma ordenada y documentada.

CAPÍTULO 6 - DESARROLLO

21.DESARROLLO DE LA APLICACIÓN

21.1. Base de datos

Tras la definición del modelo de base de datos y comprendido el funcionamiento general del sistema, se procede al desarrollo de la aplicación. Para ello, y de acuerdo con la tecnología seleccionada previamente, se utiliza SQL Server como gestor de base de datos, en el cual se realiza la creación de la base de datos y del usuario correspondiente para su administración y acceso.

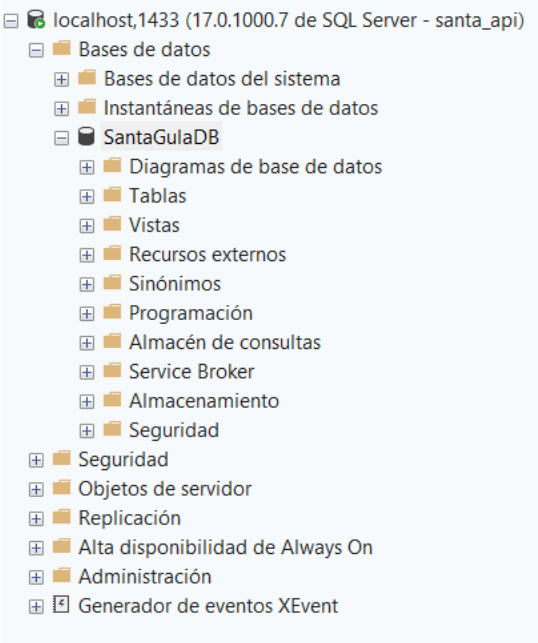
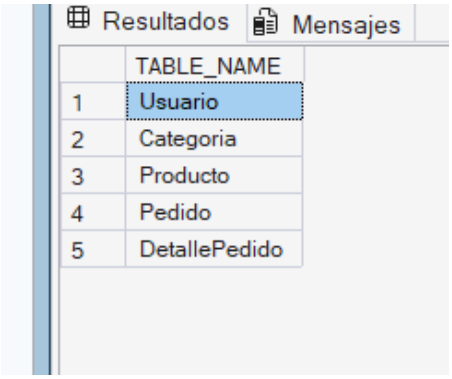


Ilustración 38: Base de Datos Creada

Una vez establecida la arquitectura de la base de datos, procedemos a la fase de implementación mediante la creación de las tablas. Este proceso se realiza siguiendo estrictamente las especificaciones y el esquema lógico que habíamos definido en las etapas previas del diseño.


Resultados Mensajes

	TABLE_NAME
1	Usuario
2	Categoria
3	Producto
4	Pedido
5	DetallePedido

Ilustración 39: Tablas Creadas

La base de datos está estructurada mediante tablas que representan las entidades principales del sistema. La tabla Usuario gestiona los roles del sistema (caja, cocina y administrador), Categoria y Producto almacenan la información del menú, mientras que Pedido y DetallePedido permiten registrar los pedidos generados junto con los productos correspondientes a cada pedido.

Con el objetivo de verificar el correcto funcionamiento del sistema y validar la lógica de las consultas, procederemos a la inserción de un conjunto de datos de prueba. Esta carga inicial nos permitirá realizar las simulaciones necesarias para asegurar que la estructura responde adecuadamente a los requerimientos establecidos.

Resultados | Mensajes

	idUsuario	nombreUsuario	rol	contra
1	1	caja	CAJA	$2b$10$Mm.caLzgXXu0AqbK0hEwyuG2kjjcQ0iWlGpomYJD8...
2	2	cocina	COCINA	$2b$10$QLbZmuUuiEUuaSn7YvXsauxnmUycng5Qs.uD0RelgI...
3	4	admin	ADMIN	$2b$10$9ImEPA2KhEMbOs/z09MbEO8JHUqcMVevL8isvbVfpq...

	idCategoria	nombreCategoria	descripcion	activo
1	1	Almuerzo	Almuerzo Completo	1
2	2	Segundo	Segundo	1
3	3	Sopa	Segundo	1
4	4	Extras	Extras actualizados	1

	idProducto	idCategoria	nombre	precio	activo
1	1	4	Jugo Natural Grande2	2.00	1
2	2	1	AlmuerzoOP2	3.50	1
3	3	2	SegundoOP1	2.75	1
4	4	2	SegundoOP2	2.75	1
5	5	3	Sopa	1.75	1
6	6	3	SopaEspecial	2.00	1
7	7	4	Jugo Natural	1.00	1

	idPedido	idUsuario	nombreCliente	referenciaUbicacion	esUrgente	estadoPago	estadoPedido	total	fechaHoraRegistro
1	1	1	Diego Rosales	Caja del Seguro	0	PENDIENTE	PENDIENTE	5.50	2025-12-29 15:44:27

	idDetalle	idProducto	idPedido	cantidad	precioUnitario	subtotal
1	1	3	1	2	2.75	5.50

Ilustración 40: Datos Ingresados

Con el propósito de realizar pruebas de consistencia, se han empleado datos vinculados al Core del negocio. No obstante, cabe precisar que estos registros son provisionales y no representan la información final, habiendo sido diseñados exclusivamente para validar el comportamiento del sistema en un entorno controlado.

Una vez definidas las entidades que conforman el sistema y creada la organización de las tablas dentro de la base de datos, se establecen las relaciones entre ellas. Este paso es fundamental para asegurar la correcta organización de la información y mantener la coherencia de los datos registrados.

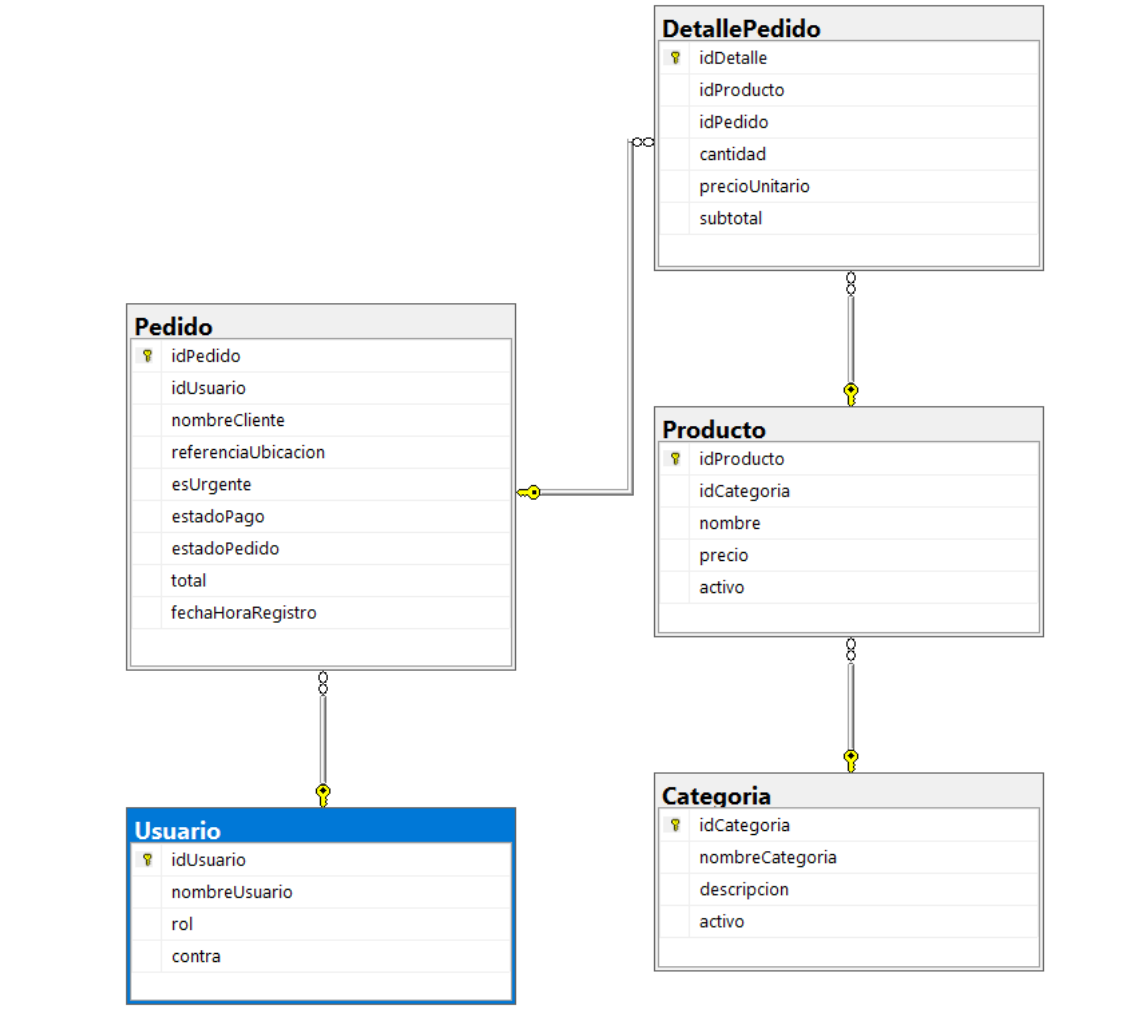


Ilustración 41: Diagrama de SQL Server

El diagrama de relaciones muestra la forma en que las tablas se encuentran vinculadas mediante claves primarias y foráneas. La tabla Usuario se relaciona con la tabla Pedido, permitiendo identificar al responsable de cada pedido registrado. Asimismo, la tabla Pedido se asocia con DetallePedido para representar los productos incluidos en cada solicitud, mientras que la tabla Producto se vincula con Categoria para clasificar los productos del menú. Esta estructura relacional garantiza la integridad referencial y facilita una gestión eficiente y consistente de los pedidos dentro del sistema.

En conjunto, la base de datos proporciona una estructura sólida y coherente que respalda el adecuado funcionamiento del sistema de gestión de pedidos planteado.

21.2. Backend

Tras concluir la configuración de la base de datos, el siguiente paso consistió en el desarrollo del backend de la aplicación. Para ello, se utilizó Node.js, de acuerdo con la selección de herramientas definida en las etapas iniciales del proyecto.

21.2.1. Arquitectura del Backend

El backend del sistema fue desarrollado siguiendo el patrón arquitectónico Modelo-Vista-Controlador (MVC), el cual permite separar de manera clara las responsabilidades de cada componente. Esta organización facilita el mantenimiento del código, mejora la legibilidad del proyecto y permite una evolución ordenada del sistema conforme se incorporan nuevas funcionalidades.

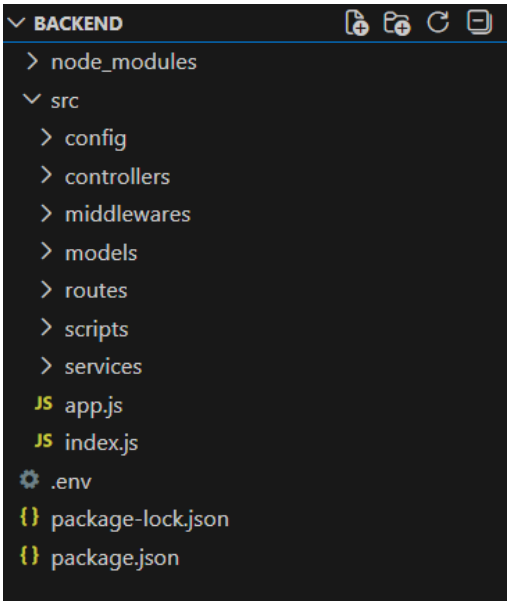


Ilustración 42: Patrón MVC

Como se observa en la ilustración, el backend se organiza en distintos módulos que responden a la arquitectura MVC.

21.2.2. Controladores

Los controladores constituyen el núcleo de la lógica del sistema, ya que son responsables de procesar las solicitudes recibidas desde el cliente, aplicar las reglas de negocio y coordinar la comunicación con los modelos. A través de los controladores se gestionan operaciones como el registro de pedidos, la actualización de estados y la consulta de información necesaria para el funcionamiento del prototipo.

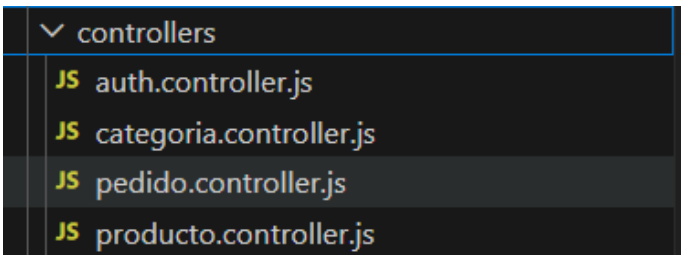


Ilustración 43: Controladores Backend

21.2.3. Modelos

Los modelos representan las entidades principales del sistema y su relación con la base de datos. Estos componentes encapsulan la estructura de los datos y permiten realizar operaciones de consulta, inserción y actualización de información de manera controlada. Su implementación garantiza la consistencia de los datos y facilita la interacción entre el backend y la base de datos.

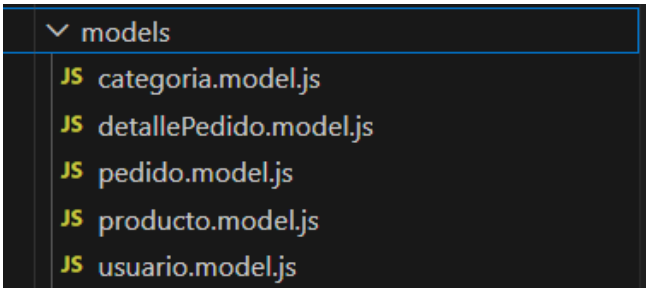


Ilustración 44: Modelos Backend

21.2.4. Rutas y APIs

Las rutas definen los puntos de acceso al sistema mediante servicios web tipo REST, permitiendo la comunicación entre el frontend y el backend. A través de estas rutas se exponen las APIs necesarias para gestionar los pedidos, productos y usuarios del sistema. Cada endpoint fue diseñado para responder a solicitudes específicas, siguiendo una estructura clara y coherente.

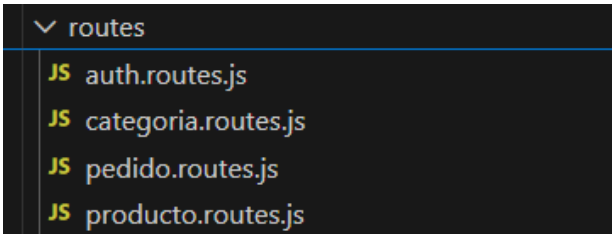


Ilustración 45: Rutas y APIs Backend

21.2.5. Middlewares y seguridad

Para fortalecer el flujo de la aplicación y garantizar un acceso controlado, se implementaron middlewares encargados de gestionar la autenticación, la autorización por roles y el manejo de errores. Estos componentes permiten validar las solicitudes antes de ser procesadas, asegurando que únicamente los usuarios autorizados puedan acceder a las funcionalidades correspondientes del sistema.

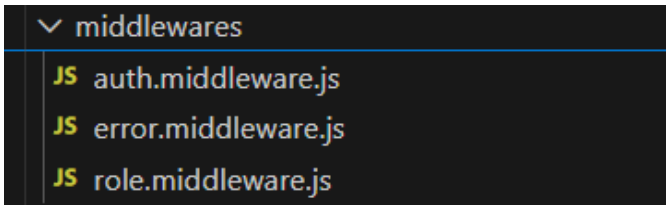


Ilustración 46: Middlewares y Seguridad Backend

21.2.6. Validación del Backend

Con el objetivo de verificar el correcto funcionamiento del backend, se realizaron pruebas sobre las APIs desarrolladas mediante la herramienta Postman. Estas pruebas permitieron validar el comportamiento de los endpoints, comprobar la correcta gestión de las solicitudes y asegurar que las respuestas generadas por el sistema se ajusten a lo esperado dentro del contexto del prototipo.

A continuación, se presentan las evidencias obtenidas mediante la ejecución de solicitudes HTTP en Postman, donde se muestra el correcto funcionamiento de los endpoints definidos para el sistema.

21.2.6.1. Autenticación

21.2.6.1.1. Login ADMIN

Se validó el proceso de autenticación mediante JWT para usuarios con rol administrador

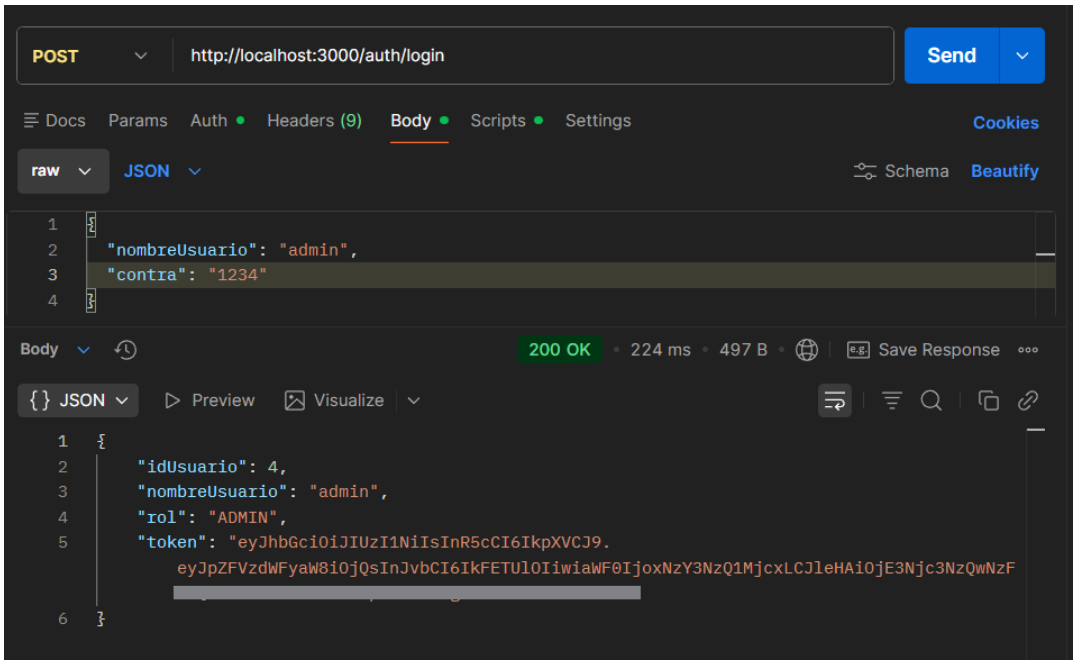


Ilustración 47: Autenticación ADMIN

21.2.6.1.2. Login CAJA

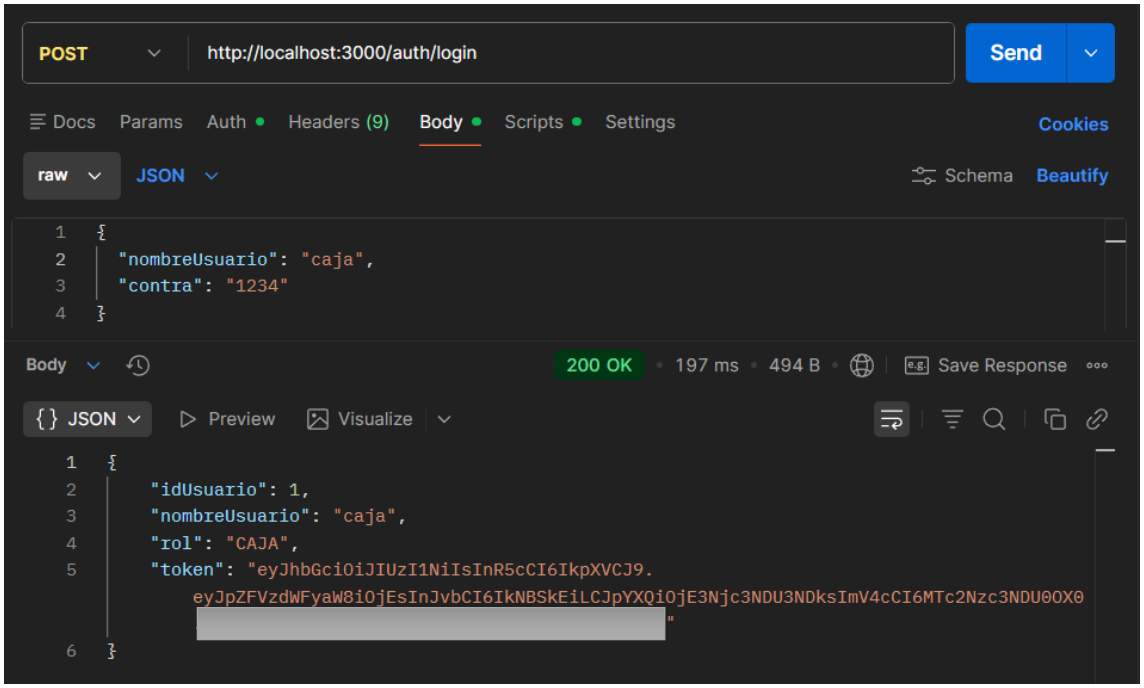


Ilustración 48: Autenticación CAJA

21.2.6.1.3. Login COCINA

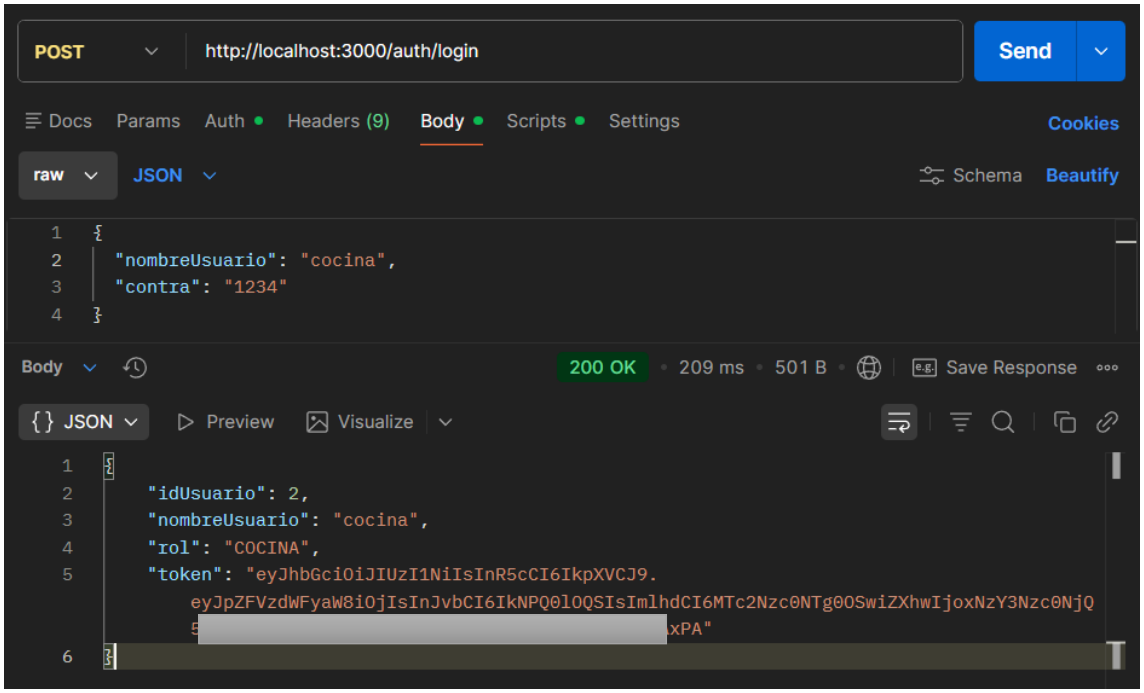


Ilustración 49: Autenticación COCINA

21.2.6.2. Categorías

21.2.6.2.1. Crear categoría

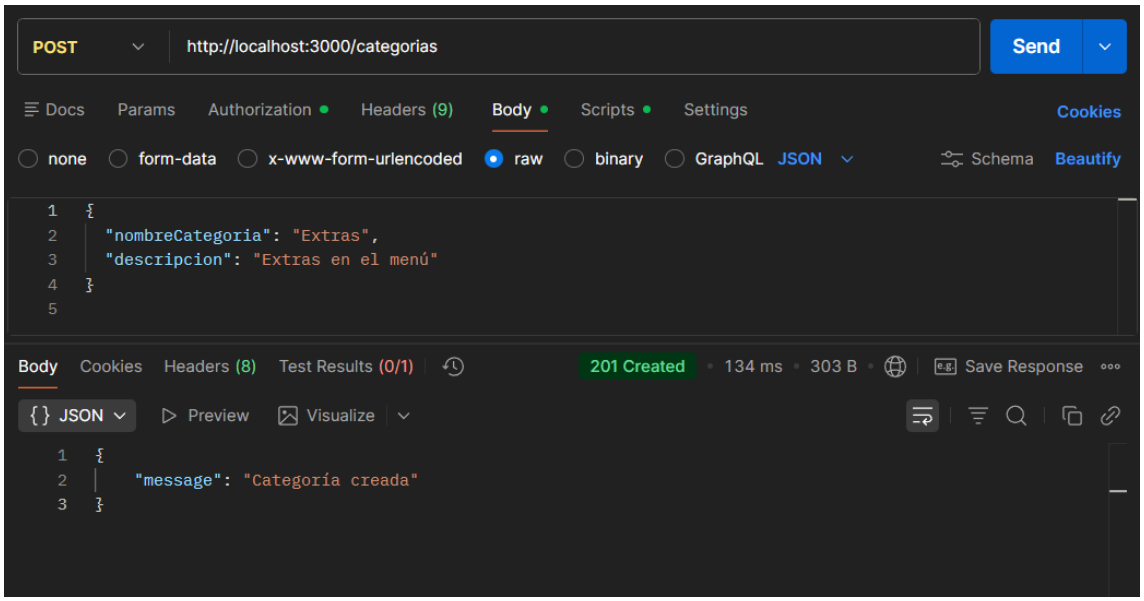


Ilustración 50: Crear Categoría

21.2.6.2.2. Actualizar categoría

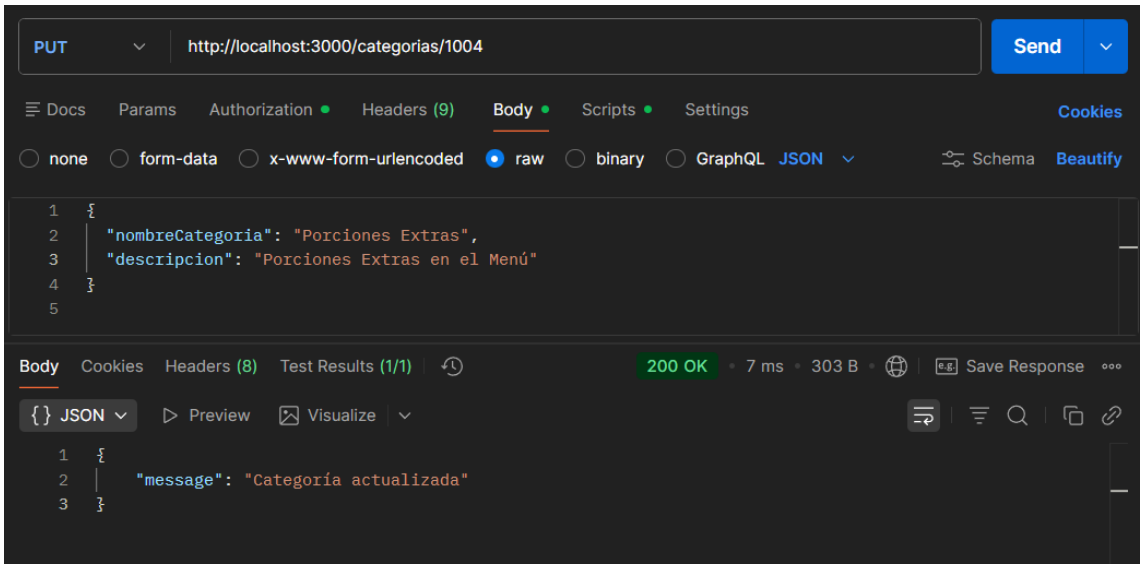


Ilustración 51: Actualizar Categoría

21.2.6.2.3. Cambiar Estado

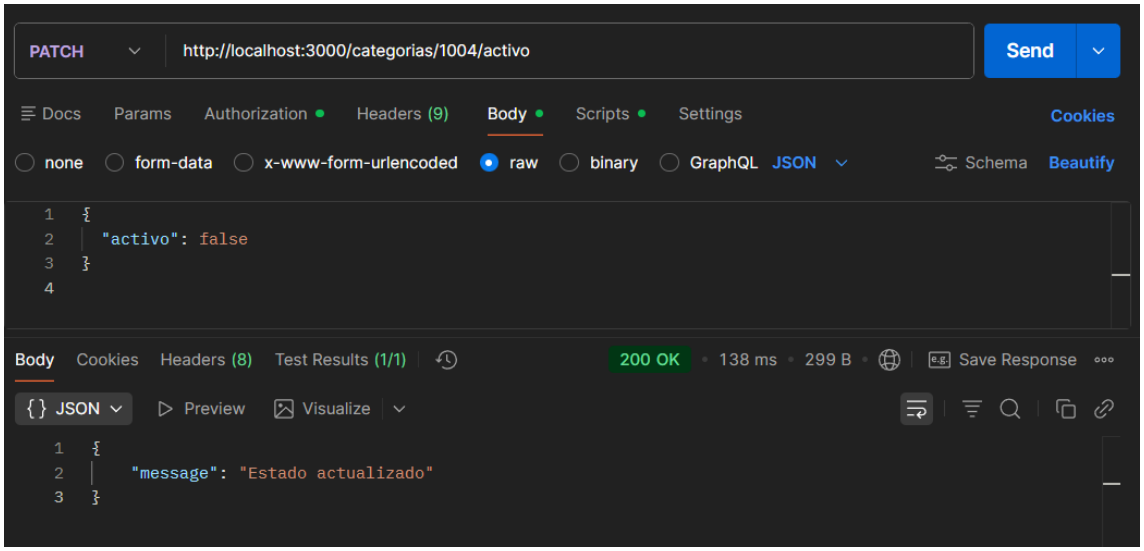


Ilustración 52: Cambiar Estado

21.2.6.2.4. Listar Categorías

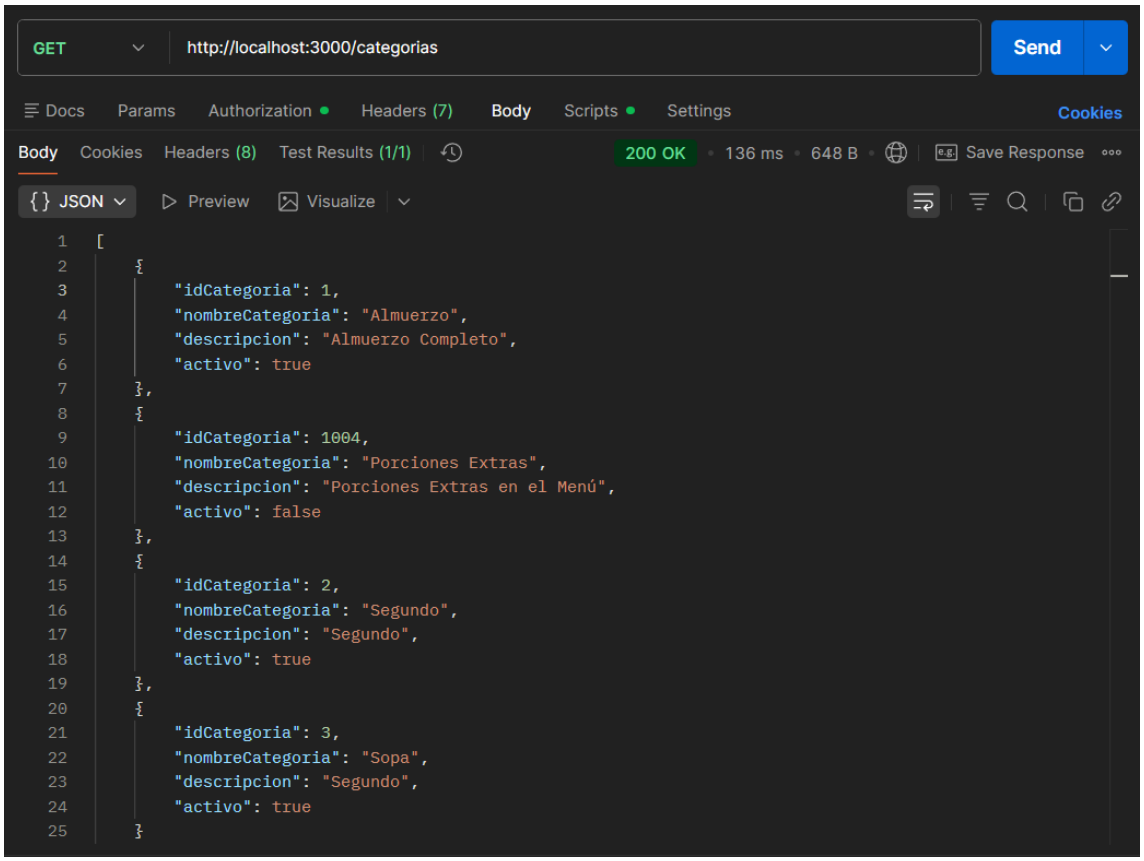


Ilustración 53: Categorías Listadas

21.2.6.3. Productos

21.2.6.3.1. Crear producto

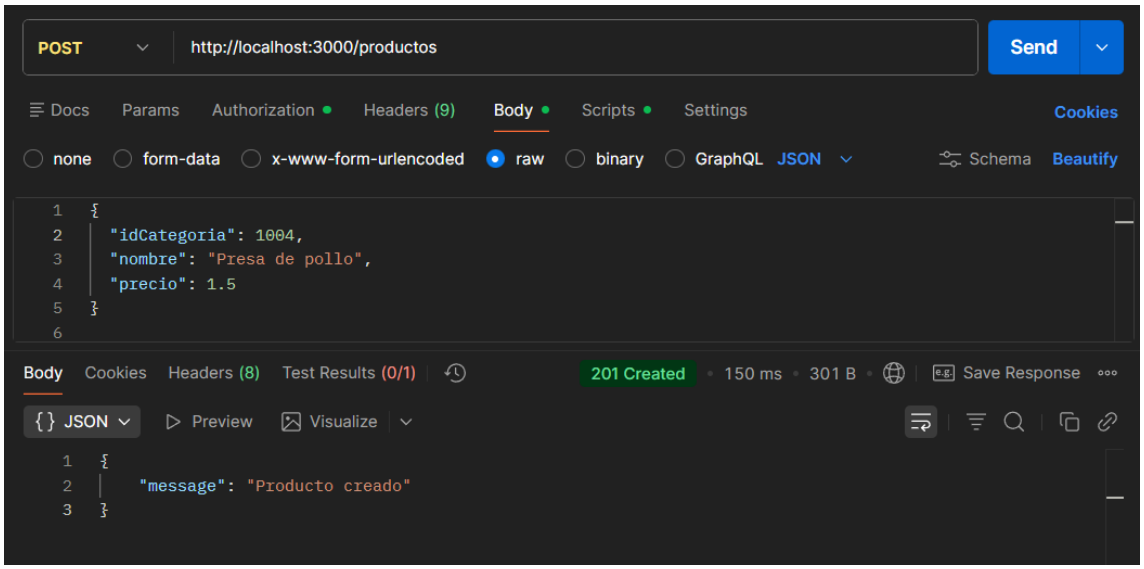


Ilustración 54: Crear Producto

21.2.6.3.2. Actualizar Producto

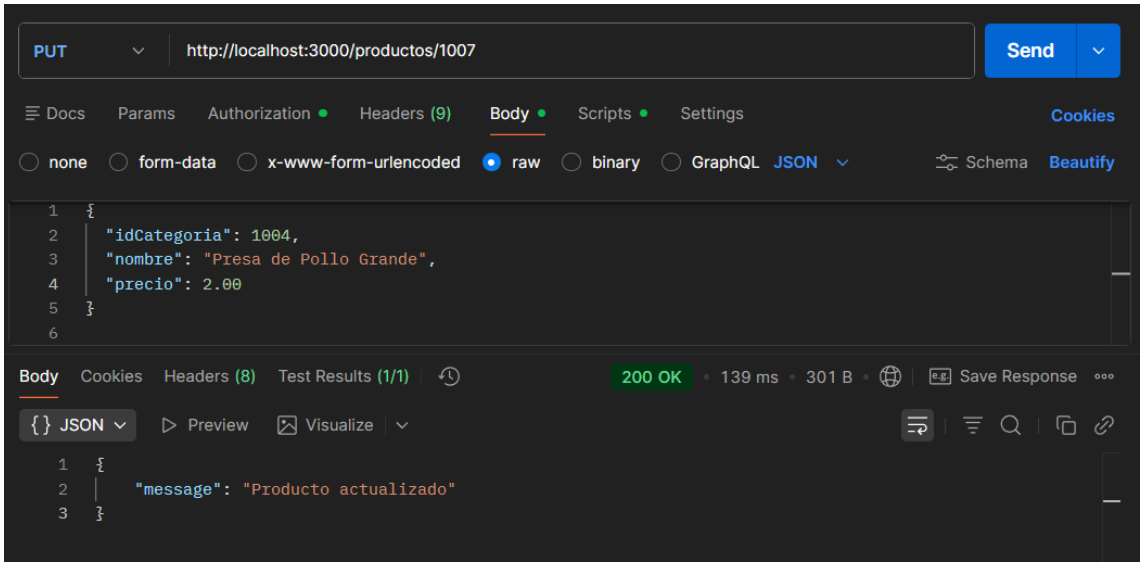


Ilustración 55: Actualizar Producto

21.2.6.3.3. Desactivar/Activar Producto

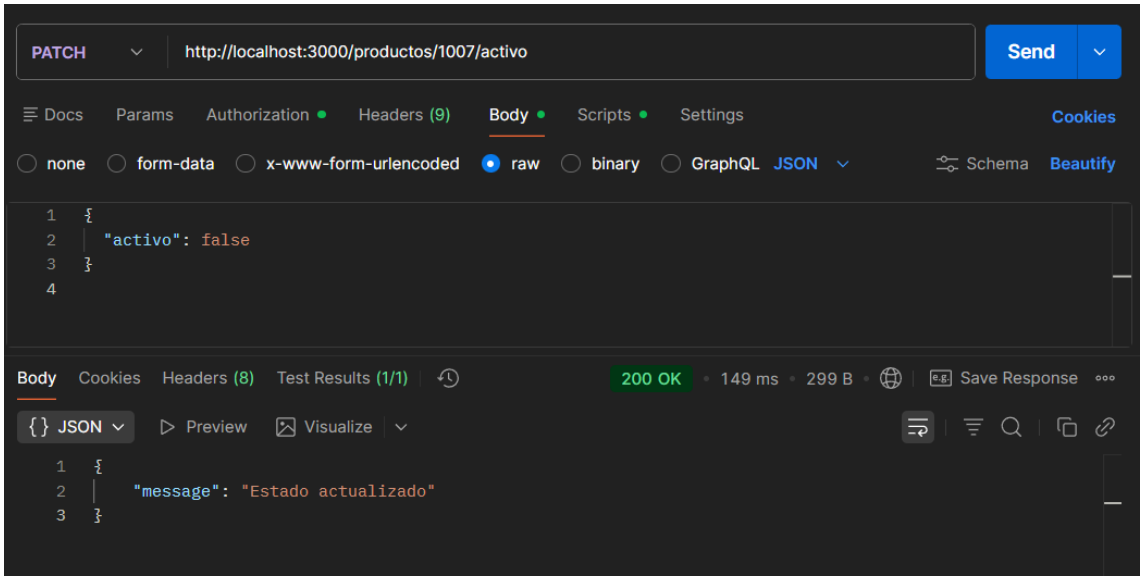


Ilustración 56: Desactivar Productos

21.2.6.3.4. Listar Productos

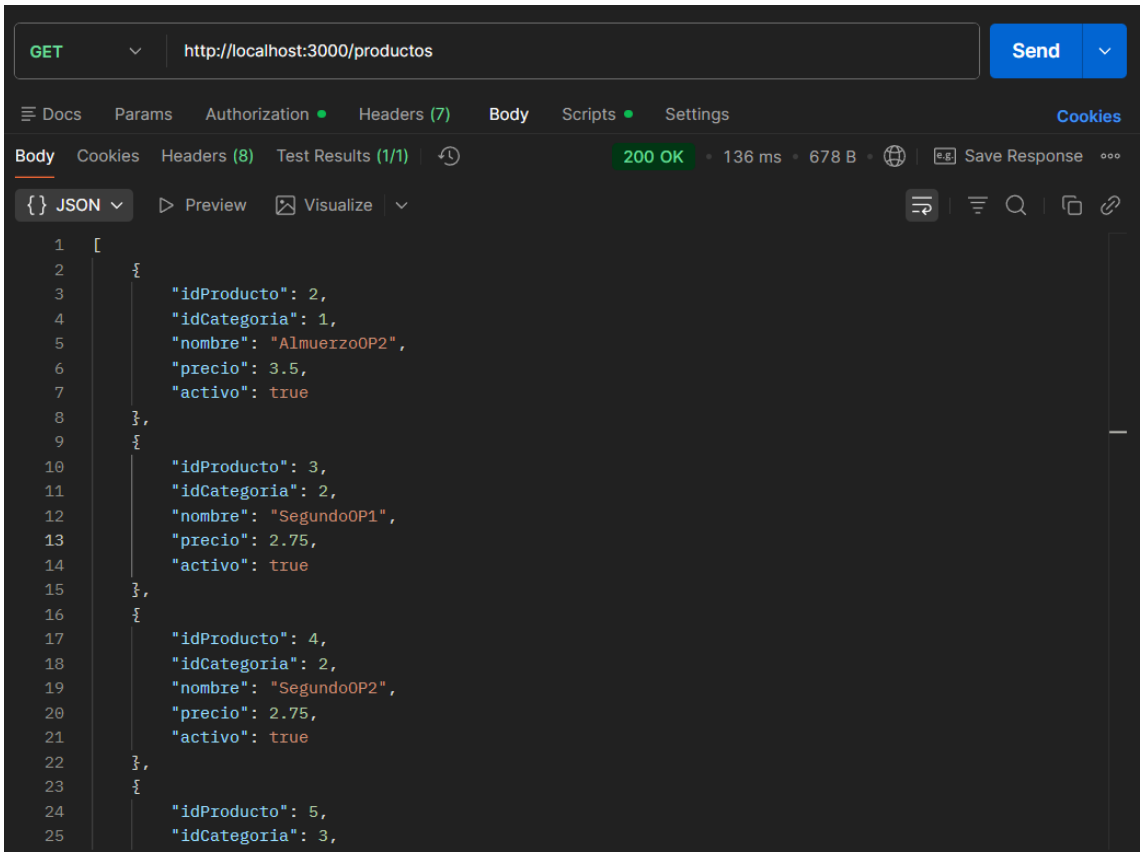


Ilustración 57: Productos Listados

21.2.6.4. Pedidos

21.2.6.4.1. Crear Pedido (CAJA)

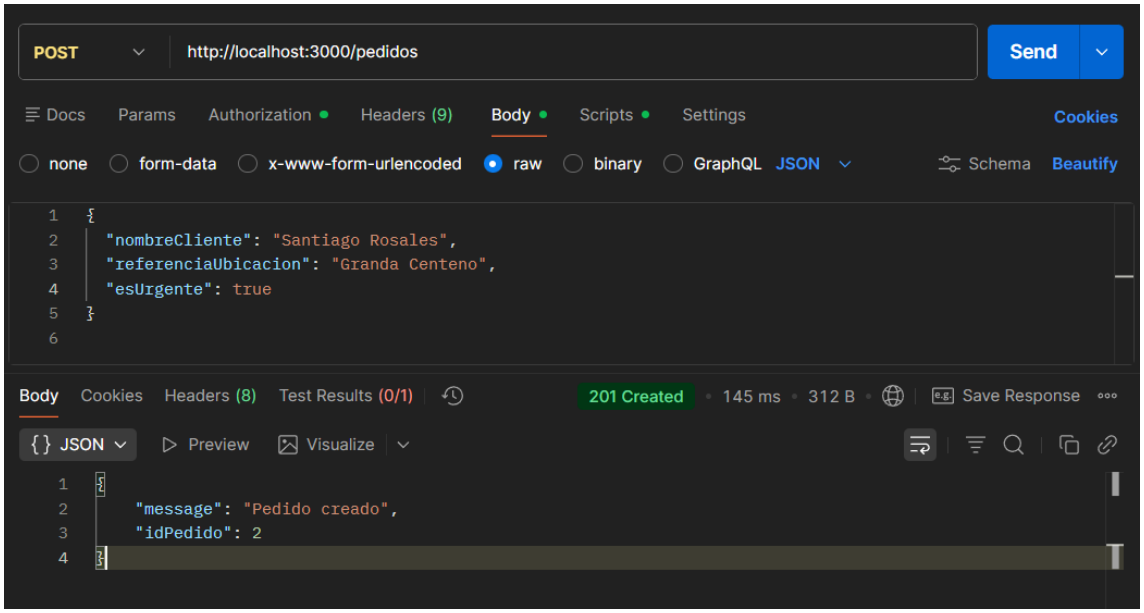


Ilustración 58: Crear Pedido (CAJA)

21.2.6.4.2. Agregar detalle al pedido

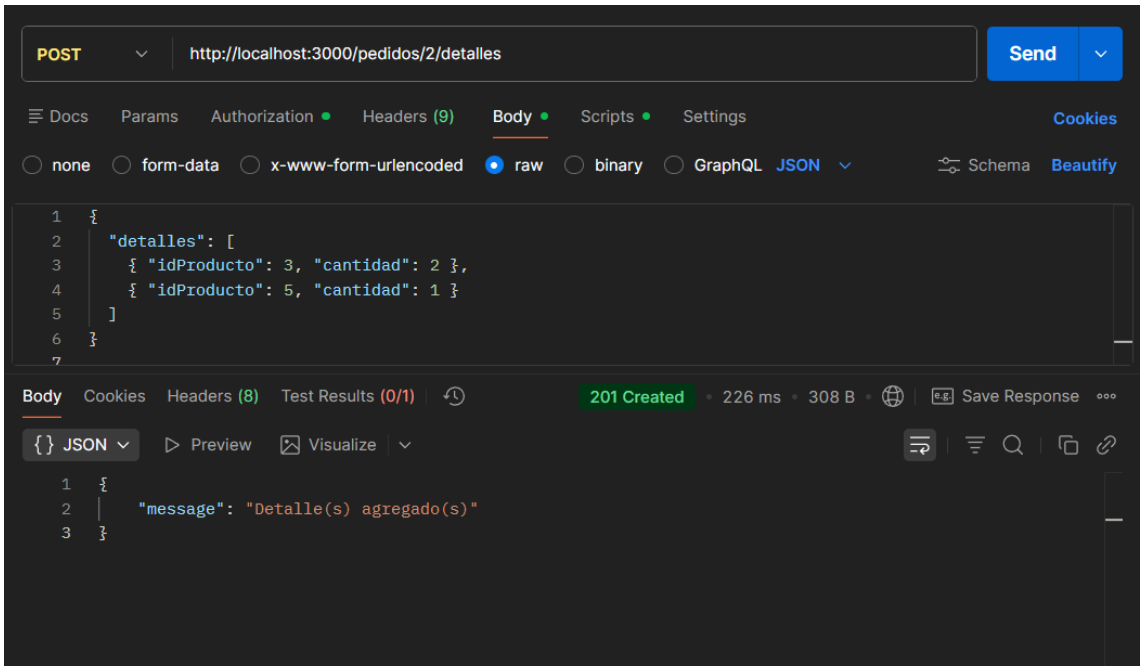


Ilustración 59: Agregar Detalle

21.2.6.4.3. Ver pedido completo

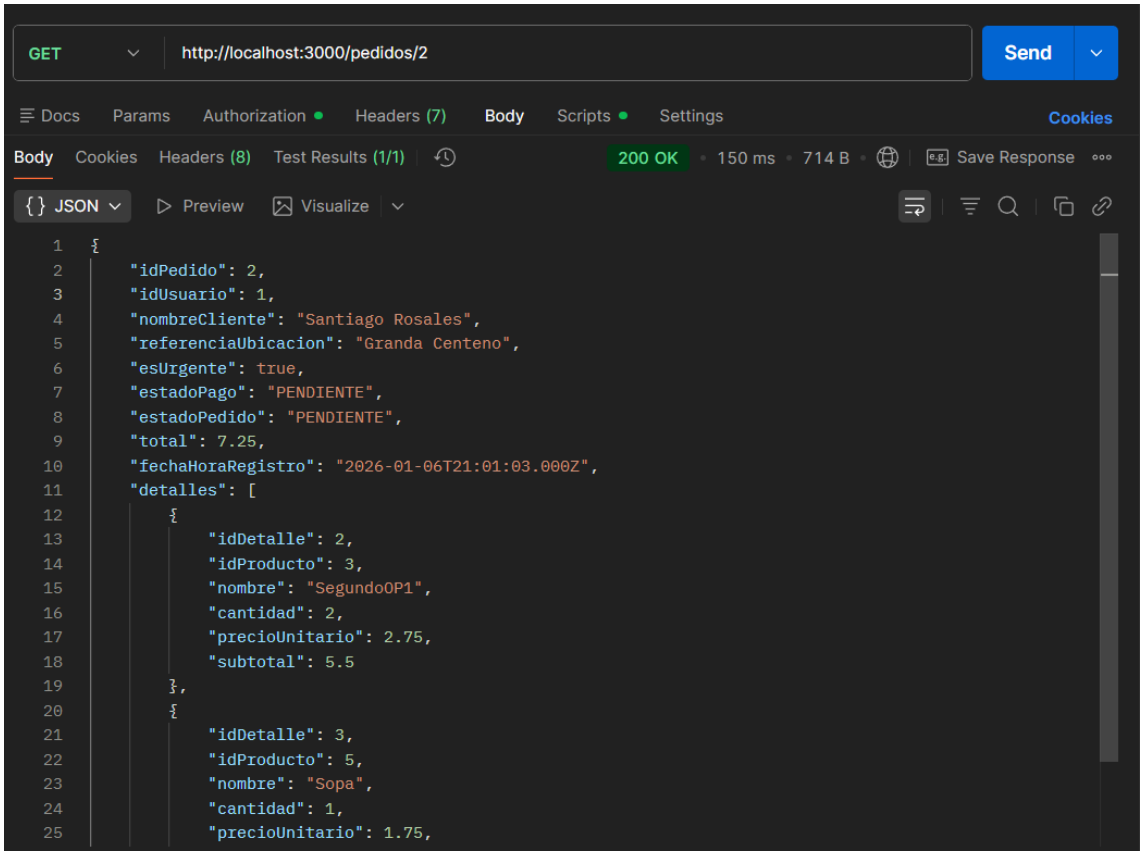


Ilustración 60: Pedido Completo

21.2.6.4.4. Listar Pedidos (COCINA)

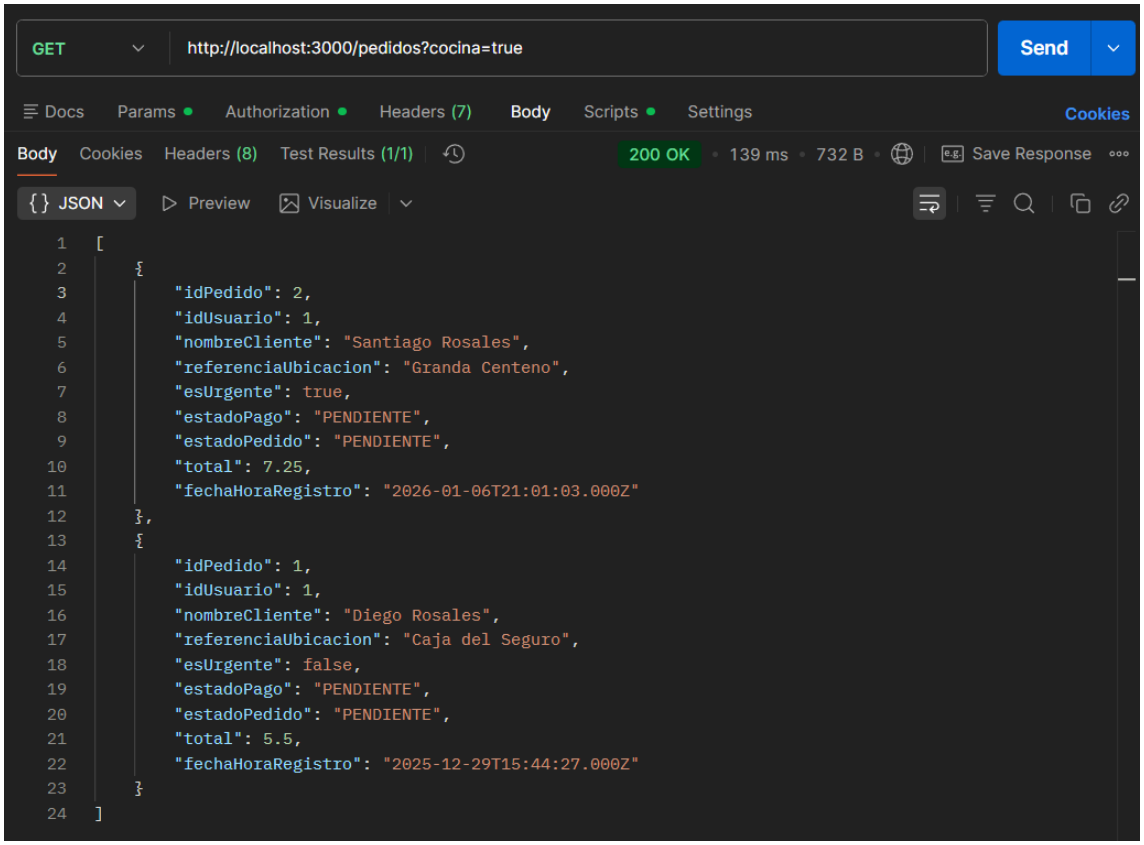


Ilustración 61: Pedidos Listados (COCINA)

21.2.6.4.5. Cambiar Estado (COCINA)

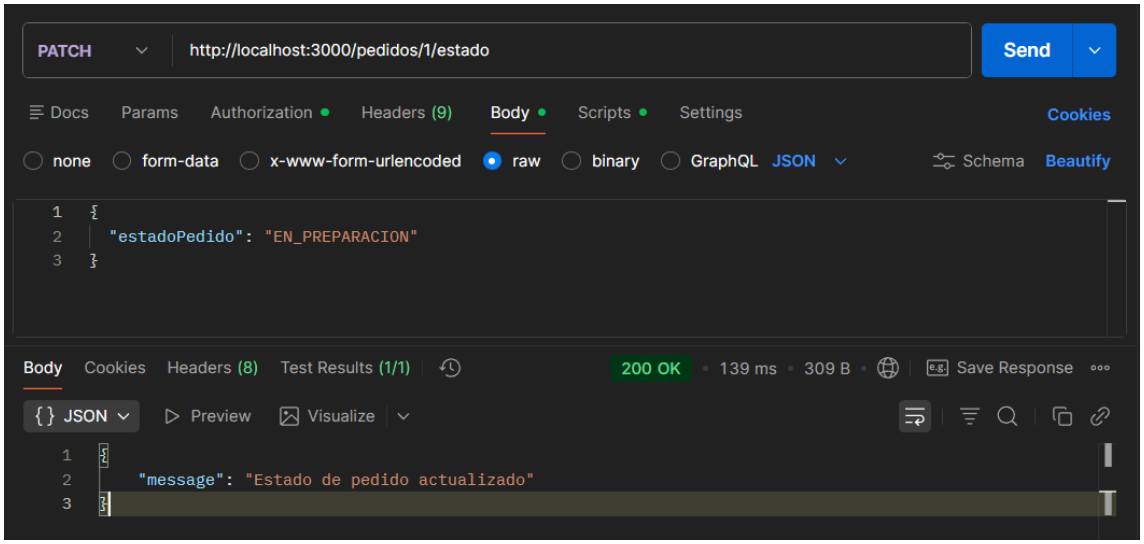


Ilustración 62: Estado Pedido Actualizado (En_PREPARACION)

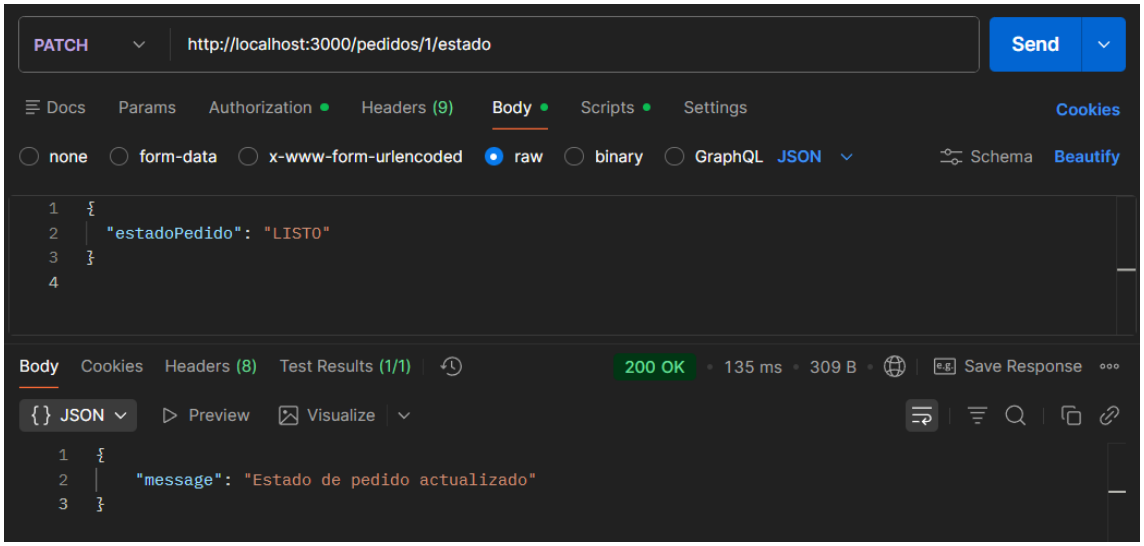


Ilustración 63: Estado Pedido Actualizado (LISTO)

21.2.6.4.6. Cambiar estado de pago (CAJA)

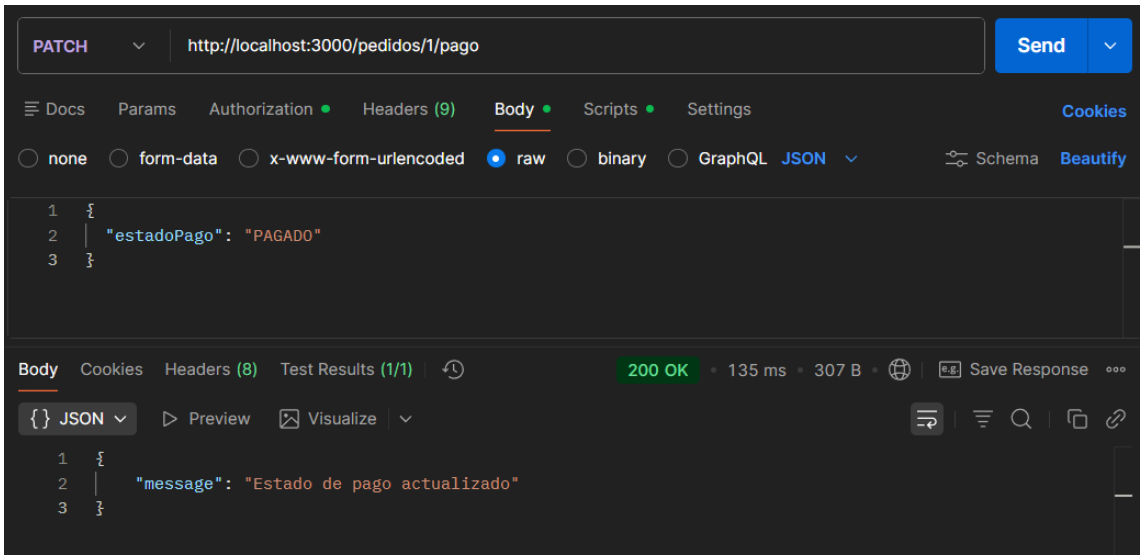


Ilustración 64: Pago Actualizado

21.2.6.4.7. Marcar como entregado

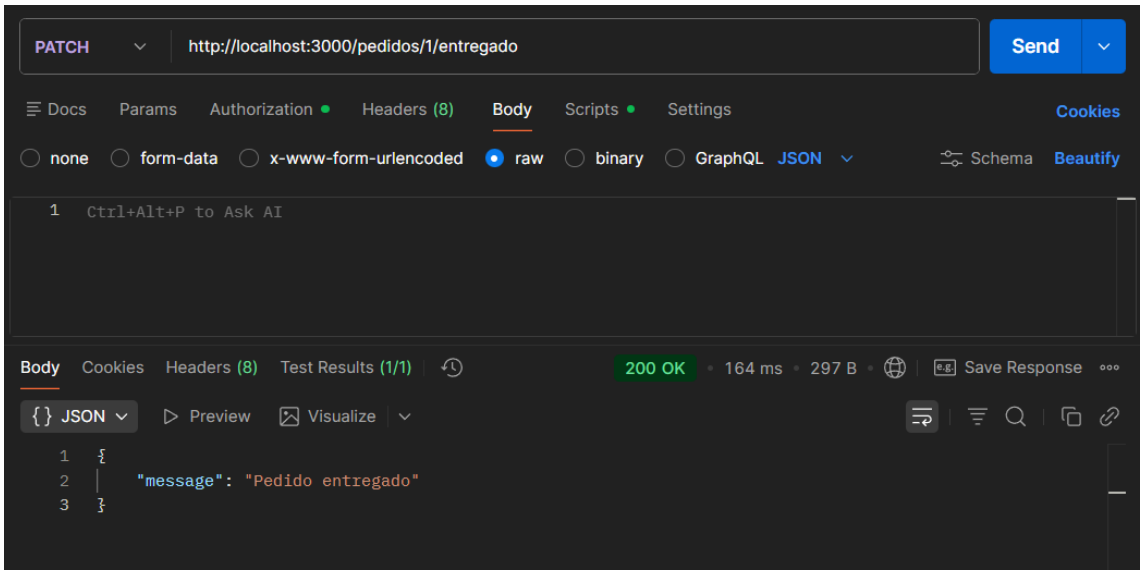


Ilustración 65: Pedido Entregado

21.2.6.4.8. Cancelar pedido

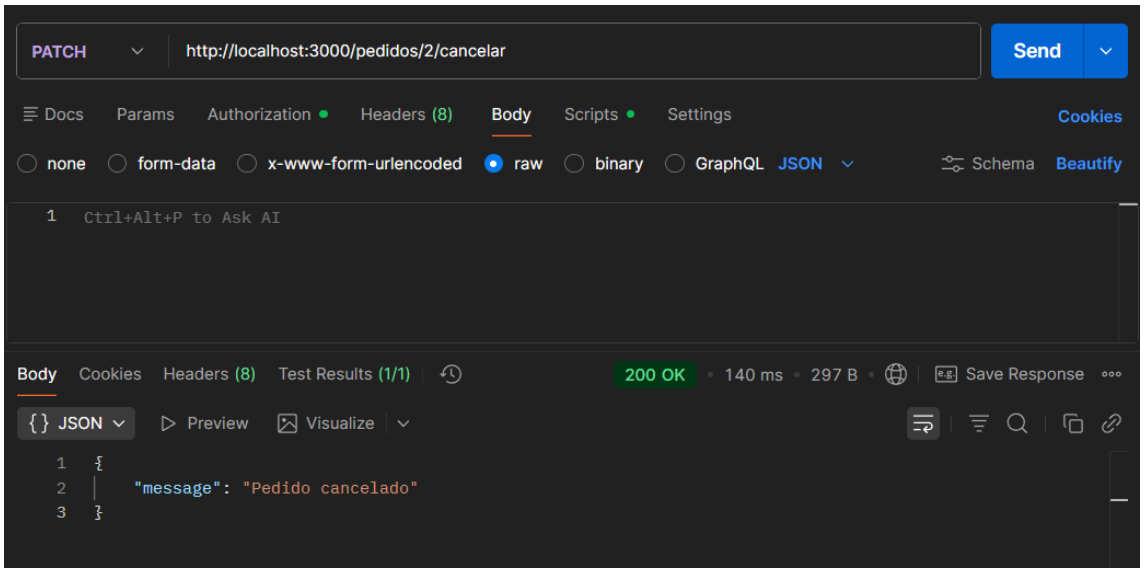


Ilustración 66: Pedido Cancelado

Los resultados obtenidos evidencian que el backend satisface los requerimientos funcionales definidos, garantizando integridad de datos, control de accesos y correcta trazabilidad del flujo de pedidos.

21.3. Frontend

Una vez concluido el desarrollo del backend junto con la correcta configuración de la base de datos se dio inicio a la implementación del frontend considerando como base el diseño del prototipo que fue previamente validado por el restaurante Santa Gula, en esta fase se adoptó el framework Angular como tecnología principal, además se incorporaron distintas librerías orientadas a fortalecer la interacción con el usuario y a elevar la calidad visual de la aplicación, permitiendo que la interfaz resultante mantenga coherencia con los lineamientos definidos y responda adecuadamente a los requerimientos funcionales establecidos durante las etapas iniciales del proyecto.

21.3.1. Estructura general del Frontend

El frontend del prototipo se encuentra estructurado mediante componentes que representan las principales vistas del sistema orientadas a los módulos de caja, cocina y administración donde cada uno cumple una función específica dentro de la aplicación. Esto facilita el uso adecuado de las funcionalidades definidas y asegura una correcta experiencia de navegación.

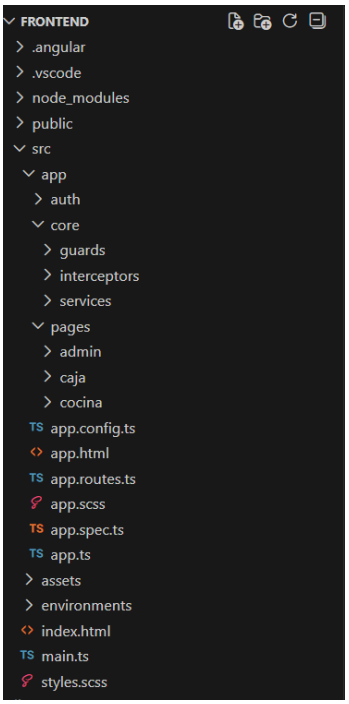


Ilustración 67: Estructura Frontend

21.3.2. Vistas del sistema

21.3.2.1. Vista de inicio / login

En esta vista el usuario ingresa sus credenciales para acceder al sistema, permitiendo validar su rol y habilitar las funcionalidades correspondientes.

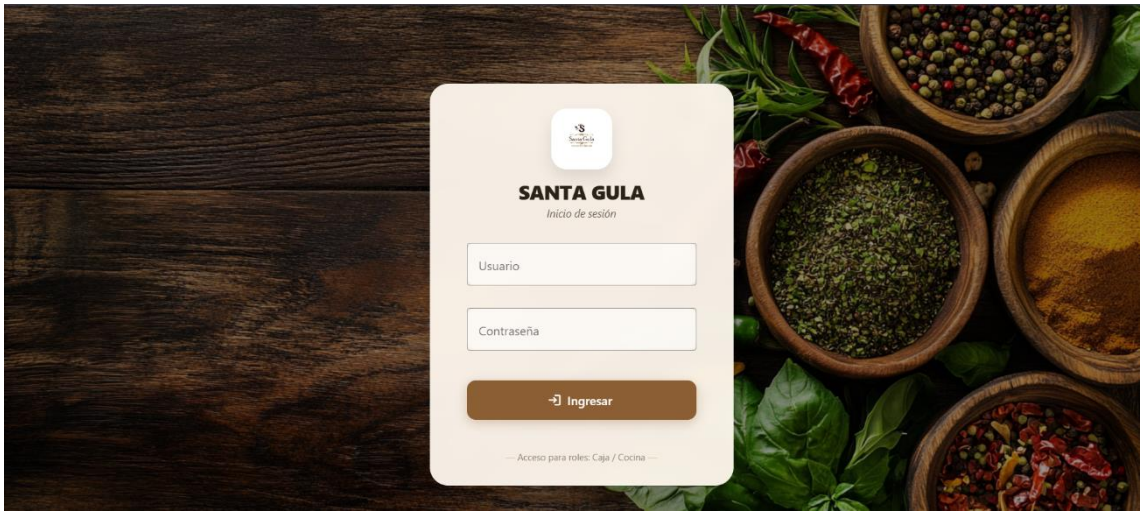


Ilustración 68: Vista de Login

21.3.2.2. Vista de caja

El módulo de caja permite registrar los pedidos del cliente, seleccionar prioridad, estado del pago, seleccionar los productos del menú y enviar la solicitud para su procesamiento, además llevar un registro de los pedidos activos diarios y un historial de pedidos. Y al enviar el pedido a caja, el sistema descarga un ticket para imprimir.

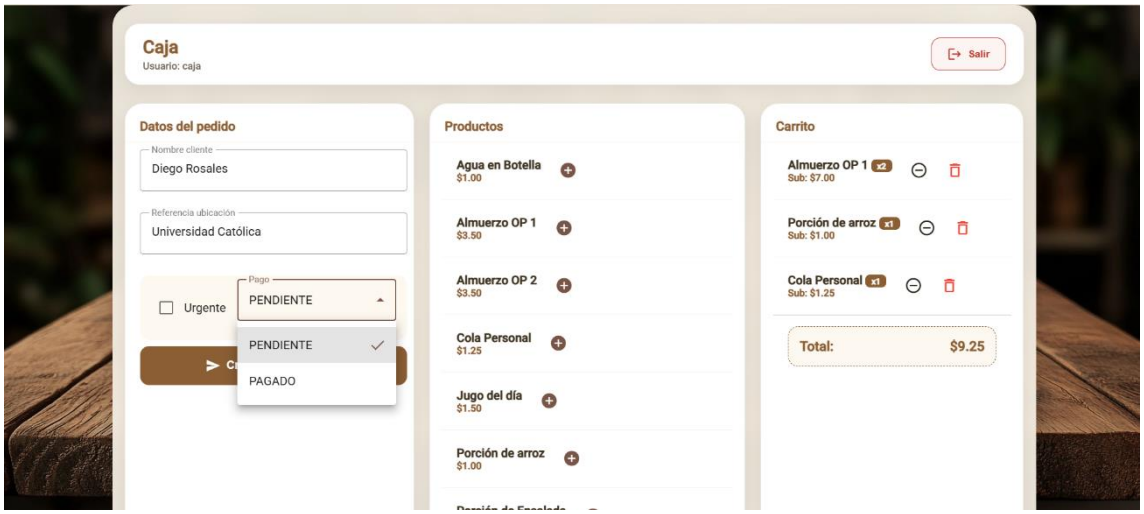


Ilustración 69: Vista Registro de Pedido

Ilustración 70: Pedidos del Día e Historial de Pedidos

SANTA GULA

Ticket de pedido

Fecha: 16/01/2026 09:48
Pago: PAGADO
Cliente: Diego Rosales
Ubic.: Universidad Católica

DETALLE
Almuerzo OP 1
x2 $3.50 Sub: $7.00
Porción de arroz
x1 $1.00 Sub: $1.00
Cola Personal
x1 $1.25 Sub: $1.25

TOTAL: $9.25

Gracias por su compra

Ilustración 71: Ticket de Pedido

21.3.2.3. Vista de cocina

En esta vista se muestran los pedidos registrados en tiempo real, permitiendo al personal de cocina visualizar el estado de cada uno y gestionar su preparación.

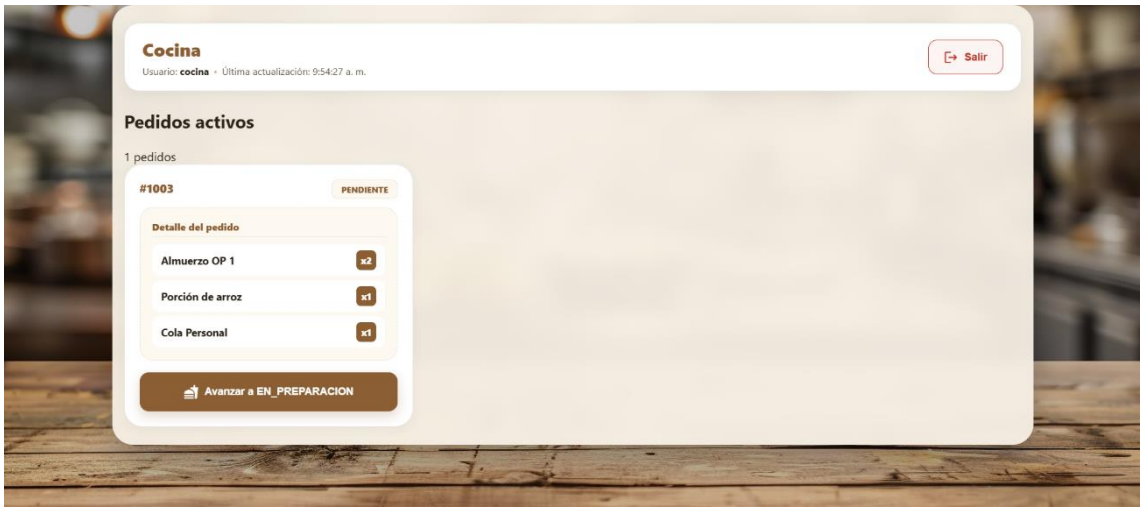


Ilustración 72: Vista Cocina

21.3.2.4. Vista Administrador

En esta vista se administra los usuarios, productos y categorías del sistema, para poder agregar, eliminar o deshabilitar dependiendo la necesidad del administrador del sistema.

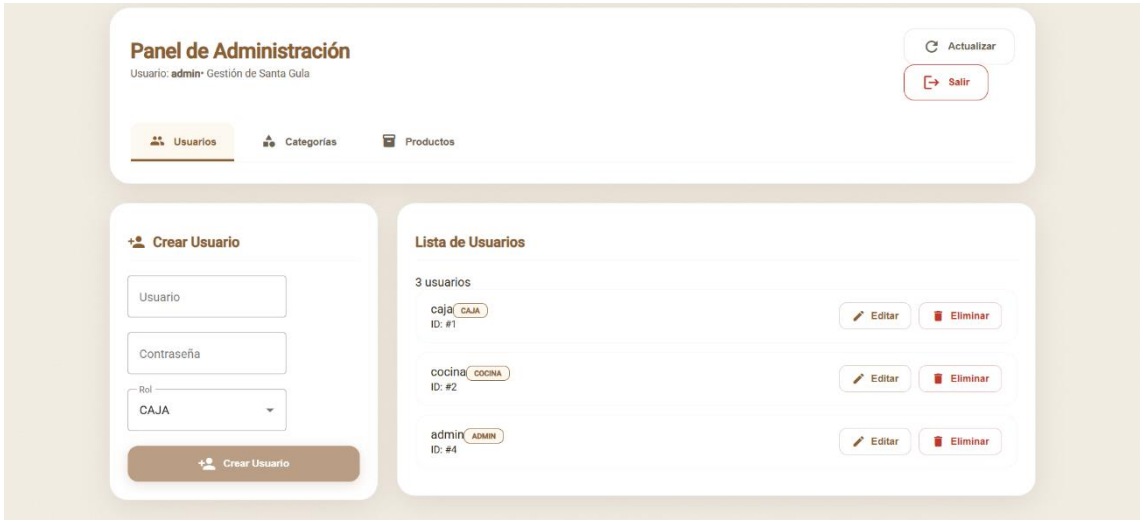


Ilustración 73: Vista Administrador (Usuarios)

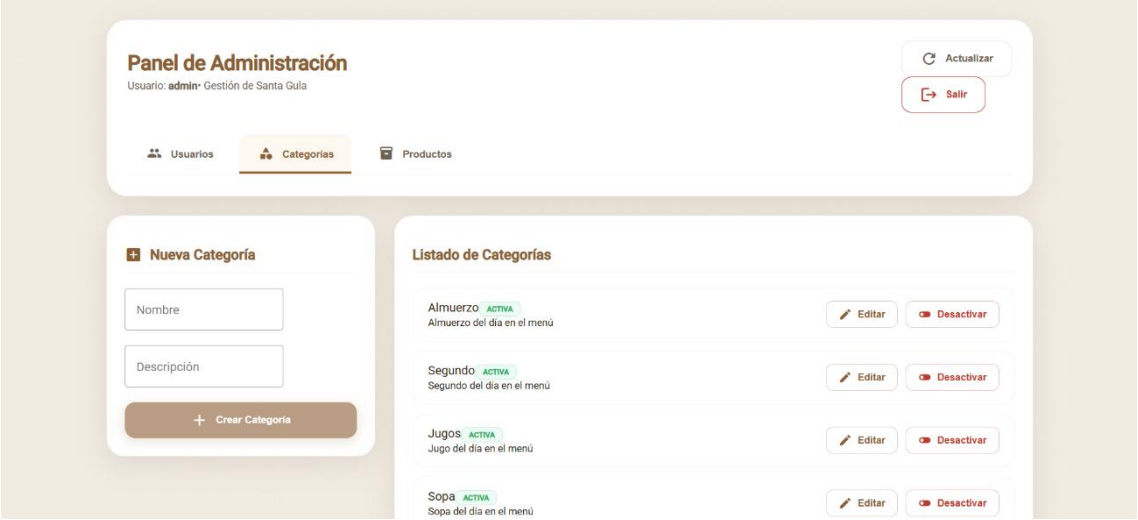


Ilustración 74: Vista Administrador (Categorías)

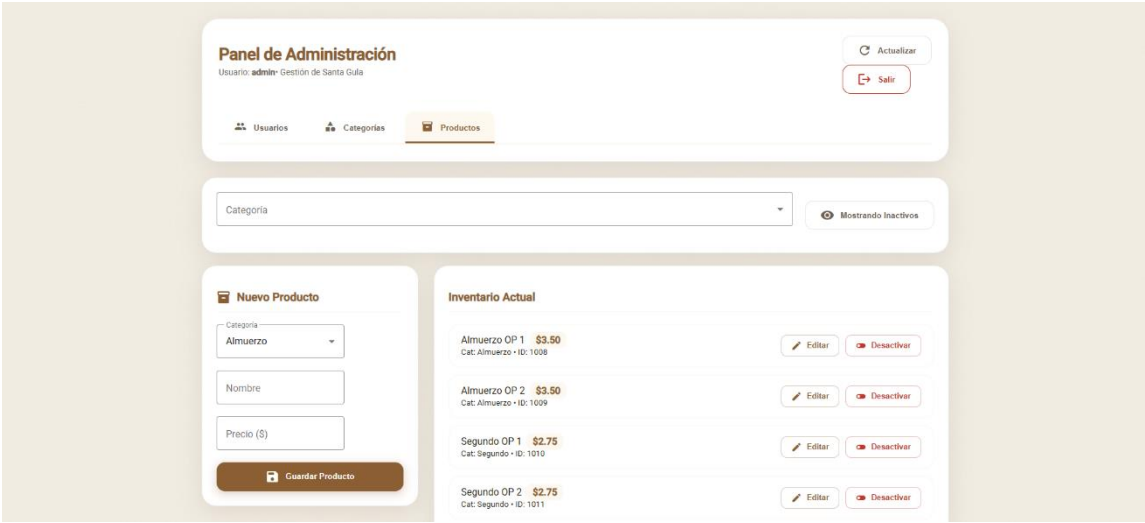


Ilustración 75: Vista Administrador (Productos)

En estas pantallas se evidencia el funcionamiento del sistema al reflejar de manera clara cada una de las funcionalidades implementadas, permitiendo observar el flujo completo de las operaciones definidas para el prototipo y facilitando la comprensión del comportamiento de la aplicación desde el punto de vista del usuario final al mismo tiempo que se valida la correcta integración entre los distintos módulos en el sistema.

21.3.3. Flujo básico de interacción

El flujo de interacción del frontend inicia con la autenticación del usuario, seguido del acceso al módulo correspondiente según su rol. Desde la caja se registran los pedidos, los cuales son enviados al sistema para su visualización en el módulo de cocina, facilitando una comunicación clara y estructurada entre ambos actores. Desde el módulo de Administrador se crean, editan o eliminan usuarios, además se administran las categorías y productos.

22. RESULTADOS

El desarrollo del prototipo del sistema Santa Gula permitió comprobar la viabilidad de una solución en la gestión de pedidos para llevar al lograr la implementación de un flujo funcional entre los módulos de caja, cocina y administrador. El backend desarrollado respondió de manera adecuada a las solicitudes realizadas por el sistema permitiendo el registro y la consulta de pedidos junto con la correcta persistencia de la información en la base de datos, este comportamiento fue verificado mediante pruebas directas a las APIs lo que evidenció un correcto desempeño acorde a los requerimientos definidos.

En el frontend se logró interfaces claras facilitando la interacción de los usuarios con el sistema durante las pruebas, la integración entre las vistas y la lógica del backend permitió una experiencia fluida e intuitiva para los involucrados, por lo que, los resultados obtenidos demuestran que el prototipo cumple con los objetivos planteados.

Desde la perspectiva del negocio, el prototipo permite evidenciar una mejora potencial en la organización del proceso de gestión de pedidos para llevar. Durante las pruebas realizadas se observó que el sistema facilita la comunicación entre caja y cocina, lo que podría contribuir a reducir errores y mejorar la coordinación interna.

23. CONCLUSIONES Y RECOMENDACIONES

23.1. Conclusiones

- El prototipo desarrollado demuestra la viabilidad técnica de un sistema web orientado a la gestión de pedidos para llevar en el restaurante Santa Gula.
- La metodología Scrum permitió organizar el desarrollo de manera ordenada y lograr un avance progresivo del sistema.
- El uso de Node.js, SQL Server y Angular proporcionó una estructura adecuada para la implementación del prototipo.
- Las pruebas realizadas confirmaron el correcto funcionamiento del flujo de pedidos entre los módulos de caja y cocina.
- La implementación de una arquitectura basada en el patrón Modelo-Vista-Controlador contribuyó a una mejor distribución de responsabilidades y a una mayor organización del código del sistema..
- La validación de las APIs mediante pruebas permitió comprobar la correcta comunicación entre el backend y la base de datos dentro del prototipo.
- El desarrollo del frontend permitió la interacción de forma adecuada a los usuarios con el sistema.
- Desde el punto de vista del negocio, el prototipo evidencia una solución viable para mejorar la organización del proceso de pedidos para llevar.
- La experiencia de desarrollo del proyecto permitió consolidar conocimientos prácticos en tecnologías web y metodologías ágiles, fortaleciendo las competencias técnicas aprendidas.

23.2. Recomendaciones

- Considerar la integración del sistema con otros procesos del negocio manteniendo una arquitectura escalable.
- Ampliar el prototipo incorporando nuevas funcionalidades en una posible futura implementación.
- Reforzar los mecanismos de seguridad en caso de llevar el sistema a producción.
- Realizar pruebas con usuarios reales para mejorar la usabilidad del sistema.
- Implementar mecanismos de control de errores más detallados para mejorar la estabilidad del sistema.
- Optimizar la interfaz de usuario considerando criterios de accesibilidad y experiencia de uso.
- Incorporar registros de actividad que permitan monitorear el funcionamiento del sistema en un entorno productivo.
- Evaluar el desempeño del sistema en escenarios con mayor carga de pedidos para identificar posibles mejoras de rendimiento.
- Documentar y estandarizar los procesos técnicos del sistema para facilitar su mantenimiento y escalabilidad.

24.BIBLIOGRAFÍA

Sommerville, I. (2005). Requerimientos del software. Ingeniería del software, 7a ed., PEARSON EDUCACIÓN, Madrid, SPA, 109-110.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). Información Básica de Scrum (The Scrum Primer). In Scrum Training Institute. https://goodagile.com/scrumprimer/scrumprimer_es.pdf

Marqués, M. (2009). Bases de datos. In Meta Biblioteca - Haciendo Ciencia. https://libros.metabiblioteca.org/entities/publication/33223a6a-b4fc-4d78-8f32-6debec4e60cd

Pressman, R. S. (2014). Ingeniería del software: Un enfoque práctico (7.ª ed.). McGraw-Hill.

Node JS. (2015). Node.js - About Node.js. Nodejs.org. https://nodejs.org/en/about

Sommerville, I. (2016). Ingeniería del software (10.ª ed.). Pearson Educación.

Garzón, Á. (2019). Modelo Vista Controlador. Repositorio - Fundación Universitaria Konrad Lorenz. https://repositorio.konradlorenz.edu.co/entities/publication/c56b5001-00bb-461e-b654-3abb59e3c225

Martínez, R., & López, S. (2020). Tecnologías de la información aplicadas a la industria alimentaria. Alfaomega.

Laudon, K. C., & Laudon, J. P. (2020). Sistemas de información gerenciales (14.ª ed.). Pearson.

Cortez, L., & Morales, J. (2021). Transformación digital en el sector gastronómico: Un enfoque desde la experiencia del cliente. Editorial Gastronómica.

Fundación Telefónica. (2021). La comunicación interna como clave del éxito empresarial. https://educacion.fundaciontelefonica.com/la-comunicacion-interna/

Power Designer. (2024). Database modeling and code analysis. Powerdesigner.biz. https://www.powerdesigner.biz/ES/

Lucidchart. (2025). Qué es el lenguaje unificado de modelado (UML). Lucidchart. https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

Asana. (2025). What is Agile methodology? A beginner's guide to agile project management. Asana. https://asana.com/resources/agile-methodology

OpenAI. (2025). ChatGPT [Modelo de lenguaje grande]. https://www.openai.com/

Actian. (2025). ¿Qué es SQL Server? Actian - a Division of HCLSoftware. https://www.actian.com/es/what-is-sql-server/

PlantUML. (2025). PlantUML de un vistazo. PlantUML.com. https://plantuml.com/es/

Team Angular. (2025). What is Angular? - Angular. Angular.dev: Angular. https://angular.dev/overview

ANEXOS

Script base de datos:

```sql
IF DB_ID('SantaGulaDB') IS NULL
BEGIN
    CREATE DATABASE SantaGulaDB;
END
GO

USE SantaGulaDB;
GO

IF OBJECT_ID('dbo.DetallePedido', 'U') IS NOT NULL DROP TABLE dbo.DetallePedido;
IF OBJECT_ID('dbo.Pedido', 'U') IS NOT NULL DROP TABLE dbo.Pedido;
IF OBJECT_ID('dbo.Producto', 'U') IS NOT NULL DROP TABLE dbo.Producto;
IF OBJECT_ID('dbo.Categoria', 'U') IS NOT NULL DROP TABLE dbo.Categoria;
IF OBJECT_ID('dbo.Usuario', 'U') IS NOT NULL DROP TABLE dbo.Usuario;
GO

CREATE TABLE dbo.Usuario (
    idUsuario       INT IDENTITY(1,1) NOT NULL,
    nombreUsuario   VARCHAR(50) NOT NULL,
    rol             VARCHAR(20) NOT NULL,
    contra          VARCHAR(255) NOT NULL,
    CONSTRAINT PK_Usuario PRIMARY KEY (idUsuario)
);
GO

CREATE TABLE dbo.Categoria (
    idCategoria         INT IDENTITY(1,1) NOT NULL,
    nombreCategoria     VARCHAR(50) NOT NULL,
    descripcion         VARCHAR(100) NULL,
    activo              BIT NOT NULL CONSTRAINT DF_Categoria_activo DEFAULT (1),
    CONSTRAINT PK_Categoria PRIMARY KEY (idCategoria)
);
GO

CREATE TABLE dbo.Producto (
    idProducto      INT IDENTITY(1,1) NOT NULL,
    idCategoria     INT NOT NULL,
    nombre          VARCHAR(80) NOT NULL,
    precio          DECIMAL(10,2) NOT NULL,
    activo          BIT NOT NULL CONSTRAINT DF_Producto_activo DEFAULT (1),
    CONSTRAINT PK_Producto PRIMARY KEY (idProducto),
    CONSTRAINT FK_Producto_Categoria FOREIGN KEY (idCategoria)
        REFERENCES dbo.Categoria(idCategoria)
);
GO

CREATE TABLE dbo.Pedido (
    idPedido            INT IDENTITY(1,1) NOT NULL,
    idUsuario           INT NOT NULL,
    nombreCliente       VARCHAR(80) NOT NULL,
    referenciaUbicacion VARCHAR(150) NULL,
    esUrgente           BIT NOT NULL CONSTRAINT DF_Pedido_esUrgente DEFAULT (0),
    estadoPago          VARCHAR(20) NOT NULL,
    estadoPedido        VARCHAR(20) NOT NULL,
    total               DECIMAL(10,2) NOT NULL CONSTRAINT DF_Pedido_total DEFAULT (0),
    fechaHoraRegistro   DATETIME2(0) NOT NULL CONSTRAINT DF_Pedido_fechaHoraRegistro DEFAULT (SYSDATETIME()),
    CONSTRAINT PK_Pedido PRIMARY KEY (idPedido),
    CONSTRAINT FK_Pedido_Usuario FOREIGN KEY (idUsuario)
        REFERENCES dbo.Usuario(idUsuario)
);
GO

CREATE TABLE dbo.DetallePedido (
    idDetalle       INT IDENTITY(1,1) NOT NULL,
    idProducto      INT NOT NULL,
    idPedido        INT NOT NULL,
    cantidad        INT NOT NULL,
    precioUnitario  DECIMAL(10,2) NOT NULL,
    subtotal        AS (CAST(cantidad AS DECIMAL(10,2)) * precioUnitario) PERSISTED,
    CONSTRAINT PK_DetallePedido PRIMARY KEY (idDetalle),
    CONSTRAINT FK_DetallePedido_Producto FOREIGN KEY (idProducto)
        REFERENCES dbo.Producto(idProducto),
    CONSTRAINT FK_DetallePedido_Pedido FOREIGN KEY (idPedido)
        REFERENCES dbo.Pedido(idPedido),
    CONSTRAINT CK_DetallePedido_cantidad CHECK (cantidad > 0),
    CONSTRAINT CK_DetallePedido_precioUnitario CHECK (precioUnitario >= 0)
);
GO

CREATE INDEX IX_Producto_idCategoria ON dbo.Producto(idCategoria);
CREATE INDEX IX_Pedido_idUsuario ON dbo.Pedido(idUsuario);
CREATE INDEX IX_DetallePedido_idPedido ON dbo.DetallePedido(idPedido);
CREATE INDEX IX_DetallePedido_idProducto ON dbo.DetallePedido(idProducto);
GO
```