PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

ESCUELA HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

SISTEMA DE MESERO DIGITAL ASISTIDO POR INTELIGENCIA ARTIFICIAL PARA LA GESTIÓN DE PEDIDOS EN UNA FRANQUICIA GASTRONÓMICA

TAMAYO ARMAS PAUL STEVE

TUTOR: MGS. ARCINIEGAS AGUIRRE STALIN MARCELO

IBARRA – ECUADOR

JULIO, 2026

Ibarra, 22 de Julio de 2026

CERTIFICACIÓN TUTOR

En mi calidad de Tutor del Trabajo de **INTEGRACION CURRICULAR** titulado:

SISTEMA DE MESERO DIGITAL ASISTIDO POR INTELIGENCIA ARTIFICIAL PARA LA GESTIÓN DE PEDIDOS EN UNA FRANQUICIA GASTRONÓMICA, presentado por el estudiante con cédula de ciudadanía N°1003863261, para obtener el Título de INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN.

Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores.

Adicionalmente, se adjunta el certificado de porcentaje de originalidad de TURNITIN.


Turnitin Informe de Originalidad

Procesado el: 23-jun-2026 15:34 -05
Identificador: 2975098915
Número de palabras: 29115
Entregado: 3

tamayo Por DIEGO ROBERTO SALGADO BAEZ

Índice de similitud
8%

Similitud según fuente
Fuentes de Internet 7%
Publicaciones: 1%
Trabajos del estudiante: 3%

Mgs. Stalin Arciniegas

Digitally signed by Mgs. Stalin Arciniegas
DN: cn=Mgs. Stalin Arciniegas gn=Mgs. Stalin Arciniegas c=EC Ecuador l=EC Ecuador o=PUCESI ou=Escuela de Ingeniería e=smarciniegas@pucesi.edu.ec
Reason: I am the author of this document
Location:
Date: 2026-07-22 14:43-05:00

(f):_____

Mgs. Arciniegas Aguirre Stalin Marcelo

C.C.: 1003496815

PÁGINA DE APROBACIÓN DEL TRIBUNAL

El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra:

Mgs. Stalin Arciniegas

Digitally signed by Mgs. Stalin Arciniegas
DN: cn=Mgs. Stalin Arciniegas gn=Mgs. Stalin Arciniegas c=EC Ecuador l=EC Ecuador o=PUCESI ou=Escuela de Ingeniería e=smarciniegas@pucesi.edu.ec
Reason: I am the author of this document
Location:
Date: 2026-07-22 14:44-05:00

(f): ……

Mgs. Arciniegas Aguirre Stalin Marcelo

C.C.: 1003496815

Santiago Quishpe Morales

Firmado digitalmente por Santiago Quishpe Morales
Fecha: 2026.07.22 18:03:41 -05'00'

(f):……

Msc. Quishpe Morales Santiago Damian

C.C.: 1002697223

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
SEGUNDO ELICEO PUSDA CHULDE

(f):……

Ms. Pusda Chulde Segundo Eliseo

C.C.: 0401567938

ACTA DE CESIÓN DE DERECHOS

Yo, *Tamayo Armas Paul Steve autor*, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: “Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia”.

Ibarra, 22 de julio de 2026

(f):_____

Tamayo Armas Paul Steve

C.C.:1003863261

AUTORÍA

Yo, Tamayo Armas Paul Steve, portador de la cédula de ciudadanía N° 1003863261, declaro que el presente trabajo de investigación es de total responsabilidad del autor, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales.

(f):……..

Tamayo Armas Paul Steve

C.C. 1003863261

DEDICATORIA

A mis padres, por ser el cimiento inquebrantable de mi vida y el faro que ha guiado cada uno de mis pasos. Gracias por su fuerza inagotable, por los innumerables sacrificios silenciosos que han hecho para verme llegar hasta este punto, y por ese apoyo incondicional que me ha sostenido en los momentos de flaqueza. A ustedes les debo la transmisión de los principios, los valores éticos y la moral que hoy rigen mi desarrollo, tanto en el ámbito personal como en el profesional. Este logro académico no es más que el reflejo de todo el amor, la educación y el esfuerzo que han invertido en mí a lo largo de toda mi vida. Este título es, sin lugar a duda, tan mío como suyo.

A mi novia, Sophie, quien se convirtió en un pilar fundamental e irremplazable durante los pasajes más arduos y estresantes de esta carrera. Gracias por ser mi refugio seguro y mi compañera incondicional, por tu infinita paciencia en las largas noches de programación y desvelo, y por no soltarme la mano en los momentos de mayor incertidumbre. Eres la persona que me inspira, día tras día, a desafiar mis propios límites, a no conformarme, a ser una mejor versión de mí mismo y a dar siempre el cien por ciento en cada meta que me propongo.

A mi hermana, por formar parte vital de mi núcleo familiar y de mi historia. Gracias por los momentos compartidos, por las alegrías en casa y por acompañarme con cariño en este largo trayecto hacia la culminación de mis estudios universitarios, recordándome siempre el valor de la familia.

Tamayo Armas Paul Steve

AGRADECIMIENTOS

A la Pontificia Universidad Católica del Ecuador Sede Ibarra, mi respetada alma mater, por haberme abierto sus puertas y proporcionarme un entorno inmejorable de excelencia académica, integridad humana y crecimiento continuo. A sus aulas, laboratorios y pasillos, de los cuales me llevo recuerdos invaluables y una sólida formación profesional que portaré con sumo orgullo a lo largo de mi trayectoria.

A mi profesora de Integración Curricular, la Dra. Laura Torres, y a mi tutor de tesis, el Mgs. Stalin Arciniegas. Expreso mi más sincera y profunda gratitud por su invaluable guía, su rigor investigativo y su inagotable paciencia durante el desarrollo de este trabajo. Sus directrices metodológicas, sus correcciones precisas y sus consejos oportunos fueron la brújula que orientó el diseño y la materialización de este proyecto tecnológico. Gracias por exigir siempre lo mejor de mí y por elevar la calidad de este trabajo a su máximo potencial.

De igual manera, extiendo un sentido agradecimiento a todos los profesores y catedráticos que, con verdadera vocación de enseñanza, dedicación y exigencia, han guiado mi camino hasta este añorado momento. Gracias por compartir su vasta experiencia, por inspirar mi pensamiento crítico y por dotarme de las herramientas técnicas y analíticas necesarias para enfrentar con solvencia los retos del mundo profesional.

A mis compañeros de generación, con quienes tuve el honor de compartir incontables horas de estudio, proyectos complejos y el peso de las épocas de exámenes. Gracias por construir un ambiente de excepcional camaradería y por haber sido siempre esa mano amiga e indispensable en los momentos de mayor presión. Juntos hemos demostrado que los grandes desafíos universitarios son mucho más llevaderos, e incluso gratificantes, cuando se enfrentan con un verdadero espíritu de equipo y solidaridad.

Finalmente, a la administración de Fritadas Doña Zita y a todas aquellas personas que, de manera directa o indirecta, me brindaron su tiempo, apertura y facilidades para que este sistema digital dejara de ser una propuesta académica y se convirtiera en una realidad operativa.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Este trabajo describe el diseño, implementación y evaluación técnica de un sistema de "Mesero Digital" multimodal asistido por Inteligencia Artificial, orientado a resolver los cuellos de botella operativos y la saturación en la toma manual de pedidos dentro de una franquicia gastronómica de alta concurrencia. La solución emplea una arquitectura distribuida fundamentada en la computación en el borde (*Edge AI*) para procesar comandos de voz localmente, garantizando la privacidad de los datos del cliente y mitigando la dependencia de servicios en la nube.

El ecosistema tecnológico integra un *frontend* interactivo (Kiosco, Monitor Kanban y Dashboard) construido con React, un servidor transaccional desarrollado en FastAPI, y almacenamiento relacional en PostgreSQL. El núcleo de procesamiento de lenguaje natural (PLN) acopla el modelo acústico Faster-Whisper para la transcripción de voz a texto y el modelo de lenguaje de gran escala Llama 3 (vía Ollama) para la extracción semántica y estructuración de comandas, aplicando validaciones estrictas con Pydantic. La orquestación transaccional y el enrutamiento asíncrono hacia la cocina se delegaron a la plataforma *low-code* n8n mediante el uso de *Webhooks*.

La evaluación de desempeño se ejecutó en un entorno operacional real bajo niveles de ruido ambiental de 65 a 75 dBA. Mediante interceptores de tiempo (time.perf_counter()), los resultados demostraron una latencia promedio de 1.24 segundos para el motor acústico, 2.32 segundos para la inferencia semántica del LLM, y 0.68 segundos en la orquestación relacional. El despliegue del sistema logró una reducción empírica del 38.5% en el Tiempo de Atención Promedio (TPA). Se registró una tasa de error transaccional del 4.0% generada por distorsiones fonéticas asociadas al ruido extremo de las máquinas de cocina. Un piloto cualitativo (n=15 comensales y n=3 operadores) basado en la Escala de Usabilidad del Sistema (SUS) arrojó un nivel de aceptación del 88.5% para la interacción multimodal y un 85.0% para la gestión de cocina.

Se concluye que la integración de *Edge AI* y orquestación *low-code* optimiza drásticamente el flujo operativo; sin embargo, la implementación de una interfaz híbrida (voz respaldada por control táctil) es indispensable para tolerar fallos acústicos. Para futuros trabajos se recomienda la adopción de hardware con micrófonos direccionales de cancelación de ruido, y la escalabilidad del orquestador hacia integraciones omnicanal (WhatsApp y aplicaciones de *delivery*).

Palabras clave: Inteligencia Artificial en el Borde (Edge AI), Procesamiento de Lenguaje

Natural, Llama 3, Arquitectura de Microservicios, n8n, Automatización de Restaurantes.

ABSTRACT

This work describes the design, implementation, and technical evaluation of a multimodal "Digital Waiter" system assisted by Artificial Intelligence, aimed at resolving operational bottlenecks and saturation in manual order-taking within a high-traffic gastronomic franchise. The solution employs a distributed architecture based on Edge computing (Edge AI) to process voice commands locally, ensuring customer data privacy and mitigating reliance on cloud services.

The technological ecosystem integrates an interactive frontend (Kiosk, Kanban Monitor, and Dashboard) built with React, a transactional server developed in FastAPI, and relational storage in PostgreSQL. The Natural Language Processing (NLP) core couples the Faster-Whisper acoustic model for speech-to-text transcription and the large language model Llama 3 (via Ollama) for semantic extraction and order structuring, applying strict validations with Pydantic. Transactional orchestration and asynchronous routing to the kitchen were delegated to the low-code platform n8n through the use of Webhooks.

The performance evaluation was executed in a real operational environment under ambient noise levels of 65 to 75 dBA. Using time interceptors (time.perf_counter()), the results demonstrated an average latency of 1.24 seconds for the acoustic engine, 2.32 seconds for the LLM semantic inference, and 0.68 seconds in relational orchestration. The system's deployment achieved an empirical reduction of 38.5% in the Average Service Time. A transactional error rate of 4.0% was recorded, generated by phonetic distortions associated with extreme noise from kitchen machinery. A qualitative pilot (n=15 diners and n=3 operators) based on the System Usability Scale (SUS) yielded an acceptance level of 88.5% for multimodal interaction and 85.0% for kitchen management.

It is concluded that the integration of Edge AI and low-code orchestration drastically optimizes the operational flow; however, the implementation of a hybrid interface (voice backed by tactile control) is indispensable to tolerate acoustic failures. For future work, the adoption of hardware with directional noise-canceling microphones and the scalability of the orchestrator towards omnichannel integrations (WhatsApp and delivery applications) are recommended.

Keywords: Edge AI, Natural Language Processing, Llama 3, Microservices Architecture, n8n, Restaurant Automation, Digital Waiter.

INTRODUCCIÓN

En la actualidad, las tecnologías basadas en la inteligencia artificial y los sistemas de interacción automatizada transforman aceleradamente los sectores tradicionales de servicios, destacándose su impacto en la industria gastronómica. Soluciones digitales innovadoras, como los meseros digitales, integran inteligencia artificial conversacional, interfaces multimodales que combinan interacción por voz y pantalla, y la automatización de flujos operativos mediante plataformas low-code como n8n. Estas tecnologías de vanguardia permiten optimizar la comunicación entre el cliente y el sistema, procesan el lenguaje natural para extraer intenciones o parámetros, y orquestan procesos internos complejos de manera autónoma, tales como el registro estructurado de pedidos, su envío directo a cocina y la actualización de estados en tiempo real.

A pesar de que la implementación de sistemas de atención automatizada y asistentes digitales basados en IA conversacional ha demostrado ser una alternativa eficaz para agilizar la interacción con los clientes y mejorar la eficiencia del servicio (Langarano Guerrero et al., 2022), muchas pequeñas y medianas empresas del sector gastronómico ecuatoriano aún no adoptan estas herramientas.

Este escenario problemático se evidencia en establecimientos tradicionales como Fritadas Doña Zita, ubicada en la ciudad de Ibarra, Ecuador. Actualmente, la mayor parte de sus procesos de atención al cliente y gestión de pedidos se ejecutan de forma manual. La carencia de un sistema automatizado que dinamice la captura de comandas, organice la cola de producción y brinde información precisa sobre los tiempos de espera representa una limitación crítica. Esta deficiencia tecnológica obstaculiza la modernización del servicio, generando un impacto negativo tanto en la eficiencia operativa del local como en la experiencia final del consumidor. Ante esta realidad, surge la necesidad de implementar una solución integral que combine interacción conversacional asistida por inteligencia artificial con la automatización de procesos internos.

Objetivo General

Desarrollar un Sistema de Mesero Digital Asistido por inteligencia artificial para la Gestión de Pedidos en una Franquicia Gastronómica Fritadas Doña Zita en la ciudad de Ibarra.

Objetivos específicos

1. Analizar los procesos operativos actuales de la franquicia Fritadas Doña Zita relacionados con la toma de pedidos, preparación en cocina y entrega, a fin de identificar cuellos de botella y oportunidades de mejora.
2. Diseñar la arquitectura técnica del sistema de mesero digital asistido por IA, especificando los componentes de interacción por voz, interfaz visual en pantalla, procesamiento de lenguaje natural, base de datos y flujos de automatización mediante n8n.
3. Implementar un prototipo funcional que permita registrar pedidos mediante voz, estructurar la información del pedido y visualizar el estado y tiempo estimado de preparación a través de una interfaz en pantalla.
4. Automatizar los flujos operativos del sistema para el envío de pedidos a cocina, la gestión de colas de atención y la actualización de inventario utilizando una plataforma de orquestación low-code.
5. Desarrollar un módulo de análisis de datos operativos que genere estadísticas sobre pedidos, tiempos de espera y comportamiento de consumo para apoyar la toma de decisiones administrativas.
6. Evaluar el desempeño del sistema mediante pruebas funcionales y de usabilidad en un entorno real, midiendo su impacto en tiempos de atención, errores en pedidos y percepción del usuario.

El presente documento se estructura en tres (3) capítulos: El primero aborda la consulta bibliográfica y conceptual que fundamenta la investigación. El segundo capítulo se centra en la descripción de las herramientas tecnológicas, la metodología del proyecto y el desarrollo del Sistema de Mesero Digital Asistido por IA. Finalmente, el tercer capítulo presenta los resultados obtenidos tras haber culminado el proyecto y realizar la evaluación de la implementación del sistema en la franquicia gastronómica Fritadas Doña Zita.

CAPÍTULO I. ESTADO DEL ARTE

El sector gastronómico experimenta una transformación digital acelerada impulsada por la Inteligencia Artificial (IA) y la automatización, convirtiendo herramientas como el reconocimiento de voz, las interfaces de autoservicio y las plataformas de orquestación en necesidades competitivas para optimizar la toma de pedidos, estandarizar el servicio y reducir la carga operativa del personal. En este contexto, la literatura científica actual subraya que la transparencia en la atención, particularmente a través de la visualización del estado del pedido y la estimación precisa de los tiempos de preparación, es un factor determinante para reducir la incertidumbre y elevar la satisfacción del cliente Roy et al. (2022). Por consiguiente, para fundamentar el desarrollo de un sistema integral de mesero digital, la presente revisión examina los avances y hallazgos previos estructurados en cuatro ejes fundamentales: la aplicación de la IA conversacional en el sector gastronómico, la interacción multimodal y los sistemas de autoservicio, la automatización operativa, y el análisis predictivo para la optimización de los tiempos de espera.

1.1. Antecedentes investigativos

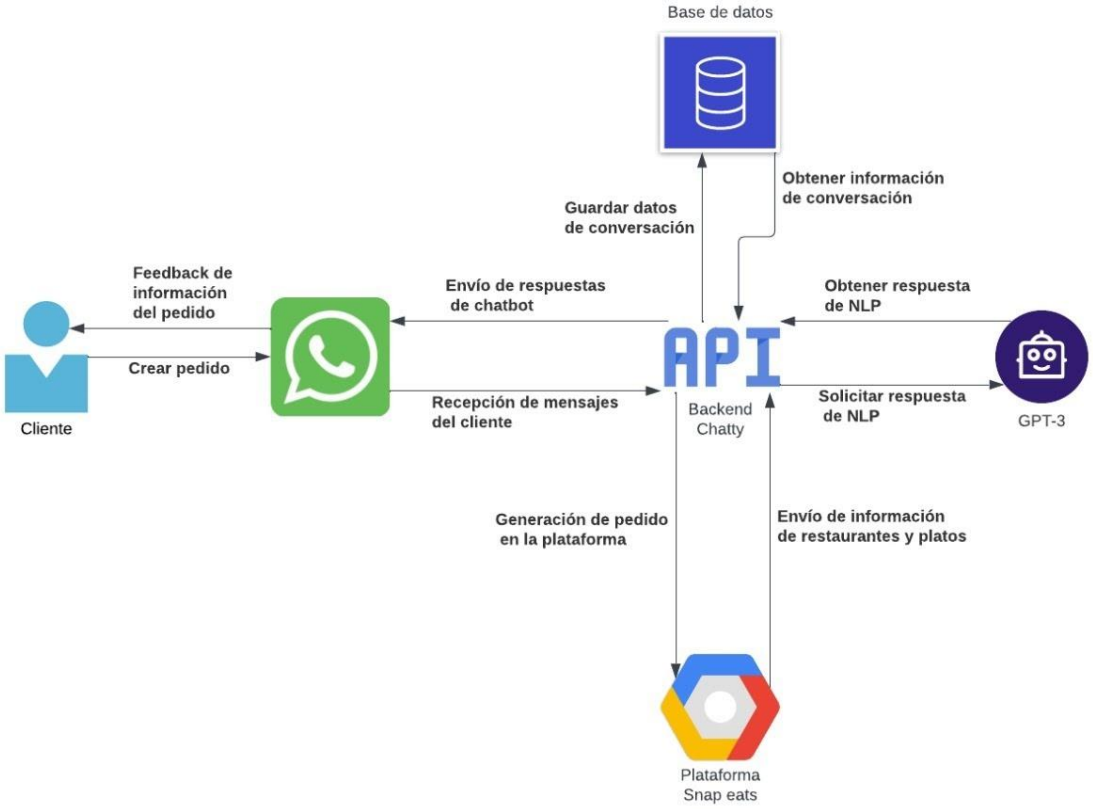
La integración de tecnologías disruptivas en el sector de servicios ha generado un cuerpo de conocimiento creciente que valida el uso de sistemas automatizados para mejorar la eficiencia y la experiencia del usuario. Los siguientes estudios constituyen la base científica sobre la cual se sustenta el desarrollo de un sistema de mesero digital asistido por inteligencia artificial.

1.1.1. Inteligencia artificial conversacional y agentes orientados a tareas

La transición hacia interfaces conversacionales ha permitido que los establecimientos gastronómicos gestionen pedidos de forma más estructurada. Langarano Guerrero et al. (2022) desarrollaron "Chatty", un agente conversacional basado en Procesamiento de Lenguaje Natural (NLP) motorizado por el modelo GPT-3 de OpenAI, diseñado para la gestión y recepción de pedidos en la plataforma de *delivery* Snap Eats a través de WhatsApp. Como se observa en su arquitectura técnica (véase Figura 1), los investigadores implementaron una arquitectura cliente-servidor enfocada en el *backend* bajo el entorno de Node.js, ejecutada en contenedores Docker. Este sistema utiliza una API propia para orquestar la comunicación en tiempo real entre la interfaz de mensajería, el modelo de inteligencia artificial y una base de datos no relacional (Firebase), permitiendo una interacción ágil y autónoma.

Figura 1

Arquitectura técnica del agente conversacional Chatty enfocado en la gestión de pedidos



Nota. El diagrama ilustra el flujo de comunicación entre el cliente, la interfaz de mensajería, el componente de lógica central (API) y el modelo de lenguaje para el procesamiento de la orden. Tomado de "Implementación de un chatbot con NLP para recibir pedidos en una plataforma de delivery", por M. Langarano, F. Montaluisa y M. Navas, 2022, *Revista Tecnológica - Espol, 34*(3), p. 157-170 (https://doi.org/10.37815/rte.v34n3.958)

Para el presente proyecto, este estudio trasciende la simple justificación del uso de asistentes virtuales; aporta una validación directa a la arquitectura tecnológica que se propone desarrollar. Específicamente, fundamenta la viabilidad operativa de utilizar Modelos de Lenguaje Grande (LLMs) como motor de razonamiento para la toma de pedidos, así como la necesidad imperativa de diseñar un ecosistema centralizado basado en el consumo de APIs que sincronice la intención del usuario con las bases de datos operativas del restaurante.

Sobre la percepción del consumidor, Guamán Tacuri et al. (2025) destacan que los chatbots cognitivos elevan significativamente la satisfacción al integrar tecnologías de aprendizaje profundo y procesamiento de lenguaje natural. Al simular la inteligencia humana, estos sistemas ofrecen respuestas dinámicas, adaptadas al contexto del usuario y con disponibilidad

ininterrumpida. Para el desarrollo del mesero digital, este estudio fundamenta la necesidad de utilizar modelos cognitivos que no solo capturen información, sino que comprendan la intención semántica real detrás de la comunicación del cliente.

Asimismo, los autores señalan que estos sistemas optimizan el servicio al reducir los tiempos de espera y automatizar tareas operativas como el seguimiento de pedidos. Un aporte crítico de este antecedente para el presente trabajo es la validación de la alta tolerancia a errores que poseen estos agentes. En el entorno de un restaurante tradicional, donde la comunicación suele ser informal o fragmentada, la capacidad de la IA para interpretar frases incompletas, ambigüedades o errores ortográficos garantiza que el pedido sea estructurado y enviado a cocina con absoluta exactitud.

Respecto a la gestión del ciclo de vida de estas soluciones, investigaciones recientes sugieren que el modelo de desarrollo incremental es el más adecuado para estabilizar sistemas que dependen de servicios externos de procesamiento de lenguaje natural. Autores como Amir y Atif (2026) demuestran que avanzar mediante ciclos funcionales permite validar la comunicación entre el *backend* y la lógica conversacional antes de la integración total, lo que asegura que cada incremento del software sea operativo y reduzca los riesgos técnicos en las pruebas de campo.

1.1.2. Interacción multimodal y sistemas de autoservicio

El diseño de interfaces que combinan múltiples canales de comunicación, como la voz y la pantalla, constituye una tendencia crítica para optimizar la accesibilidad y la eficiencia operativa. Patil et al. (2025) evaluaron el impacto del Sistema Robótico de Gestión de Restaurantes (RRMS), el cual integra asistentes de voz, interfaces de usuario (UI) y aprendizaje automático para automatizar el flujo completo desde la toma de pedidos hasta el despacho. Los resultados del estudio demuestran que esta automatización integral no solo mejora la satisfacción de los clientes y el personal por su facilidad de uso, sino que genera una reducción directa en los tiempos de espera y un aumento en las tasas de rotación de mesas.Para la presente investigación, este antecedente es fundamental ya que justifica la arquitectura multimodal propuesta, validando que el uso de voz para la rapidez y la pantalla para la confirmación visual es la estrategia más efectiva para dinamizar la atención en "Fritadas Doña Zita".

Asimismo, la investigación de Patil et al. (2025) profundiza en el uso de algoritmos de aprendizaje automático para sofisticar la experiencia del usuario y la gestión del negocio. Mediante la implementación de modelos de regresión para el pronóstico de la demanda y técnicas de filtrado colaborativo para sistemas de recomendación personalizados, el RRMS

logra adaptar el menú a las tendencias actuales y preferencias individuales. En el contexto de este proyecto, este aporte técnico sustenta la inclusión de módulos inteligentes que no solo procesen el pedido, sino que utilicen el historial de datos para optimizar el inventario y sugerir platos adicionales al cliente de manera estratégica. De esta forma, el enfoque híbrido propuesto actúa como un mecanismo de seguridad y optimización: mientras el componente multimodal mitiga los errores causados por el ruido ambiental, la lógica algorítmica garantiza un servicio eficiente y orientado a la rentabilidad del establecimiento.

1.1.3. Análisis predictivo y optimización de tiempos de espera

La gestión transparente de las expectativas del cliente y la optimización de los recursos internos constituyen factores críticos para la rentabilidad en el sector gastronómico moderno. Roy et al. (2022) argumentan que los restaurantes enfrentan hoy una competencia directa por los recursos compartidos de la cocina debido a la coexistencia de flujos de demanda simultáneos, como la atención presencial y el *delivery*. Si no se mide con precisión la carga de trabajo, la saturación operativa genera "costos por retraso" que impactan negativamente en los márgenes de ganancia. En el marco de esta tesis, este estudio proporciona el sustento teórico para el algoritmo de estimación del mesero digital, permitiendo que el sistema procese el historial de transacciones y la carga actual para informar al usuario de "Fritadas Doña Zita" sobre la demora estimada antes de confirmar su orden. Esta capacidad no solo mitiga la frustración del cliente —cuya fidelización es altamente sensible a los incumplimientos de tiempo—, sino que permite al sistema realizar ajustes dinámicos, como recomendar platos de preparación rápida durante picos de concurrencia para equilibrar la eficiencia y los ingresos.

Paralelamente, el uso de modelos de aprendizaje automático permite transformar estos datos históricos en herramientas proactivas para la gestión operativa y del talento humano. Gala (2023) validó que la implementación de algoritmos avanzados, específicamente el *Random Forest Regressor* (RFR), permite anticipar las necesidades de personal con una precisión del 94.5%. Al interpretar patrones complejos que incluyen métricas de productividad, turnos y tendencias de flujo de clientes, estos modelos otorgan a la administración la capacidad de tomar decisiones basadas en evidencia analítica. Para el presente proyecto, este hallazgo fundamenta la creación del módulo de análisis de datos administrativo, diseñado para identificar cuellos de botella antes de que ocurran. Según Gala (2023), una alineación precisa de recursos elimina los desequilibrios en la plantilla, evitando tanto el exceso de personal en momentos de baja demanda —que eleva los costos laborales— como la carencia de trabajadores durante horas pico. Este enfoque integral es indispensable para garantizar un despacho de comandas eficiente

y proteger la rentabilidad del restaurante, manteniendo los estándares de calidad del servicio frente a la alta variabilidad de la demanda característica de la comida tradicional.

1.1.4. Contexto local y aceptación tecnológica

En el ámbito ecuatoriano, investigaciones recientes validan la disposición del mercado local hacia la transformación digital en negocios tradicionales. Vivas-Naranjo y Cueva-Costales (2024) realizaron un estudio cuantitativo en la ciudad de Ibarra, específicamente en la "Cafetería La Estación 04", donde evaluaron la percepción de 87 usuarios mediante encuestas estructuradas. Los hallazgos revelaron una alta valoración de la inmediatez, con un 77% de los participantes calificando la rapidez en la gestión de pedidos como un factor fundamental para su experiencia de compra. Asimismo, la herramienta se posicionó como el canal de mayor preferencia para realizar pedidos (36.8%), superando ampliamente a las redes sociales, las llamadas telefónicas tradicionales y los sitios web. Para esta tesis, estos datos resultan determinantes, ya que proporcionan evidencia empírica de que el público objetivo en Ibarra posee la madurez tecnológica y la apertura necesarias para adoptar con éxito el sistema de mesero digital propuesto.

La solidez de esta aceptación se ve respaldada por el hecho de que el 74.7% de los clientes encuestados recomendaría el uso de servicios gestionados mediante sistemas inteligentes debido a su eficiencia operativa. Esta tendencia positiva permite fundamentar el diseño de una interfaz multimodal que potencie la agilidad del servicio y refuerce la confianza del usuario mediante la confirmación inmediata de sus transacciones.

Finalmente, el análisis del contexto investigativo y local evidencia que las soluciones actuales suelen depender exclusivamente de servicios en la nube (Cloud) o limitarse a aplicaciones de mensajería básica de texto. En este sentido, la presente investigación se diferencia técnicamente al proponer una arquitectura basada en *Edge AI*, desplegando los modelos de procesamiento de lenguaje natural y reconocimiento acústico de forma local para mitigar la latencia y la dependencia de red.

A nivel funcional, la propuesta trasciende el simple registro de órdenes al integrar una interfaz multimodal y un motor analítico basado en la teoría de colas. Esta sinergia permite gestionar dinámicamente las expectativas del cliente mediante la estimación de tiempos de espera reales, configurando una solución robusta, escalable y adaptada a la alta concurrencia de un restaurante tradicional andino como Fritadas Doña Zita.

1.2. Marco teórico

1.2.1. Contexto operativo en el sector gastronómico

Para comprender la necesidad de implementar soluciones tecnológicas en la hospitalidad, es imperativo definir la naturaleza de sus procesos operativos tradicionales y las limitaciones inherentes a los entornos de alta demanda.

El núcleo operativo de un restaurante tradicional se basa en el ciclo de la comanda (orden de pedido), el cual comprende la recepción de la solicitud por parte del personal, su transcripción manual o digital, la transmisión a la cocina, la preparación y el despacho final. En modelos no automatizados, este flujo depende directamente de la intervención humana, por lo que la latencia entre la toma del pedido y su ingreso a la línea de producción está condicionada por la disponibilidad del mesero.

Esta dependencia introduce limitaciones operativas que se hacen evidentes en escenarios de alta demanda. En horas pico, el incremento de pedidos genera cuellos de botella, entendidos como puntos de congestión donde la demanda supera la capacidad de procesamiento. Estos suelen concentrarse en la etapa de recepción de pedidos, afectando la continuidad del flujo de la comanda y derivando en tiempos de espera prolongados.

En respuesta a estas limitaciones, la digitalización gastronómica (FoodTech) propone la transición de procesos analógicos a ecosistemas digitales. Esto implica la adopción de tecnologías de autoservicio y sistemas de punto de venta (POS) conectados en la nube, los cuales permiten desacoplar la toma de pedidos de la intervención directa del personal y optimizar el flujo operativo.

1.2.2. Inteligencia artificial conversacional y modelos de lenguaje

El núcleo funcional del sistema propuesto radica en su capacidad para interpretar y procesar las solicitudes de los usuarios en lenguaje natural, simulando una interacción conversacional fluida. Este proceso no ocurre de forma aislada, sino como una cadena de transformación de datos que inicia con texto no estructurado y culmina en instrucciones operativas ejecutables. En este contexto, el Procesamiento de Lenguaje Natural (NLP) constituye la base que permite a los sistemas computacionales analizar lenguaje humano, mientras que la Comprensión del Lenguaje Natural (NLU) se encarga de interpretar su significado semántico y contextual. A partir de esta interpretación, el sistema transforma entradas no estructuradas en representaciones estructuradas que pueden ser procesadas internamente (Jurafsky & Martin, 2023).

Sobre esta base, los sistemas conversacionales tradicionales operan mediante la identificación de intenciones (intents) y la extracción de entidades (slot-filling). Este enfoque permite mapear una solicitud del usuario hacia una acción específica —por ejemplo, "ordenar comida"— y extraer parámetros relevantes como cantidad, tipo de plato o preferencias. Sin embargo, este modelo basado en reglas presenta limitaciones en escenarios abiertos o ambiguos, donde la variabilidad del lenguaje es mayor. Para superar estas restricciones, los Modelos de Lenguaje Grande (LLMs) introducen un enfoque basado en aprendizaje profundo, utilizando arquitecturas Transformer para modelar dependencias contextuales en secuencias de texto. A diferencia de los sistemas tradicionales, los LLMs no dependen exclusivamente de estructuras predefinidas, sino que generan respuestas en función del contexto, lo que permite interacciones más flexibles y naturales (OpenAI, 2023; Zhao et al., 2023).

No obstante, el comportamiento de estos modelos debe ser controlado para garantizar respuestas coherentes con el dominio del sistema. En este sentido, la ingeniería de instrucciones (Prompt Engineering) permite definir el rol del agente, sus acciones operativas y el formato de salida esperado, facilitando la integración con sistemas externos mediante estructuras como JSON. Este enfoque resulta clave para adaptar modelos generativos a entornos transaccionales como la toma de pedidos.

Finalmente, la integración de estos componentes da lugar a agentes conversacionales orientados a tareas, cuyo objetivo no es mantener una conversación abierta, sino ejecutar acciones específicas a partir de la interacción con el usuario. Estos agentes combinan capacidades de comprensión del lenguaje con acceso a servicios externos —como bases de datos, APIs o sistemas de orquestación como n8n— permitiendo transformar una solicitud en una acción concreta dentro del sistema operativo del restaurante.

En términos de implementación, actualmente existen diversas herramientas que permiten materializar estos componentes, como modelos de lenguaje (por ejemplo, APIs de OpenAI o modelos open-source como LLaMA), frameworks de orquestación de agentes (LangChain, LlamaIndex) y servicios de NLP integrados (Google Dialogflow, Rasa). Estas tecnologías facilitan la construcción de sistemas conversacionales escalables, integrables y orientados a tareas, alineados con los requerimientos del sistema propuesto.

1.2.3. Tecnologías de reconocimiento y procesamiento de voz

Para habilitar una interfaz conversacional basada en voz, el sistema debe transformar señales de audio del entorno en texto procesable por los módulos de inteligencia artificial, constituyendo la primera etapa del flujo de interacción.

En este sentido, los sistemas de Reconocimiento Automático de Habla (ASR o Speech-to-Text) permiten transcribir el habla humana mediante modelos acústicos y de lenguaje. Los enfoques actuales basados en aprendizaje profundo han mejorado su precisión, adaptándose a variaciones de acento, velocidad y vocabulario específico (Radford et al., 2023).

Sin embargo, en entornos ruidosos como restaurantes, la tasa de error de palabras (WER) puede incrementarse debido a interferencias sonoras. Para mitigar este efecto, se emplean técnicas como la supresión de ruido, cancelación de eco y el uso de micrófonos direccionales, mejorando la calidad de la señal de entrada (Du et al., 2023).

En cuanto al procesamiento, el enfoque streaming permite transcribir el audio en tiempo real, a diferencia del procesamiento batch que introduce mayor latencia. Esta característica resulta clave en el sistema propuesto, ya que permite mantener una interacción fluida durante la toma de pedidos.

Actualmente, tecnologías como Google Speech-to-Text, Azure Speech Services y modelos como Whisper facilitan la implementación de estas capacidades, permitiendo integrar reconocimiento de voz robusto en sistemas conversacionales.

1.2.4. Interacción humano-máquina (hci) y multimodalidad

El diseño de la interfaz en sistemas conversacionales determina directamente la forma en que el usuario interactúa con la inteligencia subyacente, influyendo tanto en la eficiencia del proceso como en la percepción de usabilidad del sistema. En este contexto, la interacción humano-máquina (HCI) en el sistema propuesto se fundamenta en un enfoque multimodal, entendido como la capacidad de un sistema para comunicarse mediante más de un canal sensorial de manera simultánea. La combinación de entrada por voz y retroalimentación visual en pantalla no solo permite una interacción más natural, sino que también incrementa la robustez del sistema frente a posibles errores de interpretación, ya que el usuario puede validar en tiempo real la información capturada por el módulo conversacional.

A partir de esta base, la experiencia de usuario (UX) en entornos de autoservicio como kioscos digitales se evalúa en función de métricas como la facilidad de aprendizaje, la eficiencia de uso y la tasa de error en la interacción. Esto implica que el diseño de la interfaz debe priorizar la simplicidad estructural, el uso de elementos visuales claros, botones de gran tamaño y una

jerarquía de información lineal que reduzca la complejidad cognitiva del usuario durante el proceso de pedido. En este sentido, la integración de múltiples canales no solo cumple una función de accesibilidad, sino que también contribuye directamente a la reducción de la carga cognitiva, entendida como el esfuerzo mental requerido para completar una tarea dentro del sistema. Mientras el canal de voz permite una interacción más rápida y natural, la interfaz visual actúa como un mecanismo de confirmación y corrección, asegurando coherencia entre la intención del usuario y la acción ejecutada por el sistema, lo que resulta especialmente relevante en entornos dinámicos como restaurantes de alta demanda.

1.2.5. Automatización y orquestación de procesos internos

La gestión del flujo de datos en el backend resulta vital para conectar de manera autónoma la interfaz del cliente con la lógica de negocio en sistemas de alto volumen como el mesero digital. La automatización de procesos empresariales (BPA) implica el empleo de software para ejecutar flujos de trabajo repetitivos, eliminando la intervención humana en la transferencia de datos entre sistemas aislados y optimizando la eficiencia operativa en entornos de servicio (Moreira et al., 2025). En el sistema propuesto, la BPA se aplica para procesar automáticamente los pedidos capturados por voz, validarlos contra el inventario y generar notificaciones internas sin demoras manuales.

Las plataformas de orquestación low-code, como n8n, facilitan la construcción de arquitecturas complejas de integración mediante interfaces gráficas basadas en nodos, conectando APIs, bases de datos y modelos de IA con mínima codificación tradicional (Sahay et al., 2020). Estas herramientas reducen el tiempo de desarrollo en más de 150 veces para flujos repetitivos, según evaluaciones empíricas en escenarios empresariales pequeños (Amir y Atif, 2026). En este proyecto, n8n orquesta el flujo desde la transcripción de voz hasta el envío de la orden a la cocina, asegurando resiliencia ante picos de concurrencia.

La arquitectura basada en eventos difiere de los modelos tradicionales de solicitudes periódicas al desencadenar la captura, comunicación y procesamiento de datos mediante eventos específicos, como la confirmación de un pedido, lo que garantiza respuestas inmediatas y escalabilidad en sistemas distribuidos (Chavan, 2021). Este paradigma logra menor latencia que los enfoques request-response, ideal para aplicaciones reactivas de tiempo real (Thomas, 2025). Aplicado al mesero digital, procesa el evento "pedido confirmado" para actualizar el dashboard de tiempos de espera basado en teoría de colas, notificando instantáneamente a la cocina.

Las APIs RESTful estandarizan la comunicación entre sistemas mediante métodos HTTP como POST y GET, mientras que los webhooks actúan como *callbacks* HTTP que transmiten datos en tiempo real al ocurrir un evento, formando el núcleo de las integraciones modernas **(Gupta, 2026)**. Esta combinación asegura una sincronización fluida y segura vía HTTPS en flujos de datos financieros u operativos. En Fritadas Doña Zita, aplicando esta lógica de automatización, los webhooks envían la orden procesada desde el LLM hacia n8n y de allí a la impresora de cocina, eliminando cuellos de botella humanos.

1.2.6. Modelado operativo y estimación de tiempos de espera

El modelado operativo de los sistemas de servicios se basa tradicionalmente en la teoría de colas, una disciplina matemática que analiza las líneas de espera en procesos donde las solicitudes llegan de forma aleatoria y los tiempos de procesamiento son variables. Bajo este enfoque convencional, los sistemas de atención se representan mediante modelos estocásticos lineales como M/M/1 o M/M/c, permitiendo calcular el factor de utilización (ρ), definido como la proporción entre la tasa de llegada (λ) y la tasa de servicio (μ). Un factor de utilización elevado indica una mayor congestión, lo cual resulta en un incremento de los tiempos de espera y una saturación de la capacidad instalada (Shortle et al., 2018).

Adicionalmente, el análisis del desempeño del sistema se sustenta en la Ley de Little ($L = \lambda W$), que establece una relación fundamental entre el número promedio de elementos en el sistema y el tiempo promedio que permanecen en él. Esta relación es esencial para estimar el tiempo de espera esperado en la cola (W_q), considerando la carga de trabajo y la capacidad de servicio. La aplicación de estos modelos permite una gestión cuantitativa de los flujos de atención, facilitando la toma de decisiones informadas para mejorar la eficiencia operativa y reducir la incertidumbre en los procesos de servicio (Shortle et al., 2018).

1.2.7. Arquitectura del sistema y analítica de datos

La consolidación técnica del proyecto exige la selección de herramientas de desarrollo y almacenamiento robustas.

Entornos de desarrollo frontend y backend: El Frontend (la capa de presentación visible para el cliente) se construye típicamente mediante frameworks reactivos. El Backend (lógica del servidor) gestiona la autenticación, las reglas de negocio y la comunicación segura con los modelos de IA y las pasarelas de orquestación.

Bases de datos transaccionales: Para garantizar la integridad de los pedidos, se requieren sistemas de gestión de bases de datos que cumplan con las propiedades ACID (Atomicidad,

Consistencia, Aislamiento y Durabilidad), asegurando que ninguna comanda se pierda o duplique debido a fallos de red.

Inteligencia de negocios (bi) y dashboards: La Inteligencia de Negocios agrupa metodologías para transformar datos brutos en información procesable. Mediante la extracción de Indicadores Clave de Rendimiento (KPIs)—como el plato más vendido o el tiempo promedio de despacho—y su visualización en dashboards, la administración puede tomar decisiones operativas basadas en evidencia empírica.

1.2.8. Computación en el borde (edge ai) en entornos interactivos

La Inteligencia Artificial en el Borde, o *Edge AI*, representa un cambio de paradigma frente a la computación tradicional basada en la nube (*Cloud Computing*). Consiste en integrar y desplegar modelos de aprendizaje automático directamente en los dispositivos locales o servidores perimetrales, lo más cerca posible de donde se generan los datos (Alotaibi et al., 2024). En el contexto de sistemas interactivos de atención al cliente, la dependencia exclusiva de servicios en la nube para el reconocimiento de voz y el procesamiento de lenguaje natural introduce vulnerabilidades críticas relacionadas con la latencia, caídas de conexión y costos recurrentes. Según la literatura reciente, la implementación de *Edge AI* permite que tareas computacionalmente demandantes se ejecuten en hardware local, garantizando tiempos de respuesta casi instantáneos, reduciendo el ancho de banda requerido y asegurando un manejo estrictamente confidencial de los datos del usuario (Kambala, 2024), siendo el estándar óptimo para soluciones de automatización en tiempo real.

1.2.9. Arquitectura orientada a servicios (soa) y microservicios

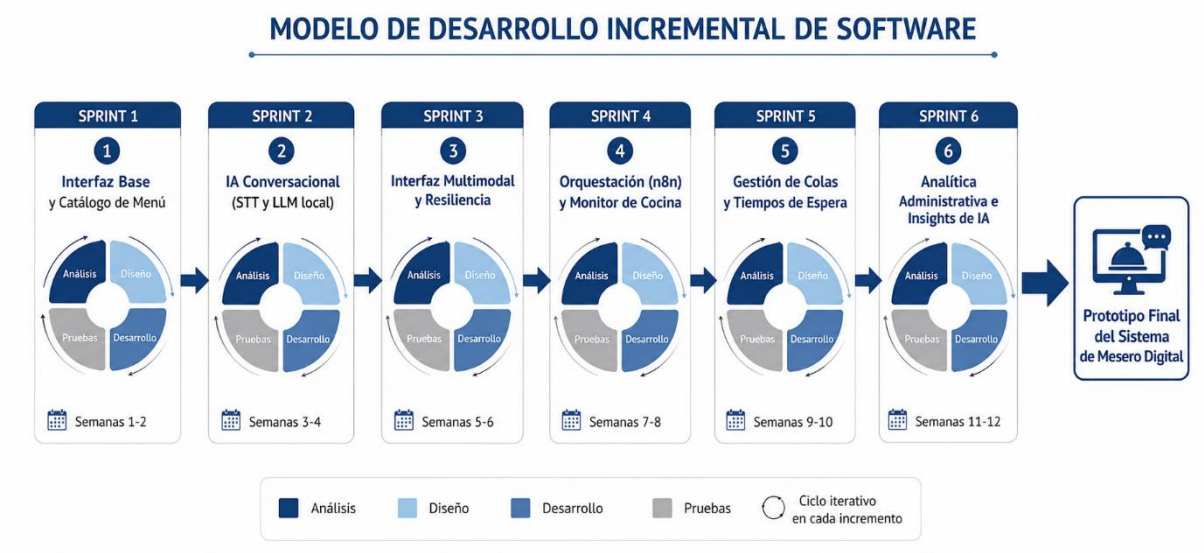
En el desarrollo de ecosistemas digitales modernos, la implementación de patrones distribuidos como la Arquitectura Orientada a Servicios (SOA) y su evolución natural hacia los microservicios, es imperativa para garantizar la resiliencia del software (Teo, 2025). En contraposición a las arquitecturas monolíticas tradicionales, donde la interfaz, la lógica de negocio y los datos están fuertemente acoplados, los modelos orientados a servicios promueven la modularidad y el aislamiento de fallos. Esta arquitectura permite desacoplar el entorno de presentación (Frontend interactivo) de los motores de procesamiento pesado de Inteligencia Artificial. Además, facilita la integración transparente de middleware u orquestadores de procesos (como n8n) que actúan como puentes de comunicación independientes hacia bases de datos y terminales secundarias. Esta segmentación asegura que el sistema sea altamente escalable y tolerante a fallos, facilitando la actualización de módulos de IA sin comprometer la disponibilidad de la plataforma completa (Teo, 2025).

1.2.10. Modelo de desarrollo incremental

El modelo incremental es una estrategia de ingeniería que combina elementos de los flujos de trabajo lineales y paralelos. En este enfoque, se aplican las fases clásicas del ciclo de vida del software —análisis, diseño, codificación y pruebas de forma repetitiva para cada componente funcional del sistema (Ibrahim et al., 2020). Tal como se muestra en la Figura 2, cada incremento incorpora estas etapas de manera secuencial, dando lugar a entregas parciales funcionales que contribuyen progresivamente a la construcción del sistema completo. Este enfoque resulta fundamental en proyectos que integran tecnologías de código abierto como Node.js y ExpressJS, ya que permite que el sistema crezca de manera controlada y escalable sin comprometer la integridad de la base de datos central.

Figura 2

Esquema del modelo de desarrollo incremental.



1.3.Matriz Comparativa

Tabla 1

Matriz Comparativa de Soluciones Tecnológicas en Gestión de Pedidos Gastronómicos

Proyecto / Autor	**Enfoque Principal**	**Tipo de Interfaz**	**Procesamiento de IA**	**Gestión Estocástica de Tiempos (Teoría de Colas)**
Chatty (Langarano et al., 2022)	Gestión vía WhatsApp	Texto (Mensajería)	Cloud (API GPT-3)	No
RRMS (Patil et al., 2025)	Sistema robótico integral	Multimodal (Voz/UI)	Cloud / Local híbrido	No (Pronóstico de demanda general)
Mesero Digital (Propuesta Actual)	**Kiosco interactivo de alta demanda**	**Multimodal (Voz/UI Táctil)**	**Edge AI (Local: Faster-Whisper / Ollama)**	**Sí (Cálculo estocástico en tiempo real)**

CAPÍTULO II. MATERIALES Y MÉTODOS

En este capítulo se describe la ejecución práctica y metodológica aplicada para la construcción del Sistema de Mesero Digital Asistido por IA. Para lograr la transición del proceso de toma de pedidos manual tradicional a un ecosistema automatizado en "Fritadas Doña Zita", se aplicó el modelo de Desarrollo de Software Incremental. Desde un enfoque estrictamente técnico, se detalla cómo se ejecutaron las fases de diseño, codificación e integración para construir una solución capaz de mitigar la saturación operativa y minimizar los errores de comunicación.

Con el propósito de evidenciar el trabajo de desarrollo realizado para alcanzar los resultados esperados, en las siguientes secciones se exponen las configuraciones de los entornos, las sentencias de base de datos y los fragmentos de código fuente que dan vida a cada componente de la arquitectura híbrida. Se documenta la programación de la interfaz interactiva en React, la lógica de validación de límites comerciales, la configuración del servidor backend en FastAPI, y la inyección de los modelos de inteligencia artificial (Faster-Whisper y Ollama) ejecutados mediante Edge AI. Finalmente, se detalla la configuración de los flujos de orquestación en la plataforma n8n, demostrando matemáticamente y a nivel de código cómo se calcula y gestiona la estimación de los tiempos de espera.

2.1. Generalidades de la investigación

La presente investigación se enmarcó dentro de un enfoque aplicado y de desarrollo tecnológico. El proyecto respondió a la necesidad de modernizar la gestión operativa de la franquicia "Fritadas Doña Zita", mediante la creación de un sistema que permitiera automatizar la toma de pedidos y reducir la saturación del personal en periodos de alta concurrencia. Se empleó un enfoque mixto, combinando métodos cuantitativos para evaluar el desempeño técnico del modelo de procesamiento de lenguaje natural y la precisión del cálculo de tiempos de espera, junto con métodos cualitativos para analizar la experiencia de usuario y la usabilidad de la interfaz multimodal. El alcance fue proyectivo, ya que se propuso el diseño, desarrollo e implementación de un prototipo funcional capaz de operar en contextos reales y optimizar la comunicación entre el cliente y la zona de producción.

2.1.1. Técnicas para la recolección de datos

Para la identificación de los requisitos funcionales y operativos del sistema, se emplearon técnicas como la revisión documental de los manuales de procesos de la franquicia, la observación directa en el local de la ciudad de Ibarra y entrevistas semiestructuradas dirigidas al personal administrativo. Toda la información obtenida permitió comprender las limitaciones

del sistema manual actual, caracterizado por errores de transcripción y cuellos de botella en la recepción de comandas. Asimismo, se realizó una revisión de literatura científica para validar el uso de algoritmos de estimación basados en la teoría de colas y servicios de inteligencia artificial conversacional.

2.1.1.1. Observación directa.

La observación directa se utilizó para analizar el flujo de trabajo tradicional y la interacción de los clientes al momento de realizar sus pedidos de forma verbal. Este proceso permitió identificar las barreras de comunicación habituales y la necesidad de una interfaz limpia que priorizara la simplicidad de navegación y la confirmación visual de la orden capturada por voz . Gracias a esta técnica, se determinó que la agilidad en la respuesta visual era crítica para reducir la incertidumbre del cliente sobre el estado de su preparación.

La Tabla 2 presenta una síntesis de la ficha de observación generada durante el diagnóstico situacional en el establecimiento.

Tabla 2

Ficha de observación de procesos actuales

FICHA DE OBSERVACIÓN	
Fecha: 15 de marzo del 2026	**Nº:** 1
Lugar: Fritadas Doña Zita - Ibarra	**Descripción:** Flujo de toma de pedidos manual
Observaciones:	Durante el diagnóstico, se evidenció que el personal dedica un tiempo promedio elevado a la transcripción de comandas, lo que genera retrasos en la transmisión de datos a cocina durante horas pico. La comunicación verbal suele verse afectada por el ruido ambiente del local, resultando en errores ocasionales en las especificaciones de los platos. Se confirmó la necesidad de un sistema que centralice la captura de datos y visualice el tiempo de espera esperado.

2.1.1.2. Entrevista semiestructurada.

La entrevista semiestructurada fue aplicada al administrador general de la franquicia "Fritadas Doña Zita", con el fin de recolectar información sobre los requisitos operativos y las métricas de rendimiento esperadas para el sistema. Este diálogo permitió definir el alcance del tablero de analítica administrativa y las necesidades de integración con los flujos de trabajo internos de la cocina. El guion detallado de esta intervención se presenta en el Anexo II.

2.1.2. Propósito

El objetivo primordial de esta investigación consistió en modernizar la gestión operativa de la franquicia mediante el desacoplamiento de la toma de pedidos de la intervención humana directa. Se buscó mitigar la saturación del personal durante las horas de mayor afluencia, evitando que los errores de comunicación o la carga de trabajo afectaran la eficiencia del servicio. Al automatizar el flujo de información, se persiguió estandarizar la captura de comandas y garantizar que cada pedido llegara a la zona de producción de forma inmediata y estructurada. Además, el proyecto se orientó a dotar a la administración de una herramienta de inteligencia de negocios que permitiera tomar decisiones basadas en evidencia empírica. Mediante la aplicación de la teoría de colas, se procuró eliminar la opacidad operativa, ofreciendo al cliente una estimación real de sus tiempos de espera basada en la carga de trabajo de la cocina. En última instancia, el fin último radicó en elevar la satisfacción del comensal a través de una experiencia tecnológica que garantizara rapidez y transparencia en todo el proceso de compra.

2.1.3. Alcance

El alcance de la investigación se delimitó desde la etapa de diagnóstico situacional en la sucursal de Ibarra hasta la evaluación final de un prototipo funcional en un entorno real. El trabajo comprendió el diseño de una arquitectura multimodal que integró la captura de voz para agilizar el pedido y una pantalla interactiva para la confirmación visual de los datos capturados. Técnicamente, el proyecto abarcó el desarrollo del backend en Python utilizando el framework FastAPI, lo que permitió una integración nativa y de baja latencia con los modelos de inteligencia artificial ejecutados en local. La creación de la interfaz de usuario se desarrolló mediante React, garantizando una experiencia visual reactiva y fluida para el entorno de kiosco. Para la lógica conversacional, se prescindió de APIs web limitadas en favor de modelos de reconocimiento de voz (STT) y procesamiento de lenguaje natural (LLMs) desplegados

localmente, capaces de extraer intenciones y parámetros del menú tradicional con alta precisión, incluso en entornos ruidosos.

Asimismo, el sistema incluyó la orquestación de procesos internos mediante la plataforma low-code n8n para el despacho automático de órdenes a cocina y la actualización en tiempo real de las líneas de espera a través de WebSockets. El alcance también consideró el diseño de un dashboard administrativo para el monitoreo de indicadores clave, tales como los tiempos promedio de atención y las estadísticas de consumo por plato. Finalmente, la investigación culminó con la ejecución de pruebas de usabilidad y rendimiento métrico en el local físico, verificando que la herramienta fuera capaz de operar bajo condiciones de ruido real y cumpliera con las expectativas de rapidez valoradas por el mercado local.

2.2. Metodología de desarrollo de la propuesta tecnológica

Para la construcción del sistema, se implementó el modelo de Desarrollo de Software Incremental, el cual se definió como un enfoque que permitió la entrega de funciones operativas en etapas sucesivas. Se eligió esta metodología porque facilitó la gestión de la complejidad técnica al dividir el proyecto en módulos manejables, asegurando que cada incremento añadiera valor real al sistema base sin comprometer la estabilidad de las partes ya desarrolladas (Ibrahim et al., 2020). Este proceso fue fundamental para abordar de manera sistemática los problemas operativos identificados en "Fritadas Doña Zita", tales como la saturación en la toma de pedidos y la falta de información sobre los tiempos de espera.

El desarrollo se llevó a cabo mediante ciclos repetitivos, en los cuales cada incremento integró las fases fundamentales del ciclo de vida del software. En una primera etapa, se realizó un levantamiento general de los requerimientos del sistema, lo que permitió definir una visión global y establecer la arquitectura base. Posteriormente, en cada incremento se efectuó un análisis más detallado de los requerimientos específicos del módulo en desarrollo, posibilitando el refinamiento progresivo y adaptativo de las funcionalidades.

A partir de este análisis, se procedió al diseño de la arquitectura técnica correspondiente a cada incremento, definiendo las estructuras de datos y los flujos de interacción necesarios para la automatización. En la fase de codificación, estos diseños se implementaron como componentes lógicos, integrando servicios de procesamiento de lenguaje natural y herramientas de orquestación de datos. Finalmente, cada incremento fue sometido a pruebas rigurosas con el fin de validar la correcta transmisión y estructuración de la información, garantizando la reducción de errores de transcripción antes de avanzar al siguiente ciclo de desarrollo.

Este avance gradual permitió que el proyecto evolucionara de manera controlada, facilitando la detección temprana de posibles fallos en la lógica de negocio o en la interpretación de las comandas. Al trabajar de forma incremental, se aseguró que la infraestructura tecnológica fuera escalable y que la integración de la teoría de colas para el cálculo de tiempos de espera se basara en un núcleo de software previamente validado y estable. De esta manera, el modelo incremental no solo guio la construcción técnica, sino que sirvió como un mecanismo de aseguramiento de la calidad durante todo el proceso de desarrollo de la propuesta.

2.2.1 Participantes y roles del proyecto

Para garantizar el éxito en la ejecución del modelo de desarrollo incremental, se definieron roles claros que establecen la gobernanza del proyecto, la validación de los requisitos y la ejecución técnica. Esta estructura organizativa asegura que el sistema tecnológico se alinee perfectamente con las necesidades operativas de la franquicia. A continuación, en la Tabla 3, se detallan las responsabilidades de cada participante:

Tabla 3

Roles y responsabilidades en el desarrollo del sistema

Rol en el Proyecto	Participante / Representación	Responsabilidades principales
Cliente / Beneficiario *(Stakeholder)*	Administrador general de "Fritadas Doña Zita"	Proveer la información sobre los procesos actuales, validar los requisitos funcionales del menú, facilitar el entorno para pruebas en sitio y aprobar los incrementos operativos.
Supervisor Metodológico *(Tutor)*	Arciniegas Aguirre Stalin Marcelo (Tutor)	Guiar la estructura académica del proyecto, supervisar el cumplimiento del modelo de desarrollo incremental y validar el rigor de la arquitectura técnica propuesta.
Ingeniero de Desarrollo	Tamayo Armas Paul Steve (Autor)	Diseñar la arquitectura multimodal, codificar el servidor local y la interfaz web, integrar los modelos de IA (STT y LLM), configurar la orquestación en *n8n* y documentar las pruebas.

2.3 FASE I: Diagnóstico, planificación y levantamiento de requisitos

2.3.1 Product backlog: planificación de historias de usuario

Una vez definida la ruta metodológica, se procedió a la recolección de requerimientos mediante historias de usuario, las cuales permitieron descomponer las necesidades operativas de la franquicia en unidades de desarrollo manejables. En la Tabla 4 se detallan las historias de usuario que sirvieron como base para la construcción del sistema, priorizando las funcionalidades según su impacto en el servicio al cliente.

Tabla 4

Historias de usuario para el sistema de mesero digital asistido por IA

ID	Nombre	Est. Esfuerzo (Hrs)	Importancia	Descripción de la historia de usuario	Criterios de Aceptación	Dependencias
HU-01	**Inicialización del Kiosco**	**10**	**1000**	En calidad de administrador, requiero inicializar el sistema en "Modo Kiosco" con el fin de restringir el acceso a otras aplicaciones y garantizar la interacción exclusiva con el menú.	- El navegador debe forzar la pantalla completa. - Botones de cerrar/minimizar ocultos. - No se debe poder salir al escritorio.	**Ninguna**
HU-02	**Visualización del Menú**	**25**	**980**	Desde la perspectiva del comensal, es necesario visualizar el menú estructurado por categorías temáticas, permitiendo ubicar con agilidad los productos a ordenar.	- Categorías claras y filtrables. - Mostrar nombre, precio e imagen referencial. - Tiempo base de preparación visible.	**HU-01**
HU-03	**Captura de pedidos por voz (STT)**	**35**	**1000**	Siendo usuario del sistema (comensal), necesito registrar mi pedido mediante dictado por voz tras accionar un botón, con el propósito de agilizar la orden minimizando la navegación táctil.	- El sistema activa el micrófono al interactuar. - Audio procesado localmente con Faster-Whisper.	**HU-02**

					- Transcripción en tiempo real en pantalla.	
HU-04	**Interpretación Semántica (LLM)**	**40**	**1000**	Como componente lógico, requiero procesar el texto transcrito mediante un modelo de lenguaje local (LLM) para extraer intenciones, estructurando la salida en formato JSON.	- Identificar platos del menú de Doña Zita. - Ignorar pedidos fuera del menú. - Generar JSON con ítems y personalizaciones.	**HU-03**
HU-05	**Confirmación Visual y Edición**	**20**	**950**	En mi rol de comensal, preciso visualizar un resumen de la orden procesada por la IA, de modo que pueda confirmar o modificar platillos a través de la interfaz táctil previo a su envío.	- Presentar ítems, detalles y costo total. - Permitir modificar cantidades (+/-). - Jerarquía visual simple.	**HU-04**
HU-06	**Estimación de Tiempo de Espera**	**30**	**900**	Desde el rol de comensal, es indispensable conocer el tiempo estimado de preparación antes de la confirmación final, con el objetivo de reducir la incertidumbre sobre el servicio.	- Cálculo aplicando teoría de colas basada en órdenes pendientes. - Actualización dinámica al modificar el carrito.	**HU-05**
HU-07	**Envío y Orquestación (n8n)**	**25**	**980**	El sistema requiere emplear n8n para la orquestación de la comanda, garantizando la persistencia en PostgreSQL y su transmisión a la zona de producción.	- n8n debe recibir el JSON validado. - Insertar registro en la tabla de pedidos.	**HU-05, HU-06**

					- Emitir evento hacia el monitor de cocina.	
HU-08	**Monitor de Recepción en Cocina**	**30**	**950**	En calidad de personal de cocina, requiero visualizar las comandas entrantes en tiempo real dentro de un monitor dedicado, a fin de iniciar la preparación sin demoras.	- Mostrar órdenes por orden de llegada (FIFO). - Resaltar especificaciones ("sin tostado"). - Botón para marcar orden como "Completada".	**HU-07**
HU-09	**Actualización de Estado**	**20**	**850**	Siendo comensal, necesito identificar visualmente en una pantalla de estados cuando mi orden se encuentre "Lista", permitiendo mi acercamiento oportuno para el retiro.	- Actualización instantánea vía WebSockets al completarse en cocina. - Alerta visual/sonora en pantalla principal.	**HU-08**
HU-10	**Generación de Reportes**	**25**	**750**	En el rol de administrador, es necesario generar reportes históricos segmentados por fechas, facilitando el análisis del volumen transaccional y tiempos de despacho.	- Gráficos de rotación de platos. - Exportación a formato digital (CSV/PDF). - Filtros por fecha.	**HU-07, HU-08**
HU-11	**Recomendaciones IA (Administrativas)**	**35**	**800**	Como gestor del local, requiero que el modelo inteligente analice los cortes de ventas semanales con el fin de obtener recomendaciones sobre el manejo de inventario.	- LLM debe procesar el resumen de ventas. - Generar sugerencias estratégicas claras.	**HU-10**

HU-12	**Manejo de Excepciones**	**15**	**850**	A nivel de sistema, es imperativo gestionar excepciones operativas (fallos de red, transcripción) para preservar la continuidad del kiosco interactivo.	- Si falla el micrófono, sugerir uso táctil. - Si el LLM falla, permitir ingreso manual. - Bloquear platos agotados.	**Todas**

1

2.3.2. Sprint backlog: descomposición y estimación técnica

Tras la estructuración formal de las historias de usuario, se ejecutó un análisis técnico exhaustivo para determinar la prioridad de implementación y el esfuerzo requerido por cada componente del ecosistema. Como resultado de este proceso de refinamiento del *backlog*, detallado en la Tabla 5, se contempla una estimación total de 310 horas de trabajo para la fase de ejecución. Este tiempo se encuentra estratégicamente distribuido en función de la complejidad arquitectónica de las tareas, abarcando el diseño de interfaces dinámicas (*Frontend*), la configuración de flujos de orquestación de datos (*Middleware*), y la codificación e integración de los modelos de Inteligencia Artificial en el servidor local (*Backend*).

Tabla 5

Sprint Backlog: Análisis y priorización de tareas técnicas

Prioridad	ID Historia de Usuario	ID Tarea	Descripción de la Tarea	Horas - Esfuerzo
1000	HU-01 (Inicialización)	T1-1	Configuración del enrutamiento base en React y layout principal.	4
		T1-2	Implementación de la API Fullscreen para inmersión total.	4
		T1-3	Desarrollo de bloqueos lógicos para navegación externa.	2
	HU-03 (Captura Voz)	T3-1	Configuración de FastAPI para recepción asíncrona de audio.	8
		T3-2	Integración y optimización del modelo Faster-Whisper local.	12
		T3-3	Desarrollo en React empleando MediaRecorder para micrófono.	8
		T3-4	Conexión cliente-servidor para transcripción en tiempo real.	7

	HU-04 (Interpretación IA)	T4-1	Despliegue del entorno local para el LLM utilizando Ollama.	8
		T4-2	Diseño de ingeniería de prompts para extracción de entidades.	10
		T4-3	Creación de endpoint en FastAPI para estructurar salida en JSON.	12
		T4-4	Implementación de validadores (Pydantic) para coherencia del JSON.	10
980	HU-02 (Visualización)	T2-1	Diseño del modelo relacional en PostgreSQL para el catálogo.	6
		T2-2	Desarrollo de API REST (GET) en FastAPI para exponer catálogo.	6
		T2-3	Maquetación de tarjetas y diseño responsivo en React.	8
		T2-4	Programación de lógica de filtrado dinámico por categorías.	5
	HU-07 (Orquestación)	T7-1	Despliegue del entorno de contenedores para n8n.	5
		T7-2	Creación de Webhook receptor en n8n para capturar JSON.	6
		T7-3	Configuración del nodo de integración entre n8n y PostgreSQL.	6

		T7-4	Flujo de emisión de alertas hacia la terminal de cocina.	8
950	HU-05 (Edición visual)	T5-1	Diseño del componente React para estado global del carrito.	6
		T5-2	Funciones aritméticas (suma, resta, eliminación) de ítems.	6
		T5-3	Mapeo visual del JSON devuelto por la Inteligencia Artificial.	5
		T5-4	Interfaz modal para la confirmación final de la transacción.	3
	HU-08 (Monitor Cocina)	T8-1	Maquetación del panel Kanban de recepción en React.	8
		T8-2	Conexión bidireccional WebSockets entre cliente y servidor.	8
		T8-3	Renderizado de tarjetas priorizadas por hora de llegada (FIFO).	8
		T8-4	Lógica de mutación de estados al marcar como "Completado".	6
900	HU-06 (Estimación Tiempos)	T6-1	Lógica de estimación de demora basada en teoría de colas (Ley de Little).	10
		T6-2	Consulta SQL para sumatoria de tiempos de preparación en cola.	8
		T6-3	Endpoint en FastAPI para el cálculo dinámico de demora.	8

		T6-4	Integración del indicador visual en el panel del comensal.	4
850	HU-09 (Actualización estado)	T9-1	Configuración del broadcast de eventos asíncronos en FastAPI.	8
		T9-2	Componente de notificaciones emergentes (Toasts) en frontend.	6
		T9-3	Alertas auditivas asociadas al cambio de estado del pedido.	6
	HU-12 (Manejo errores)	T12-1	Programación de fronteras de error (Error Boundaries) en React.	5
		T12-2	Implementación de timeouts y contingencia en peticiones HTTP.	5
		T12-3	Flujos alternativos (ej. caída de STT transiciona a teclado).	5
800	HU-11 (Recomendaciones IA)	T11-1	Script Python para extracción de métricas semanales.	10
		T11-2	Diseño de prompt analítico para inyectar datos estadísticos al LLM.	12
		T11-3	Vista administrativa para mostrar insights generados.	13
750	HU-10 (Reportes Admin)	T10-1	Consultas SQL agregadas para análisis histórico de comandas.	8
		T10-2	Integración de librerías de visualización (react-chartjs-2) en el panel.	10
		T10-3	Módulo de transformación y exportación de datos a CSV.	7

2.3.3 Planificación y distribución de sprints

Con las tareas técnicas estimadas en horas, se procedió a organizar el flujo de trabajo bajo el marco del desarrollo incremental. Se establecieron seis iteraciones operativas (Sprints), agrupadas lógicamente para asegurar que cada entrega aporte una funcionalidad validada y estable al ecosistema del "Mesero Digital".

Esta distribución garantiza que el núcleo de la interfaz y la inteligencia artificial se consoliden antes de avanzar hacia la orquestación y la analítica, mitigando riesgos de integración en etapas tardías. En la Tabla 6 se detalla la asignación específica de las tareas de desarrollo a cada uno de los Sprints planificados.

Tabla 6

Distribución de tareas de desarrollo por Sprints

Sprint	**Duración**	**Enfoque de la Iteración**	**Historias de Usuario**	**Tareas Asignadas**
Sprint 1	2 Semanas	Estructura Base e Interfaz de Menú	HU-01, HU-02	T1-1, T1-2, T1-3, T2-1, T2-2, T2-3, T2-4
Sprint 2	2 Semanas	Núcleo Conversacional de Inteligencia Artificial	HU-03, HU-04	T3-1, T3-2, T3-3, T3-4, T4-1, T4-2, T4-3, T4-4
Sprint 3	2 Semanas	Interacción Multimodal y Resiliencia	HU-05, HU-12	T5-1, T5-2, T5-3, T5-4, T12-1, T12-2, T12-3
Sprint 4	2 Semanas	Orquestación y Monitor de Producción	HU-07, HU-08	T7-1, T7-2, T7-3, T7-4, T8-1, T8-2, T8-3, T8-4
Sprint 5	2 Semanas	Gestión de Colas y Tiempos de Espera	HU-06, HU-09	T6-1, T6-2, T6-3, T6-4, T9-1, T9-2, T9-3
Sprint 6	2 Semanas	Analítica Administrativa e Insights	HU-10, HU-11	T10-1, T10-2, T10-3, T11-1, T11-2, T11-3
TOTAL	**12 Semanas**	**Ciclo completo de desarrollo e integración**	**12 HU**	**310 Horas estimadas**

2.4. FASE II: Diseño del Sistema

2.4.1. Casos de uso

Con el objetivo de representar de manera clara y estructurada las funcionalidades y el comportamiento del sistema propuesto, se elaboró el diagrama de casos de uso. Esta herramienta permite identificar las interacciones entre los actores principales y los procesos funcionales del "Mesero Digital".

En el ecosistema propuesto en la Figura 3 se identifican cuatro actores principales que interactúan de acuerdo con los permisos y responsabilidades asignadas:

Cliente: Representa al usuario final del kiosco. Este actor visualiza el menú, interactúa con el micrófono para dictar su pedido y confirma la orden en la pantalla táctil.

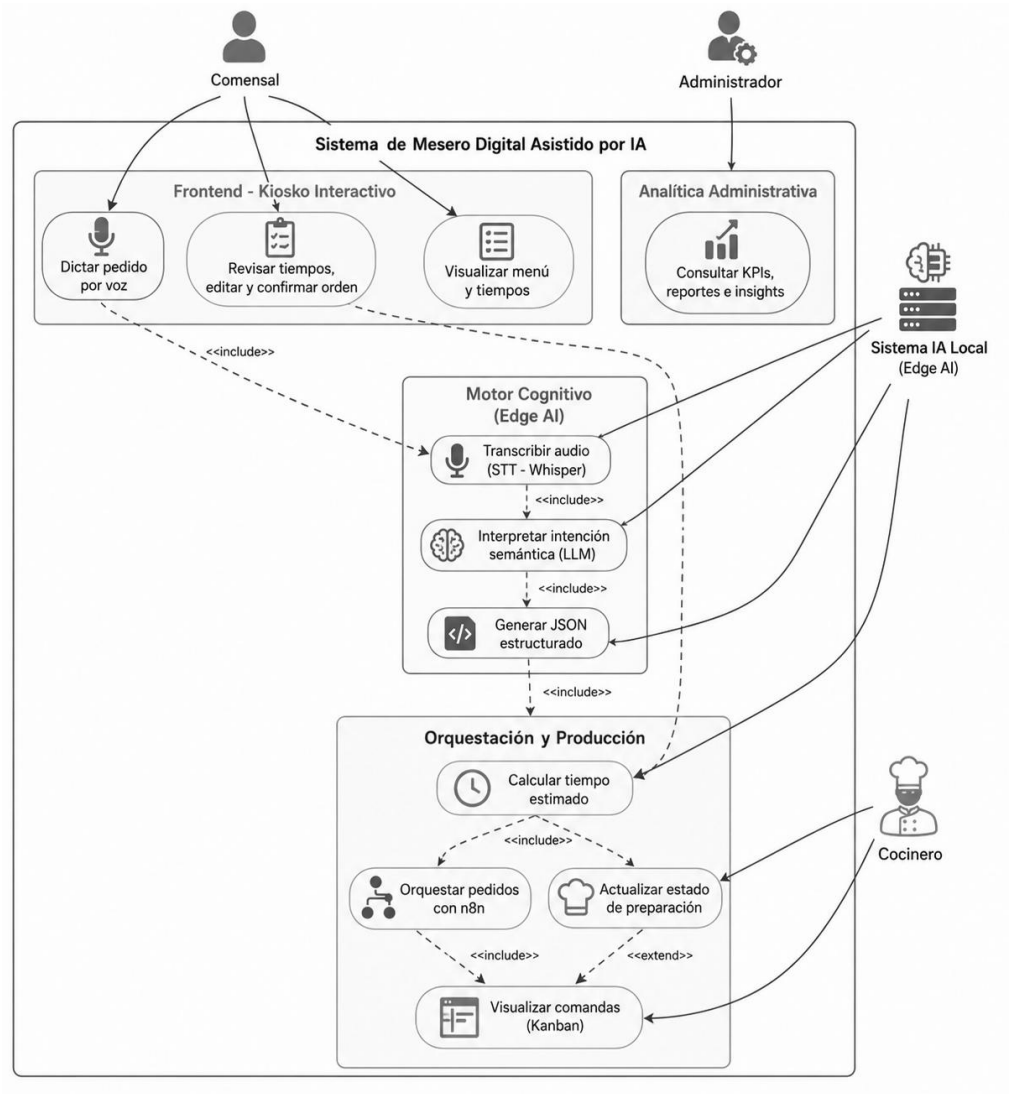
Sistema IA Local: Es un actor autónomo que opera en el servidor (FastAPI / n8n). Se encarga de transcribir el audio, interpretar el lenguaje natural para generar datos estructurados y calcular la estimación de demora.

Cocinero: Es el personal de producción que interactúa con el monitor de recepción de comandas y actualiza los estados a "Completado".

Administrador: Corresponde al rol responsable de la gestión general del negocio, quien consume los reportes analíticos generados por la plataforma.

Figura 3

Diagrama de Casos de Uso del Sistema de Mesero Digital



La Tabla 7 detalla los casos de uso críticos del "Sistema de Mesero Digital Asistido por IA", especificando actores, descripción, precondiciones, el flujo de eventos que involucra tanto a los usuarios físicos como a los componentes lógicos (Inteligencia Artificial y Orquestador), y las postcondiciones operativas.

Tabla 7

Descripción de Casos de Uso Principales

CASO DE USO	UC-01: Interacción con Menú Táctil
Actor	Comensal
Descripción	El comensal interactúa con el kiosco digital para explorar el catálogo de productos disponibles. El sistema presenta la información estructurada, permitiendo filtrar por categorías y visualizar detalles como precios, imágenes referenciales y tiempos base de preparación.
Precondiciones	• El kiosco debe estar encendido e inicializado en "Modo Kiosco" (Pantalla completa). • El catálogo de la base de datos PostgreSQL debe estar activo y poblado.
Flujo Normal	1. El comensal se acerca al kiosco y toca la pantalla inicial. 2. El sistema (FastAPI) consulta el catálogo activo en la base de datos. 3. El sistema renderiza el menú organizado por categorías (ej. Fritadas, Bebidas). 4. El comensal selecciona una categoría específica. 5. El sistema filtra y muestra únicamente los platillos de dicha categoría. 6. El comensal visualiza los detalles de los platos para tomar una decisión.
Postcondiciones	• El comensal conoce la oferta gastronómica y está listo para iniciar el proceso de orden.

CASO DE USO	UC-02: Toma de Pedido por Voz (IA)
Actor	Comensal, Sistema IA Local
Descripción	El comensal dicta su pedido utilizando lenguaje natural. El motor de Inteligencia Artificial procesa el audio localmente, extrae intenciones, cantidades y personalizaciones (extras/sin), y estructura un resumen exacto en la pantalla, reemplazando la navegación manual.
Precondiciones	• El comensal debe estar en la pantalla principal del menú.

	• Los servicios locales de Faster-Whisper y Ollama deben estar en ejecución.
Flujo Normal	1. El comensal presiona y mantiene el botón de captura de audio (STT). 2. El comensal dicta su pedido (ej. "Quiero una fritada especial sin cebolla"). 3. El frontend empaqueta el audio y lo envía al servidor FastAPI. 4. El Sistema IA (Faster-Whisper) transcribe el audio a texto plano. 5. El Sistema IA (Ollama) recibe el texto y el contexto del menú (Prompt Engineering). 6. El LLM extrae las entidades y devuelve un objeto JSON estructurado. 7. FastAPI valida el JSON utilizando esquemas Pydantic. 8. El sistema muestra el carrito actualizado con el plato y la modificación ("sin cebolla").
Postcondiciones	• El carrito de compras se encuentra poblado con los ítems correctos. • El sistema ha detectado y registrado las modificaciones o excepciones del plato.
CASO DE USO	UC-03: Orquestación y Envío de Comanda
Actor	Comensal, Sistema IA Local, n8n (Orquestador)
Descripción	El comensal confirma su pedido revisado. El sistema calcula la demora estimada y delega la transacción a n8n, el cual se encarga de registrar los datos financieros y notificar al área de producción sin bloquear la pantalla del usuario.
Precondiciones	• Debe existir al menos un ítem validado en el carrito de compras. • La plataforma n8n debe estar a la escucha de Webhooks.
Flujo Normal	1. El comensal verifica el subtotal y presiona "Confirmar Pedido". 2. El sistema calcula el tiempo estimado de preparación según la carga de la cocina. 3. FastAPI emite un evento (Webhook) con el Payload del pedido hacia n8n. 4. El kiosco muestra un mensaje de éxito y un número de turno al comensal. 5. n8n procesa el JSON e inserta los registros en PostgreSQL (Pedido, Detalle_Pedido, Modificacion_Item).

	6. n8n emite una alerta por WebSocket hacia la terminal secundaria correspondiente. 7. El monitor de cocina muestra la nueva tarjeta Kanban al personal.
Postcondiciones	• Pedido registrado financieramente en la base de datos. • Inventario comprometido según la receta base. • Personal de cocina notificado en tiempo real con especificaciones exactas.
CASO DE USO	UC-04: Actualización de Estado y Despacho
Actor	Cocinero, Comensal
Descripción	El personal de producción actualiza el estado de los platillos conforme avanza su preparación. Al finalizar, el sistema emite notificaciones asíncronas para que el comensal se acerque a retirar su pedido, cerrando el ciclo de servicio.
Precondiciones	• Debe existir un pedido en estado "SOLICITADO" o "PREPARANDO" en el monitor Kanban de cocina.
Flujo Normal	1. El cocinero visualiza la tarjeta del pedido en su pantalla. 2. El cocinero termina el ensamblaje o cocción del platillo. 3. El cocinero presiona el botón "Completado" en el monitor táctil. 4. El sistema actualiza el atributo estado_item en la base de datos a "ENTREGADO". 5. El sistema envía un evento de *broadcast* hacia el Frontend principal. 6. La pantalla de estados (o el kiosco) emite una alerta visual y sonora anunciando el número de turno. 7. El comensal identifica su turno y retira sus alimentos en barra.
Postcondiciones	• El flujo transaccional del platillo se cierra exitosamente. • El monitor de cocina se libera para atender la siguiente comanda en cola.

Nota. Casos de uso críticos que evidencian la transición del modelo de toma de pedidos tradicional hacia un ecosistema automatizado. Incluye la interacción multimodal (UC-02) y la orquestación asíncrona de datos para producción (UC-03).

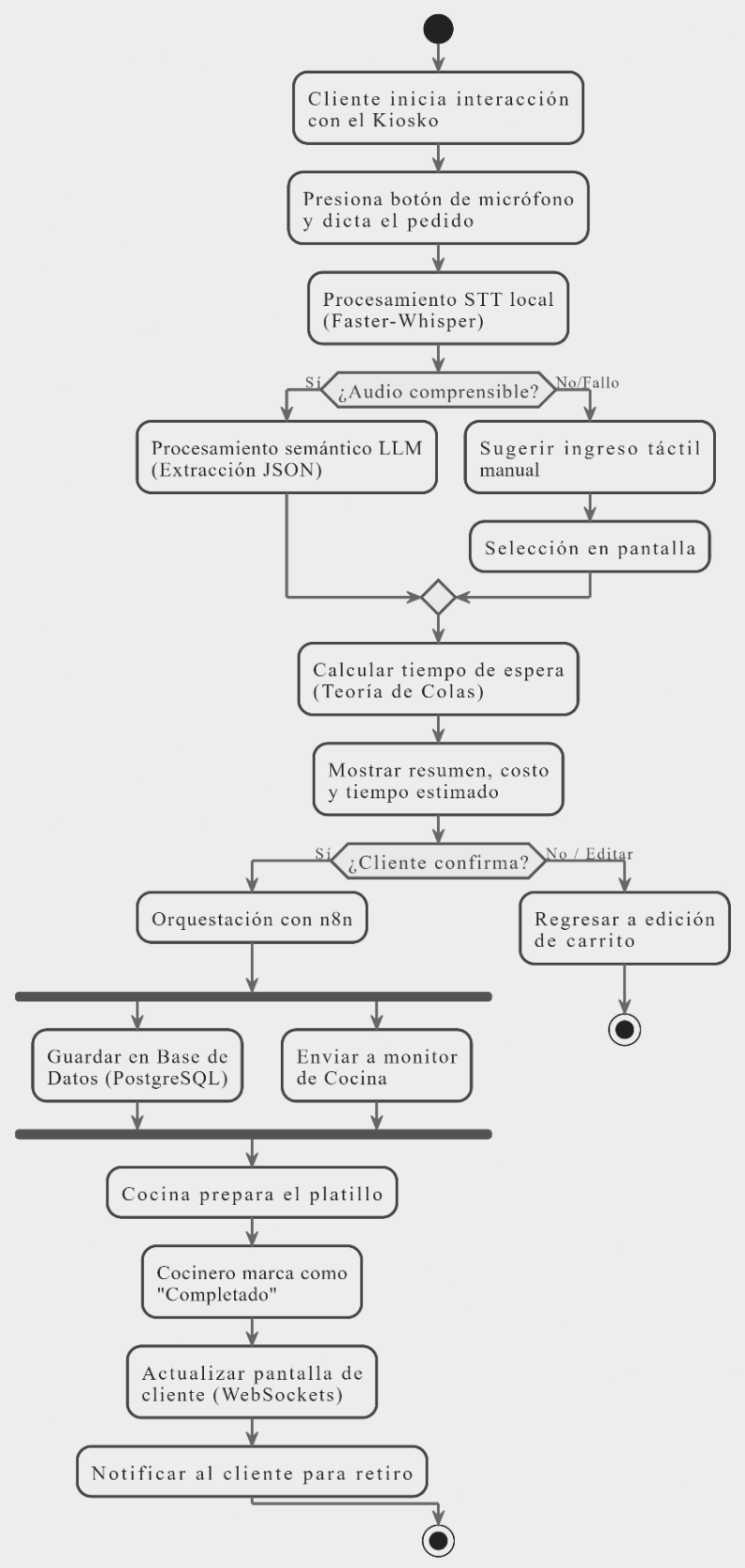
2.4.2. Diagrama de flujo del proceso

La Figura 4 ilustra el diagrama de flujo del proceso de gestión de pedidos del "Mesero Digital". Este esquema describe de manera secuencial las actividades y decisiones involucradas desde que el cliente interactúa con el kiosco hasta que la orden es completada y entregada.

El objetivo principal de este diagrama es representar la lógica operativa de la Inteligencia Artificial y la orquestación de datos en tiempo real. El proceso inicia con la captura de la intención del usuario mediante reconocimiento de voz (STT). Si el audio es ilegible, el sistema prevé una alternativa táctil. Posteriormente, el Modelo de Lenguaje (LLM) estructura la petición y el sistema calcula la demora estimada basada en la carga actual de la cocina. Finalmente, una vez confirmada la orden, la plataforma *n8n* orquesta la transmisión de la información hacia la base de datos y la terminal de producción, culminando con la actualización asíncrona del estado del pedido mediante WebSockets para notificar al cliente.

Figura 4

Diagrama de Flujo del Proceso de Gestión de Pedidos.

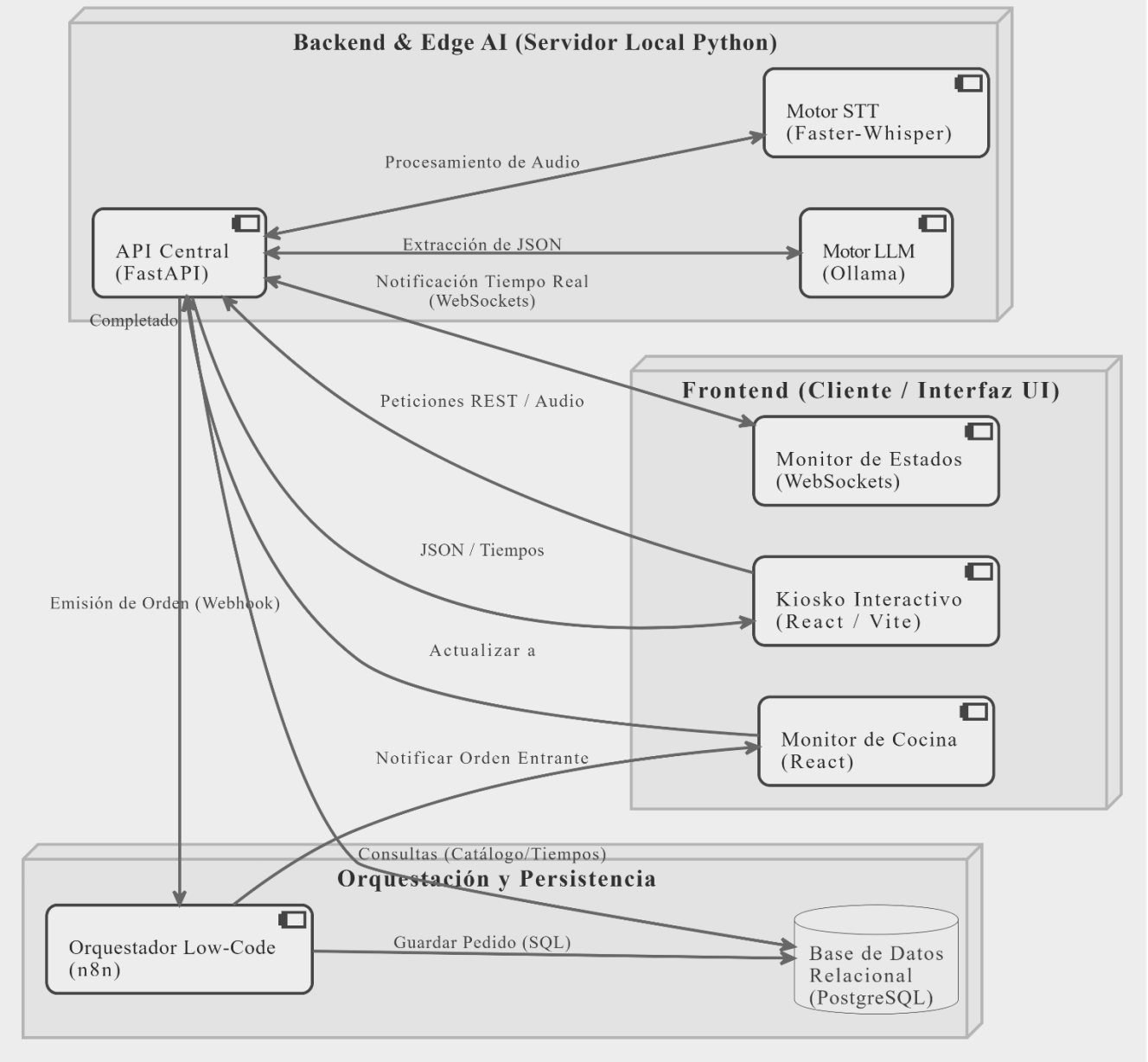


2.4.3. Arquitectura del sistema

La arquitectura adoptada para el Sistema de Mesero Digital corresponde a un ecosistema distribuido e híbrido que combina los paradigmas de Arquitectura Orientada a Servicios (SOA), Computación en el Borde (*Edge AI*) y Arquitectura Guiada por Eventos (EDA), diseñada bajo principios de baja latencia, alta disponibilidad, escalabilidad y procesamiento local. La Figura 5 detalla la organización de los componentes y su interacción técnica. El *Frontend* (Kiosco y Pantalla de Estados) se implementa utilizando React, encargado de la presentación visual y la captura de medios (audio). Este se comunica mediante peticiones HTTP/REST para operaciones síncronas y a través de WebSockets para la propagación asíncrona de eventos con el *Backend* desarrollado en FastAPI (Python). FastAPI actúa como el cerebro central, envolviendo los modelos locales de Inteligencia Artificial: *Faster-Whisper* para la transcripción acústica y *Llama/Mistral* (vía Ollama) para el razonamiento semántico. Para garantizar la independencia de los flujos de trabajo, la persistencia de datos recae sobre PostgreSQL, asegurando el cumplimiento de las propiedades transaccionales (ACID). Por su parte, la plataforma n8n actúa como el núcleo del paradigma guiado por eventos (*Event-Driven*), operando como un *middleware* de orquestación reactiva que captura los *Webhooks* de confirmación emitidos desde el *Backend* para enrutar las comandas de forma asíncrona hacia los monitores de producción en la cocina, garantizando con ello un desacoplamiento efectivo de los procesos y permitiendo que el ecosistema fluya de manera ininterrumpida.

Figura 5

Arquitectura del Sistema de Mesero Digital Asistido por IA.



2.4.4. Diseño del procesamiento de lenguaje natural (PLN)

Para dotar al sistema de capacidad cognitiva local y asegurar la privacidad de los datos bajo el paradigma de computación en el borde (*Edge AI*), se diseñó un módulo de Procesamiento de Lenguaje Natural (PLN) fundamentado en la orquestación de modelos de código abierto. El diseño lógico contempla un flujo de transformación de datos secuencial dividido en tres etapas principales:

A. Transcripción Acústica (Speech-to-Text)

En la primera etapa, el flujo de audio capturado es procesado por el modelo acústico *Faster-Whisper*, el cual genera una transcripción rápida en texto plano. En la etapa de diseño arquitectónico, se determinó que para optimizar el rendimiento en el hardware local y minimizar la latencia de respuesta, el modelo debía ser instanciado utilizando una cuantización de 8 bits (int8), garantizando velocidad sin comprometer severamente la precisión del reconocimiento en un ambiente ruidoso. Además, se planificó la inyección de un glosario de palabras clave (nombres de platos locales) en el *prompt* inicial del modelo acústico para mejorar el reconocimiento de modismos gastronómicos

B. Razonamiento Semántico y Extracción de Intenciones (LLM)

Posteriormente, la cadena de texto obtenida es enviada al Modelo de Lenguaje de Gran Escala (LLM) integrado mediante el motor *Ollama*. El sistema fue diseñado para operar bajo una técnica estricta de *Prompt Engineering* (Ingeniería de Instrucciones). Se definió que el LLM reciba un contexto dinámico que contiene la estructura del menú disponible, el estado actual del carrito del cliente y las reglas de límites comerciales. El diseño prohíbe explícitamente que la respuesta de la IA sea conversacional abierta; en su lugar, se restringe su dominio de acción obligando al modelo a emitir exclusivamente un objeto JSON estructurado que contenga la acción a ejecutar (agregar, modificar o eliminar).

C. Validación Estructural y Síntesis de Voz (TTS)

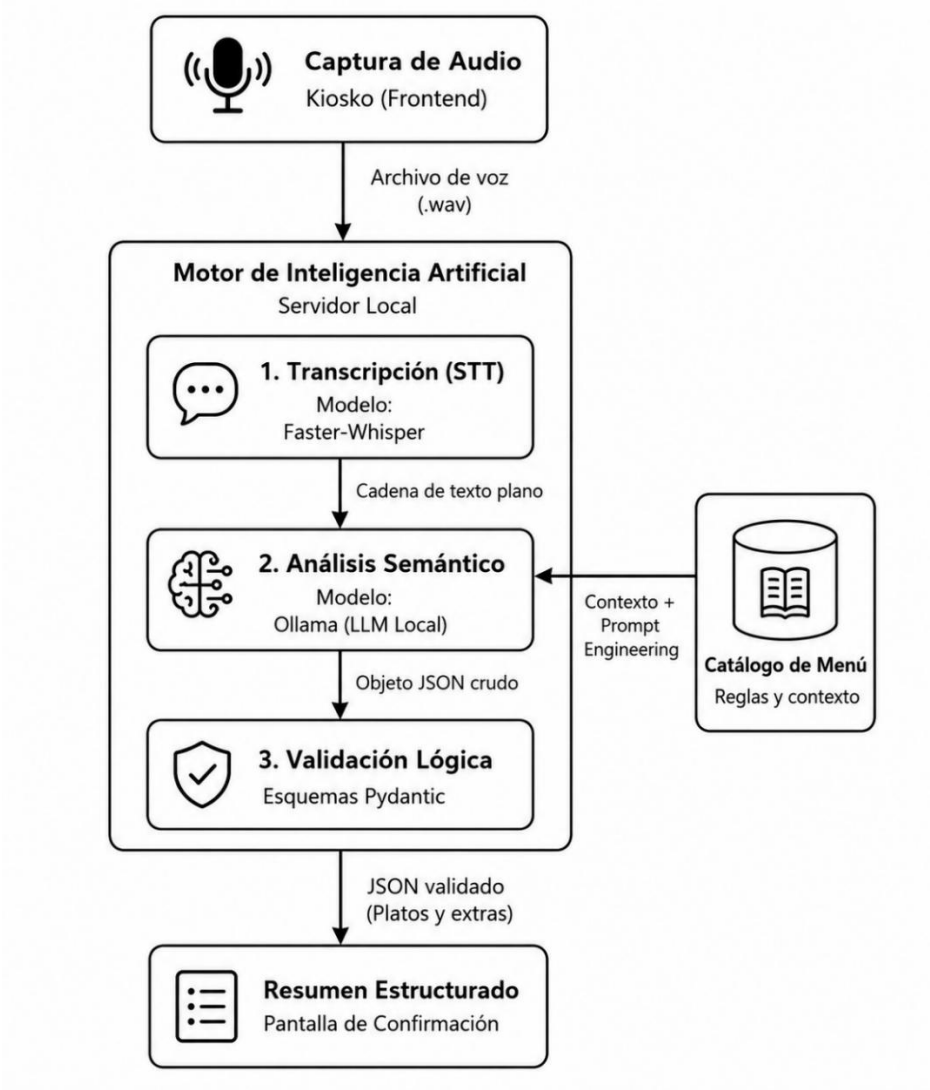
Finalmente, el diseño establece una capa de seguridad lógica para procesar la salida del LLM. Se determinó el uso de esquemas de validación estricta (*Pydantic*) para garantizar que el sistema extraiga con precisión matemática los identificadores de los platos, las cantidades y las restricciones dictadas por el comensal. Una vez que el JSON es validado, el texto de respuesta amigable generado por la IA es enviado a un motor de Síntesis de Voz (*Text-to-Speech*), el cual

renderiza el audio final que será reproducido en el kiosco, cerrando así el ciclo de interacción multimodal.

Para detallar gráficamente esta transformación de datos, la Figura 6 ilustra el diagrama de flujo del motor de Inteligencia Artificial. En el esquema se evidencia la transición desde la captura analógica de la voz hasta su estructuración semántica, destacando la inyección de contexto (Catálogo de Menú) y la validación final que precede a la confirmación visual en el kiosco.

Figura 6

Diagrama del Procesamiento de Lenguaje Natural



2.4.5. Diseño de la orquestación de datos con n8n

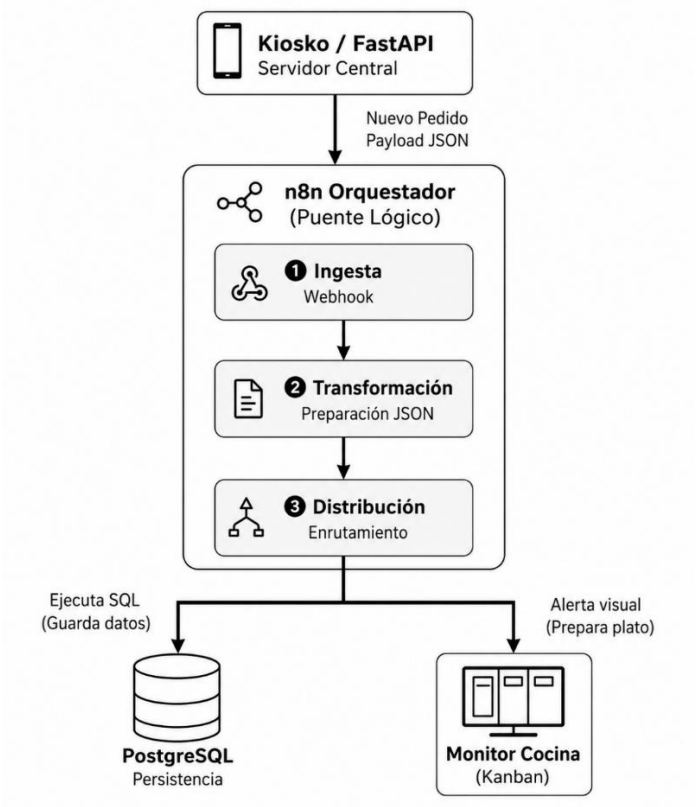
El diseño del flujo operativo delegó la responsabilidad del enrutamiento y la lógica de integración a la plataforma n8n, materializando de forma práctica el paradigma de la Arquitectura Guiada por Eventos (*Event-Driven Architecture - EDA*). Bajo este enfoque, se estableció un modelo de comunicación asíncrono donde los *Webhooks* actúan como los desencadenadores reactivos, y n8n opera como el puente lógico entre el núcleo de Inteligencia Artificial y la zona operativa del restaurante.

El flujo diseñado determina que, al confirmarse una orden válida en el kiosco, FastAPI emite de manera inmediata un *payload* hacia el orquestador. A partir de esta notificación, el flujo automatizado ejecuta secuencialmente tres acciones críticas: 1) la ingesta y transformación de los datos para su normalización estructural, 2) la ejecución de las sentencias SQL de inserción transaccional en la base de datos PostgreSQL, y 3) el desencadenamiento de la señal de actualización hacia el monitor de cocina para que refleje la nueva orden en tiempo real.

Para sintetizar visualmente este modelo de desvinculación y el enrutamiento paralelo de las tareas descritas, la Figura 7 expone el diagrama conceptual de la arquitectura de integración.

Figura 7

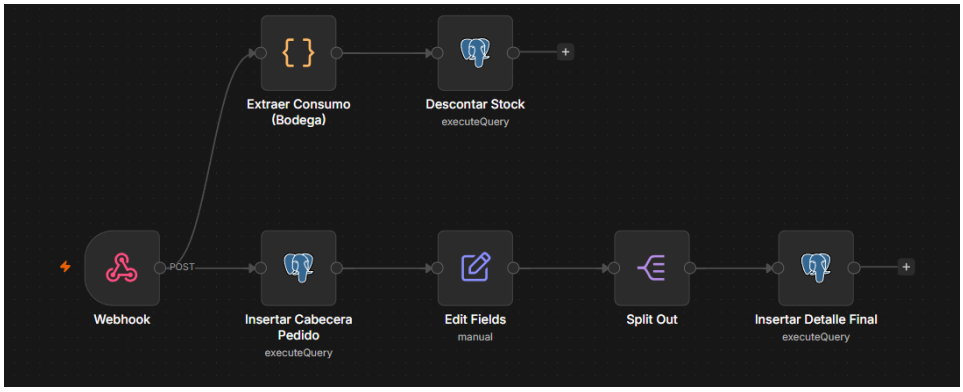
Diagrama de Orquestación de Datos con n8n



Para ilustrar gráficamente este proceso de desvinculación y coreografía de datos, la Figura 8 presenta el lienzo de diseño implementado en la plataforma n8n, evidenciando el enrutamiento paralelo de las tareas.

Figura 8

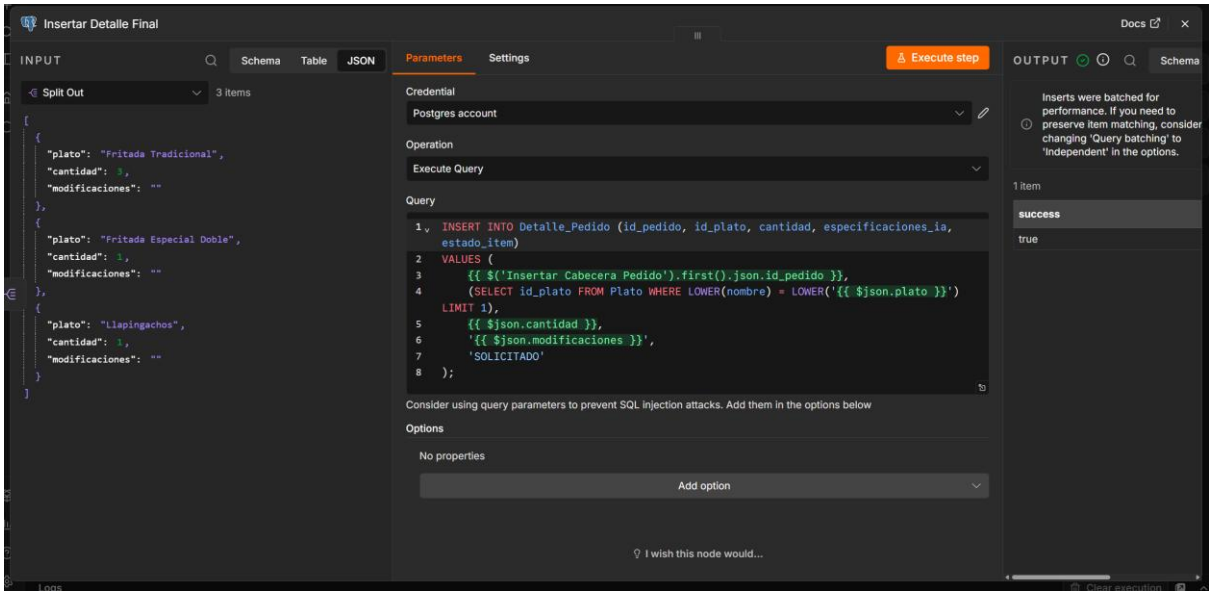
Lienzo del flujo de orquestación reactiva en n8n.



Este enrutamiento asegura que la transacción operativa y financiera quede respaldada con integridad en el motor de persistencia. Para evidenciar el mapeo de variables ejecutado por el orquestador, la Figura 9 expone la configuración interna del nodo de PostgreSQL, donde se emplean expresiones dinámicas para inyectar el nombre del comensal y el número de paleta directamente en la base de datos.

Figura 9

Configuración de inserción transaccional (Nodo PostgreSQL) en n8n.



2.4.6. Diseño de la base de datos relacional, inventario y gestión de sesiones

Para garantizar la integridad, escalabilidad y persistencia de las transacciones generadas por el Kiosco, se diseñó un modelo relacional normalizado hasta la Tercera Forma Normal (3NF) implementado sobre el motor PostgreSQL. Este diseño asegura el cumplimiento estricto de las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), fundamentales para soportar la concurrencia en un entorno gastronómico de alta demanda.

El esquema lógico se estructuró en cuatro macro-módulos interconectados, diseñados para reflejar la operatividad real del restaurante sin incurrir en redundancias de datos (como la creación de tablas físicas para "carritos de compras", cuyo estado efímero se delega exclusivamente a la memoria del *Frontend*):

Catálogo e Inventario Integral: Compuesto por las entidades *Categoria*, *Plato*, *Ingrediente* y la tabla puente *Receta*. En el ámbito de la ingeniería de operaciones, el sistema instrumenta una política de inventarios basada en el descuento automatizado por receta estándar en tiempo real. Esta política sirve de soporte digital para la regla de control físico **FIFO (First-In, First-Out / PEPS)** implementada en la franquicia, garantizando un rastreo preciso de materias primas perecederas y mitigando mermas operativas. Asimismo, se introduce formalmente en la entidad *Plato* el atributo técnico tiempo_prep_min (tiempo estándar de elaboración) junto con la bandera booleana requiere_coccion. La evaluación conjunta de estos campos desde la base de datos permite segmentar el flujo logístico, enrutando los productos de despacho inmediato hacia la barra y dejando los platos calientes como el insumo matemático fundamental para los algoritmos estocásticos de estimación de colas.

Transaccionalidad y Persistencia de Sesión: Conformado por las entidades *Mesa*, *Pedido* y *Detalle_Pedido*. Esta arquitectura permite agrupar múltiples interacciones bajo una única sesión operativa. La entidad *Detalle_Pedido* actúa como el núcleo de trazabilidad, registrando marcas de tiempo críticas (fecha_solicitud, fecha_inicio_preparacion, fecha_entrega) y mutando su atributo estado_item (Solicitado, Preparando, Entregado). Esto facilita el cierre de cuentas consolidadas y alimenta de forma directa el tablero de analítica de tiempos de atención.

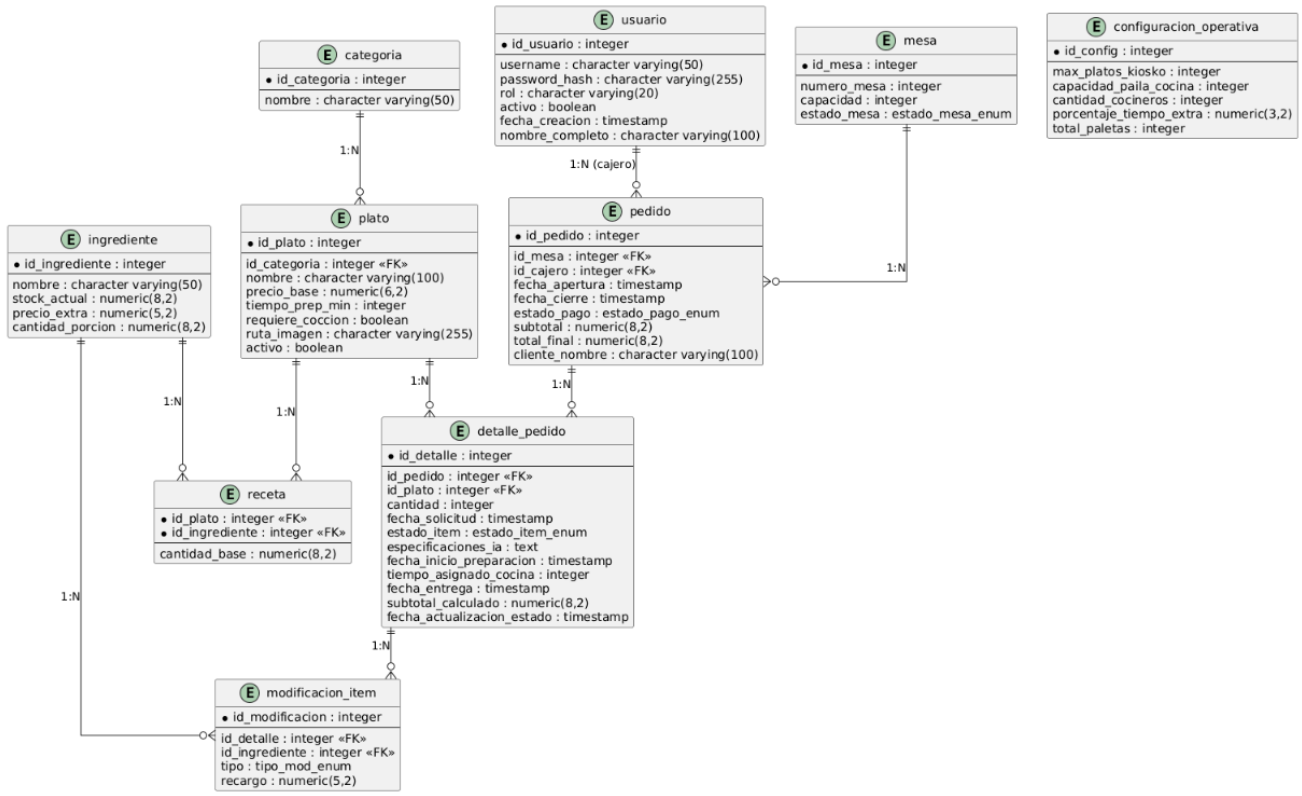
Personalización Dinámica (IA): Soportado por la entidad *Modificacion_Item*, este módulo otorga flexibilidad al procesamiento de lenguaje natural. Permite que las intenciones extraídas por el modelo de IA bajo la taxonomía del campo tipo (EXTRA, SIN, CAMBIO) se traduzcan en datos estructurados que ajustan dinámicamente el costo final del pedido mediante el atributo recargo.

Parametrización Dinámica y Teoría de Colas: Soportado por la entidad de dominio único *configuracion_operativa*. Este diseño erradica el antipatrón de valores estáticos en el código fuente (*hardcoding*). Atributos como max_platos_kiosco alimentan en tiempo real las reglas de restricción comercial del LLM, mientras que variables operativas como capacidad_paila_cocina y cantidad_cocineros dotan al sistema de la capacidad de recalcular dinámicamente los tiempos de espera frente a fluctuaciones en la plantilla del personal.

En la Figura 10 se muestra esquema relacional definitivo que sustenta la lógica de negocio, el control analítico de los inventarios y el flujo de estados operativos del sistema "Mesero Digital".

Figura 10

Diagrama Entidad-Relación



2.4.7. Algoritmo de estimación de tiempos de espera basados en teoría de colas

Para dar cumplimiento al objetivo específico de eliminar la opacidad operativa y gestionar de forma transparente las expectativas del comensal, se diseñó un algoritmo determinista-estocástico para el cálculo dinámico del tiempo de espera.

A diferencia de los modelos de colas convencionales de atención individual ($M/M/c$), el sistema propuesto fue diseñado bajo un Modelo de Procesamiento por Lotes con Sobrecarga Marginal. Este enfoque simula fielmente la realidad física del área de producción caliente (paila), donde múltiples platillos pueden cocinarse en paralelo de forma simultánea, incurriendo en un costo de tiempo extra fraccionario por cada unidad adicional agregada al lote.

A. Modelado Matemático del Sistema

Para el diseño de este algoritmo, se establecieron variables operativas dinámicas extraídas directamente de la base de datos relacional y de los parámetros de configuración del local:

Carga Total del Sistema (P_{total}): Es la sumatoria de las unidades que requieren cocción activa que ya se encuentran en cola (estados 'SOLICITADO' y 'PREPARANDO') más las unidades del nuevo pedido ingresado por el cliente. Si la orden no contiene elementos de cocción (ej. solo bebidas), el algoritmo retorna un tiempo de despacho inmediato (3 minutos).

Tiempo Base de Cocción (T_{max}): Se identifica el platillo que requiere el mayor tiempo de preparación de entre todos los activos en el sistema. Este valor establece la base temporal para la cocción de una tanda conjunta.

Restricciones Físicas (C y N): Se definen mediante la capacidad máxima de unidades que soporta la paila (C) y la cantidad de cocineros activos (N).

Holgura por Sobrecarga Marginal (α): Es un porcentaje de penalización temporal que se añade al tiempo base por cada platillo adicional dentro de un mismo lote de cocción.

El modelo matemático agrupa la carga total en lotes (tandas) completos y un lote parcial sobrante. El tiempo necesario para procesar un lote completamente lleno (T_{llena}) se define bajo la ecuación:

$$T_{llena} = T_{max} + [T_{max} \times \alpha \times (C - 1)]$$

Bajo la misma lógica, se calcula el tiempo del lote parcial con las unidades residuales ($T_{parcial}$). Finalmente, el algoritmo asume un entorno de concurrencia ideal (como si existiera un único "súper-servidor") y procede a dividir la carga temporal total de los lotes equitativamente entre los cocineros humanos reales disponibles (N), generando una curva de crecimiento de tiempos fluida y constante:

$$TiempoEstimado = \frac{(TandasCompletas \times T_{llena}) + T_{parcial}}{N}$$

B. Diseño del Pipeline de Datos (Lógica de Integración)

Para instrumentar esta lógica matemática en el ecosistema de software sin comprometer los tiempos de respuesta del kiosco, se diseñó el siguiente flujo asíncrono:

Agregación Transaccional: El servicio *Backend* intercepta el *payload* del carrito del cliente y ejecuta una consulta SQL prioritaria para sumar el inventario de platos activos en cocina, filtrando estrictamente aquellos cuya bandera requiere_coccion sea verdadera. Simultáneamente, extrae el valor del tiempo máximo (MAX(tiempo_prep_min)).

Cálculo de Algoritmo en Memoria: Las variables devueltas por el motor transaccional son inyectadas en la función matemática en la capa de la API (*FastAPI*). Aquí se determinan los operadores de división entera (para lotes completos) y módulo (para sobrantes), aplicando la sobrecarga marginal.

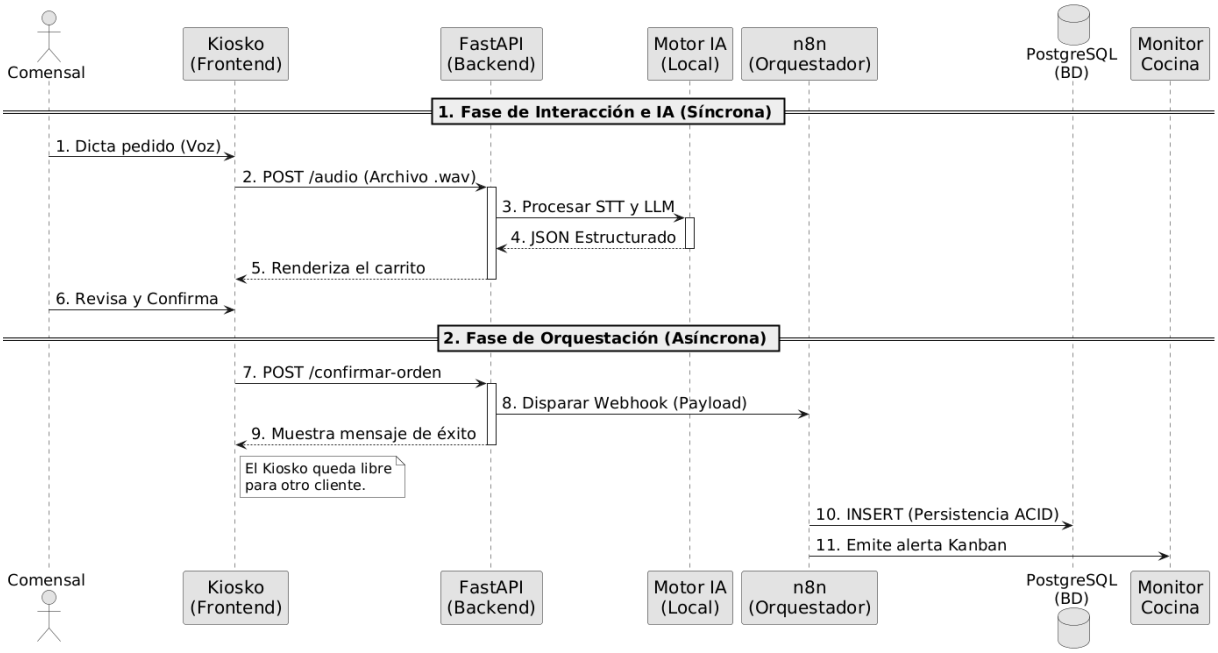
Despacho Reactivo: El tiempo de espera final calculado (redondeado a un número entero que representa los minutos de espera) es devuelto en formato JSON al cliente para actualizar de forma inmediata el indicador visual en la interfaz del usuario antes de la confirmación del pago.

2.4.8. Diagrama de secuencia transaccional

Para modelar la interacción dinámica entre los componentes distribuidos durante el ciclo de vida de una orden, se elaboró un diagrama de secuencia. Este diagrama describe el flujo síncrono y asíncrono desde el momento en que el comensal dicta su pedido a través del micrófono, hasta que la orden es registrada en la base de datos y visualizada en el monitor de la cocina. Como se observa en la Figura 11, el diagrama evidencia el aislamiento de los motores de IA (Faster-Whisper y Ollama) como servicios encapsulados en el backend, y resalta el rol de n8n como orquestador encargado de notificar a las terminales secundarias sin bloquear la respuesta de la interfaz principal.

Figura 11

Diagrama de Secuencia



2.4.9. Diseño de la interfaz visual (ux/ui)

La interfaz del Kiosco fue diseñada bajo principios de usabilidad táctil y accesibilidad, priorizando un alto contraste y tipografía legible para minimizar la carga cognitiva del comensal. La pantalla principal se estructuró en tres zonas funcionales: un panel lateral de navegación por categorías, una cuadrícula central para la visualización del menú, y un panel lateral de resumen de la orden. Como ilustra el wireframe en la Figura 12, el elemento central y más prominente de la interfaz es el botón de captura de audio (STT), estratégicamente posicionado en la zona inferior para facilitar la interacción multimodal, permitiendo al usuario transicionar fluidamente entre la selección manual y el dictado por voz.

Figura 12

Wireframe de la Interfaz Visual

2.4.10. Diseño del módulo de analítica descriptiva e inteligencia de negocios

Para dar cumplimiento al objetivo específico orientado al soporte de decisiones administrativas, se diseñó una arquitectura de extracción y agregación de datos relacionales capaz de transformar el historial transaccional de *PostgreSQL* en conocimiento estratégico. El diseño de este módulo rompe con los esquemas de reportes planos tradicionales y se estructura en un pipeline dividido en tres capas operativas independientes, eliminando por completo el acoplamiento directo y garantizando la fluidez de la interfaz administrativa:

Capa de Extracción y Agregación (Backend FastAPI): Se diseñó un conjunto de endpoints de consulta analítica bajo el prefijo /admin/reportes/. Estos componentes ejecutan funciones de agregación espacial y temporal en la base de datos utilizando cláusulas optimizadas (SUM, COUNT, AVG) combinadas con funciones de extracción cronológica (EXTRACT(DOW...) y EXTRACT(HOUR...)). El sistema parametriza dinámicamente las consultas según filtros

temporales ("Hoy", "Semana", "Mes") enviados desde el cliente, optimizando el uso de índices relacionales para mitigar el impacto en la memoria del servidor.

Capa de Visualización de Métricas (Frontend React + Chart.js): Diseñada bajo un enfoque modular de componentes reutilizables, esta interfaz traduce las matrices numéricas del servidor en representaciones visuales interactivas. El módulo de inventario y menús gestiona gráficos de barras horizontales independientes para correlacionar los ingresos monetarios frente a las cantidades físicas vendidas por plato. El área de operaciones emplea un gráfico de líneas continuas para contrastar los tiempos reales de cocción contra los Acuerdos de Nivel de Servicio (SLA) fijados en 20 minutos. Finalmente, el control de carga se instrumenta mediante un mapa de calor bidimensional con una cuadrícula de CSS Grid dinámica que mapea un rango horario de 9h a 22h contra los 7 días de la semana, calculando la opacidad del color (rgba) proporcionalmente a la densidad real de transacciones registradas.

Capa de Inferencia Consultiva (Edge AI Analytical Core): Corresponde al núcleo avanzado de inteligencia de negocios. Se diseñó un flujo donde el frontend consolida las matrices de KPIs operativos actuales y los despacha en un payload unificado hacia el endpoint /generar-analisis-ia. Este endpoint actúa como un puente de inyección contextual que alimenta al modelo local *Llama 3* a través de *Ollama*. El modelo es gobernado por una ingeniería de instrucciones sistémica con una temperatura de 0.0, anulando la creatividad del algoritmo y forzándolo a actuar como un consultor gastronómico automatizado que detecta cuellos de botella y genera sugerencias en formato Markdown.

Para formalizar las variables relacionales que sustentan la construcción de este módulo y delimitar su origen en la base de datos, la Tabla 8 detalla la especificación lógica de los indicadores analíticos diseñados.

Tabla 8

Especificación Lógica e Indicadores del Módulo de Analítica Descriptiva

Indicador Analítico	**Módulo del Dashboard**	**Componente Visual / Gráfico**	**Ecuación Lógica / Origen en Base de Datos**
Ingresos Totales (KPI)	Resumen Ejecutivo	Tarjeta con minigráfico (Sparkline)	$\sum(\text{total}_{\text{final}})$ de la tabla $Pedido$ filtrado por periodo.
Ticket Promedio (KPI)	Resumen Ejecutivo	Tarjeta con minigráfico (Sparkline)	$\frac{\sum(\text{total}_{\text{final}})}{COUNT(id\ pedido)}$ de la tabla $Pedido$

Top Ventas e Ingresos	Ventas y Menú	Barras Horizontales (Chart.js)	$SUM\ (cantidad)$ y SUM (subtotal_calculado) agrupado por id_plato en *Detalle_Pedido*.
Distribución de Horas Pico	Operación y Cocina	Mapa de Calor (CSS Grid)	$COUNT(id\ pedido)$ agrupado por EXTRACT(DOW) y EXTRACT(HOUR) en *Pedido*.
Eficiencia de Preparación	Operación y Cocina	Líneas Continuas (Chart.js)	AVG (fecha_entrega - fecha_inicio_preparacion) agrupado por hora en *Detalle_Pedido*.
Diagnóstico Operativo	Asesor Operativo IA	Contenedor dinámico (Markdown)	Inferencia determinista de *Llama 3* inyectando matrices de KPIs y Horas Pico.

2.5. Herramientas de desarrollo y tecnologías (stack tecnológico)

Para la construcción del "Sistema de Mesero Digital Asistido por IA", se seleccionó un conjunto de herramientas de código abierto (*Open Source*) orientadas al alto rendimiento, la baja latencia y la capacidad de ejecución local. El ecosistema tecnológico se divide en cinco capas principales:

2.5.1. Capa de presentación (frontend e interfaces)

Esta capa gestiona la interacción con el comensal y el personal del restaurante.

2.5.1.1. React.js

Biblioteca principal de JavaScript utilizada para el desarrollo de las interfaces de usuario (El Kiosco y el Monitor de Cocina). Se eligió por su arquitectura basada en componentes, que permite un manejo eficiente del estado global (ej. el carrito de compras) y actualizaciones del DOM en tiempo real sin recargar la página.

2.5.1.2. Tailwind CSS

Framework de diseño utilizado para la maquetación visual. Permite la creación de interfaces de alto contraste y diseño responsivo (*Responsive Design*) mediante clases de utilidad, garantizando la accesibilidad táctil en la pantalla del kiosco.

2.5.1.3. MediaRecorder API

Interfaz nativa de los navegadores web (HTML5) empleada para capturar el flujo de audio del micrófono del comensal y empaquetarlo en fragmentos (blobs) para su envío al servidor.

2.5.2. Capa de lógica de negocio (backend)

Encargada de recibir las peticiones, procesar la concurrencia y servir de puente con los motores de Inteligencia Artificial.

2.5.2.1. Python (v3.10+)

Lenguaje de programación principal del lado del servidor, elegido por su robusto ecosistema de librerías nativas para Inteligencia Artificial y procesamiento de datos.

2.5.2.2. FastAPI

Framework web moderno y de alto rendimiento para la construcción de la API REST. Se seleccionó por su soporte nativo para programación asíncrona (async/await), indispensable para gestionar el flujo de audio sin bloquear otras peticiones de la red, y por su generación automática de documentación interactiva (Swagger/OpenAPI).

2.5.2.3. Uvicorn

Servidor ASGI (*Asynchronous Server Gateway Interface*) utilizado para ejecutar FastAPI, proporcionando la velocidad y concurrencia necesarias para el manejo de WebSockets.

2.5.2.4. Pydantic

Librería de validación de datos integrada en FastAPI. Se utiliza para definir los esquemas estrictos de las entidades (ej. el formato del menú) y garantizar que el objeto JSON devuelto por la IA sea estructuralmente correcto antes de procesarlo.

2.5.3. Capa de inteligencia artificial cognitiva (edge ai)

Responsable del procesamiento del lenguaje natural de manera local, asegurando privacidad y tolerancia a fallos de red.

2.5.3.1. Faster-Whisper

Implementación optimizada del modelo acústico de OpenAI. A diferencia del modelo estándar, Faster-Whisper utiliza el motor *CTranslate2*, lo que reduce drásticamente el consumo de memoria RAM/VRAM y acelera la transcripción de voz a texto (STT) en hardware de consumo comercial.

2.5.3.2. Ollama

Plataforma de orquestación local para Modelos de Lenguaje Grande (LLMs). Permite descargar, cuantizar (comprimir) y ejecutar modelos de razonamiento avanzado de forma nativa en el servidor sin depender de APIs en la nube.

2.5.3.3. Modelo Base (Llama 3 / Mistral)

El cerebro semántico del sistema, ejecutado a través de Ollama. Su función exclusiva es recibir el texto transcrito y aplicar *Prompt Engineering* para realizar la extracción de entidades (intenciones, platillos, cantidades y restricciones).

2.5.4. Capa de orquestación y persistencia de datos

2.5.4.1. n8n

Plataforma de automatización de flujos de trabajo basada en nodos. Actúa como el *middleware* (puente lógico) del sistema. Se utiliza para recibir los *Webhooks* de FastAPI, transformar la estructura de datos, y enrutar las sentencias SQL hacia la base de datos y las alertas hacia el monitor de cocina (Desvinculación de procesos).

2.5.4.2. PostgreSQL

Sistema de Gestión de Bases de Datos Relacional (RDBMS) de nivel empresarial. Seleccionado por su estricto cumplimiento de las propiedades ACID y su fiabilidad transaccional para el manejo del catálogo, inventario y facturación del kiosco.

2.5.5. Capa de infraestructura y herramientas de soporte

2.5.5.1. Docker & Docker Compose

Tecnología de contenerización utilizada para encapsular y desplegar de manera aislada los servicios de n8n, PostgreSQL y Ollama. Garantiza que el sistema funcione de manera idéntica en el entorno de desarrollo y en el servidor de producción del restaurante.

2.5.5.2. Git / GitHub

Sistema de control de versiones distribuido utilizado para el seguimiento del código fuente, el manejo de ramas (*branches*) durante los Sprints y el respaldo del proyecto.

2.5.5.3. PlantUML

Herramienta de modelado en texto empleada para la generación automatizada y estandarizada de los diagramas de arquitectura, bases de datos y secuencias del sistema.

2.6. Diseño del plan de pruebas

2.6.1. Pruebas funcionales

La prueba del software es un elemento crítico para la garantía de calidad del producto final y representa una revisión formal de las especificaciones, del diseño y de la codificación. El objetivo principal de esta fase es diseñar casos que sistemáticamente saquen a la luz diferentes clases de errores, optimizando el tiempo y el esfuerzo invertido.

Para la validación del "Sistema de Mesero Digital Asistido por IA", se definió una estrategia orientada a las pruebas de caja negra funcionales. Estas pruebas realizan evaluaciones sobre la interfaz del programa (entradas y salidas) sin necesidad de conocer la lógica interna del código fuente. Particularmente, se estructuraron casos de prueba basados en escenarios, los cuales se enfocan directamente en lo que hace el usuario final.

Este proceso implica la generación de casos de prueba que determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para cada objetivo particular del sistema. Finalmente, se definieron pruebas funcionales para asegurar que el software, ensamblado en su totalidad, funciona según las expectativas operativas del restaurante.

En la Tabla 9 se presenta la plantilla metodológica base que se diseñó y estandarizó para la documentación de todos los casos de prueba de aceptación del sistema.

Tabla 9

Plantilla Estándar para el Diseño de Casos de Prueba

<table>
<tr><td colspan="4">PRUEBA DE ACEPTACIÓN N.º [ID]</td></tr>
<tr><td colspan="4">Nombre: [Nombre funcional del caso de prueba]</td></tr>
<tr><td colspan="4">Descripción: [Descripción detallada del objetivo de la prueba y lo que se espera validar]</td></tr>
<tr><td colspan="4">Escenarios</td></tr>
<tr><td>Condición</td><td>Acción Inyectada</td><td>Resultado Esperado</td><td>Resultado Obtenido</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">Observaciones:</td></tr>
<tr><td colspan="4">[Espacio destinado para documentar anomalías, tiempos de respuesta o notas técnicas tras la ejecución de la prueba]</td></tr>
</table>

Para validar la correcta operatividad de la interfaz gráfica y la gestión del estado del cliente, se diseñó el primer caso de prueba. La Tabla 10 detalla los escenarios orientados a la interacción híbrida con el menú, garantizando que el sistema cargue el catálogo dinámico, procese la

eliminación de platillos y aplique la lógica financiera al inyectar ingredientes extra mediante el modal de personalización.

Tabla 10

Caso de Prueba: Interacción Híbrida y gestión de Estado en el Frontend

PRUEBA FUNCIONAL N.º 1			
Nombre: Gestión Reactiva del Carrito y Modificaciones Manuales			
Descripción: El sistema debe cargar el catálogo desde la base de datos, permitir la adición/eliminación de platillos y habilitar el modal de edición para inyectar ingredientes extra, actualizando el subtotal en tiempo real en la vista del cliente.			
Escenario / Condición	**Acción del Usuario**	**Resultado Esperado**	**Resultado Obtenido**
Carga del Menú y Fotos	Carga inicial del Kiosco.	El sistema renderiza la cuadrícula con información de la base de datos e imágenes servidas desde archivos estáticos locales.	*(Se evalúa en Cap. 3)*
Eliminación de Ítem	Toque en el botón (🗑️) de un plato en el carrito.	Mutación del arreglo en memoria, eliminando el objeto y recalculando el total matemático de la comanda.	*(Se evalúa en Cap. 3)*
Inyección de Extras	Toque en Editar (📝) y selección del *checkbox* de un ingrediente extra.	El sistema busca el recargo en la base de datos, lo suma al precio base del plato y actualiza el subtotal en la vista.	*(Se evalúa en Cap. 3)*

El núcleo innovador del kiosco radica en su capacidad cognitiva y multimodal. Por ello, la Tabla 11 expone los escenarios de prueba diseñados para evaluar el Procesamiento de Lenguaje Natural (PLN). Estas pruebas planifican la verificación de los modelos acústicos y de lenguaje, asegurando que extraigan intenciones complejas y retornen un formato JSON determinista que respete las reglas matemáticas del negocio.

Tabla 11

Caso de Prueba: Procesamiento de Lenguaje Natural y Transaccionalidad IA

PRUEBA FUNCIONAL N.º 2			
Nombre: Captura Acústica y Razonamiento Lógico Estricto			
Descripción: El motor de IA debe transcribir el audio analógico, extraer intenciones de agregación, subdivisión o eliminación, y retornar un JSON estructurado que sirva como *payload* para el carrito de compras.			
Entrada de Audio (Dictado)	**Intención Detectada**	**Resultado Esperado**	**Resultado Obtenido**
"Quiero tres fritadas, pero dos sin mote"	División Matemática.	El LLM emite JSON con 2 acciones AGREGAR: Una cantidad 1 normal, y otra cantidad 2 con el objeto modificador SIN Mote.	*(Se evalúa en Cap. 3)*
"Puedes quitarme un combo 1 del carrito"	Resta de Inventario.	El LLM emite una acción QUITAR apuntando al platillo con cantidad 1.	*(Se evalúa en Cap. 3)*
"A mi Fritada que ya pedí ponle extra aguacate"	Modificación sobre existente.	El LLM elimina la fritada antigua (QUITAR) e inserta una nueva (AGREGAR) con la etiqueta EXTRA.	*(Se evalúa en Cap. 3)*

Un factor determinante en la resiliencia del sistema es la prevención de transacciones erróneas. En la Tabla 12 se presentan las pruebas correspondientes a la validación reactiva, diseñadas para asegurar que el Kiosco bloquee comandas que excedan los límites, valide stock y dimensione los tiempos de preparación.

Tabla 12

Caso de Prueba: Validación de Reglas de Negocio, Stock y Tiempos

PRUEBA FUNCIONAL N.º 3			
Nombre: Restricciones de Sistema y Cálculo de Cola Asíncrono			
Descripción: El cliente debe ejecutar peticiones asíncronas para auditar que el pedido cumpla con los límites comerciales, exista disponibilidad física en bodega y se calcule el tiempo proyectado.			
Escenario Operativo	**Acción en el Kiosco**	**Resultado Esperado**	**Resultado Obtenido**
Límite Comercial Alcanzado	El usuario agrega platos hasta superar el límite parametrizado (Ej. 15).	El sistema activa la bandera de límite excedido, despliega una alerta y desactiva el botón de confirmación.	*(Se evalúa en Cap. 3)*
Validación de Bodega	El usuario solicita un ingrediente con stock físico 0.	El endpoint retorna la alerta de déficit y React renderiza la advertencia de "Bodega Insuficiente".	*(Se evalúa en Cap. 3)*
Inferencia de Tiempos	Confirmación de pedido en un horario de alta saturación (Hora pico).	El backend aplica la política de lotes y devuelve un tiempo proyectado elevado (Ej. 25-30 min).	*(Se evalúa en Cap. 3)*

Una vez confirmado el pedido, se debe garantizar el correcto enrutamiento hacia la cocina. La Tabla 13 describe el diseño de la prueba para la orquestación mediante *n8n*, la cual evaluará la inserción de cabeceras y detalles en la base de datos de manera automatizada.

Tabla 13

Caso de Prueba: Orquestación y Enrutamiento de Comandas

PRUEBA FUNCIONAL N.º 4			
Nombre: Orquestación Transaccional mediante Webhooks y n8n			
Descripción: Al confirmar la orden, el sistema debe disparar un Webhook hacia n8n, el cual iterará el arreglo de platos mediante un nodo *Split Out* e insertará la cabecera y el detalle en PostgreSQL.			
Estado del Sistema	**Acción del Usuario / Sistema**	**Resultado Esperado**	**Resultado Obtenido**

Pedido con múltiples ítems	Clic en "Confirmar Orden".	Emisión de petición al Webhook de n8n. El flujo inserta 1 registro maestro y N registros de detalle iterados.	*(Se evalúa en Cap. 3)*
Inserción Relacional	n8n procesa la cabecera del pedido.	El nodo de Detalle utiliza dinámicamente la llave foránea (id_pedido) retornada por el nodo Cabecera.	*(Se evalúa en Cap. 3)*
Falla del Orquestador	Desconexión del servicio n8n.	El kiosco captura el fallo de red (Timeout) y muestra una alerta amigable al cliente.	*(Se evalúa en Cap. 3)*

El control de producción exige sincronización entre el área de preparación y las comandas entrantes. La Tabla 14 detalla las pruebas diseñadas para el monitor Kanban, validando que el personal pueda operar transiciones de estado bajo una política estricta de primero en entrar, primero en salir (FIFO).

Tabla 14

Caso de Prueba: Actualización de Estados en Producción

PRUEBA FUNCIONAL N.º 5			
Nombre: Monitor de Cocina y Transición de Estados			
Descripción: El panel de producción debe recibir comandas asíncronamente y permitir a los cocineros cambiar el ciclo de vida del pedido, impactando la base de datos en tiempo real.			
Estado de la Orden	**Acción del Sistema / Cocinero**	**Resultado Esperado**	**Resultado Obtenido**
Nueva Confirmación	Sistema en reposo. Ingresa nueva orden en BD.	El ciclo de *Short Polling* detecta la orden y renderiza la tarjeta en la columna "Solicitado" agrupando por mesa.	*(Se evalúa en Cap. 3)*
Orden Completada	Clic táctil en el botón "LISTO".	El sistema despacha el llamado a la API, registra la hora de entrega en BD y elimina la tarjeta visualmente de la cola.	*(Se evalúa en Cap. 3)*

Finalmente, la capacidad del sistema para apoyar decisiones gerenciales debe ser validada. La Tabla 15 expone el diseño de pruebas para el Dashboard Administrativo, asegurando el cálculo de mapas de calor y la obtención de análisis a través del Oráculo IA.

Tabla 15

Caso de Prueba: Analítica de Datos y Reportes

PRUEBA FUNCIONAL N.º 6			
Nombre: Analítica Descriptiva y Reporte Gerencial IA			
Descripción: El módulo administrativo debe extraer KPIs, construir gráficas dinámicas e inyectar el contexto hacia el LLM local para obtener sugerencias operativas autónomas.			
Módulo Seleccionado	**Acción del Administrador**	**Resultado Esperado**	**Resultado Obtenido**
Operación y Cocina	Acceso a la vista de "Mapa de Calor".	Renderización del mapa calculando la intensidad de color basada en la saturación histórica por hora y día.	*(Se evalúa en Cap. 3)*
Asesor Operativo IA	Clic en el botón "Generar Análisis Real".	Consolidación de *payload* hacia Ollama, retornando texto formateado en *Markdown* con detección de cuellos de botella.	*(Se evalúa en Cap. 3)*

2.6.2. Diseño del plan de evaluación del desempeño y usabilidad

Para complementar las pruebas funcionales de caja negra y validar el cumplimiento del último objetivo específico de la investigación, se estructuró un plan de evaluación independiente orientado a la medición empírica del rendimiento técnico, la eficiencia operativa y la usabilidad en el entorno operacional del restaurante. El diseño metodológico rechaza el uso de simuladores en entornos controlados y establece que las pruebas se ejecutarán en la sucursal física bajo condiciones reales de concurrencia y ruido ambiental (estimado entre 65 dBA y 75 dBA).

El diseño experimental establece dos estrategias complementarias de recolección de datos:

Auditoría de Métricas de Rendimiento Técnico (Cuantitativo): Para evitar el sesgo humano en la medición de la velocidad del sistema, se planificó la instrumentación de interceptores de tiempo en el código fuente de *FastAPI*. Utilizando la función de alta precisión time.perf_counter(), el sistema registrará autónomamente la diferencia temporal (ΔT) en la ejecución de un lote de prueba proyectado de $N = 50$ comandas de voz. Este protocolo permitirá aislar la latencia del motor acústico (*Faster-Whisper*), el tiempo de inferencia semántica (*Llama 3*) y el tiempo de orquestación en *n8n*. Adicionalmente, la eficiencia

operativa se medirá contrastando el Tiempo de Atención Promedio (TPA) automatizado frente a los tiempos históricos de la toma de pedidos manual.

Evaluación Cualitativa y Sistematización de Usabilidad: Para evaluar la carga cognitiva y la aceptación de las interfaces, se descartó el uso de encuestas estandarizadas abstractas, optando por un protocolo de pruebas de usuario guiadas seguidas de entrevistas semiestructuradas post-servicio. Se planificó la selección de una muestra representativa de $N = 15$ comensales y $N = 3$ operadores de cocina. El diseño de la entrevista cualitativa evaluará por separado cada módulo interactivo (Kiosco táctil, dictado por voz, modal de extras, algoritmo de tiempos y monitor Kanban) para identificar fricciones físicas de uso (UX) y niveles de adaptabilidad tecnológica.

Para operacionalizar este plan metodológico y establecer los criterios de validación que el software deberá alcanzar en la fase de despliegue, la Tabla 16 expone la matriz de variables de desempeño y usabilidad del sistema con sus respectivos umbrales de éxito.

Tabla 16

Matriz de Variables de Evaluación, Desempeño y Usabilidad del Sistema

Dimensión	**Variable de Medición**	**Método y Herramienta de Recolección**	**Métrica / Unidad de Medida**	**Umbral de Éxito Diseñado**
Técnica	Latencia de Transcripción	Interceptores lógicos en *FastAPI* (*Faster-Whisper*)	Segundos por ráfaga de audio	≤1.5 segundos por comanda
Técnica	Latencia de Estructuración	Interceptores lógicos en *FastAPI* (*Llama 3*)	Segundos por inferencia JSON	≤3.0 segundos por comanda
Operativa	Tasa de Error Transaccional	Contraste de *logs* de IA vs. Correcciones en Caja	Porcentaje (%) de platos erróneos	≤2.0% de discrepancia
Operativa	Reducción del TPA	Comparativa de *timestamps* en tabla *Pedido*	Descenso porcentual (%)	≥35.0% frente al modelo manual
Usabilidad	Aceptación del Comensal	Pruebas guiadas e interacciones multimodales ($N = 15$)	Porcentaje de retroalimentación positiva	≥80.0% de aprobación
Usabilidad	Adaptabilidad en Cocina	Entrevistas post-servicio del Monitor Kanban ($N = 3$)	Porcentaje de retroalimentación positiva	≥75.0% de aprobación

2.7. FASE III: Codificación y Desarrollo

En esta fase se materializa la construcción del sistema de información bajo el Modelo de Desarrollo Incremental. A continuación, se documenta la ejecución técnica y lógica de la programación, dividida en las seis iteraciones (Sprints) planificadas previamente en el *Sprint Backlog*. En cada iteración se detalla el cumplimiento de las Historias de Usuario (HU) y la resolución de sus respectivas tareas técnicas.

2.7.1. Ejecución del Sprint 1: Estructura Base e Interfaz de Menú

Durante el primer Sprint (2 semanas), se dio cumplimiento a las historias de usuario HU-01 (Inicialización) y HU-02 (Visualización). Se abordaron las tareas técnicas enfocadas en el establecimiento de la arquitectura base del *Frontend* en React. Se implementó el enrutador principal para manejar las vistas independientes (Kiosco, Monitor de Cocina, Pantalla de Turnos y Panel de Administración), asegurando el bloqueo de navegación externa mediante la API *Fullscreen* para emular un entorno de Kiosco inmersivo. La Figura 13 ilustra la configuración de este enrutamiento base y el manejo del estado global de la aplicación.

Figura 13

Enrutamiento principal de vistas y bloqueos lógicos en React.

```
<Router>
    {/* Barra de navegación técnica (Simulador de múltiples terminales) */}
    <nav style={{
      background: '#1e293b',
      padding: '15px',
      display: 'flex',
      gap: '25px',
      justifyContent: 'center',
      borderBottom: '3px solid #334155',
      flexWrap: 'wrap'
    }}>
      <Link to="/" style={linkStyle}> Vista Kiosco</Link>
      <Link to="/cocina" style={linkStyle}> Monitor Cocina</Link>
      <Link to="/turnos" style={linkStyle}> Pantalla Turnos</Link>
      <Link to="/caja" style={adminStyle}> Caja / Cobros</Link>
      <Link to="/AdminDashboard" style={adminStyle}> Administración</Link>
    </nav>

    {/* Enrutador de Componentes */}
    <Routes>
      {/* Entorno interactivo del comensal */}
      <Route path="/" element={<Kiosco />} />
```

```
        {/* Terminal secundaria del cocinero */}
        <Route path="/cocina" element={<MonitorCocina />} />

        {/* Pantalla pública de visualización */}
        <Route path="/turnos" element={<PantallaTurnos />} />

        {/* Terminal del cajero para cobrar y despachar */}
        <Route path="/caja" element={<Caja />} />

        {/* Panel de control del gerente/administrador */}
        <Route path="/AdminDashboard" element={<AdminDashboard />} />
      </Routes>
    </Router>
```

Simultaneamente, para cumplir con el despliegue del catálogo dinámico, se diseñó el modelo relacional en *PostgreSQL* y se construyó el núcleo del *Backend* en *FastAPI*. Se desarrollaron los *endpoints* necesarios para exponer las categorías y los productos con sus respectivos precios y tiempos de cocción hacia la interfaz responsiva. En la Figura 14 se detalla la construcción de la API RESTful que permite la consulta asíncrona del menú principal, conectándose directamente con la capa de persistencia de datos.

Figura 14

Endpoint en FastAPI para la extracción relacional del catálogo de productos.

```
@router.get("/menu")
def obtener_menu():
    print("🗄️ Consultando menú dinámico en PostgreSQL...")
    conn = None
    cursor = None
    try:
        conn = get_db_connection()
        cursor = conn.cursor(cursor_factory=RealDictCursor)

        cursor.execute("""
            SELECT c.id_categoria, c.nombre AS categoria_nombre,
                   p.id_plato, p.nombre AS plato_nombre, p.precio_base, p.tiempo_prep_min
            FROM Categoria c
            JOIN Plato p ON c.id_categoria = p.id_categoria
        """)
        filas = cursor.fetchall()

        categorias_dict = {}
        for fila in filas:
            id_cat = fila['id_categoria']
            if id_cat not in categorias_dict:
```

```
            categorias_dict[id_cat] = {
                "id_categoria": id_cat,
                "nombre": fila['categoria_nombre'],
                "platos": []
            }

        categorias_dict[id_cat]["platos"].append({
            "id_plato": fila['id_plato'],
            "nombre": fila['plato_nombre'],
            "descripcion": f"Tiempo estimado: {fila['tiempo_prep_min']} min",
            "precio": float(fila['precio_base'])
        })

    return {"categorias": list(categorias_dict.values())}

except Exception as e:
    print(f"❌ Error de base de datos: {e}")
    return {"error": "No se pudo conectar a la base de datos"}
finally:
    if cursor: cursor.close()
    if conn: conn.close()
```

2.7.2. Ejecución del Sprint 2: Núcleo Conversacional de Inteligencia Artificial

Esta iteración representó el núcleo de la investigación de la arquitectura cognitiva, abarcando de forma integral el desarrollo de las historias de usuario HU-03 (Captura Voz) y HU-04 (Interpretación IA) a través de la ejecución programada de las tareas técnicas T3-1 a T4-4. El propósito fundamental de este ciclo de dos semanas fue materializar un pipeline multimodal capaz de procesar señales de audio analógicas del entorno real, transcribirlas a texto plano de alta precisión de forma local y extraer computacionalmente las intenciones del comensal sin depender de servicios en la nube.

Para dar cumplimiento a las tareas de captura y conversión acústica (T3-1, T3-2 y T3-4), se configuró en el servidor central un endpoint asíncrono bajo la plataforma *FastAPI*. En la capa del cliente (*React*), se orquestó la captura mediante el flujo del micrófono de la interfaz binaria empleando la API MediaRecorder. Al emitirse la petición *Multipart* hacia el backend, el archivo binario es interceptado y procesado de inmediato por el modelo acústico optimizado *Faster-Whisper*. Para contrarrestar la latencia de hardware y aislar el ruido intrínseco de un restaurante, el motor acústico fue instanciado utilizando una cuantización entera de 8 bits (int8) en conjunto con un diccionario léxico restrictivo (initial_prompt) inyectado en el decodificador de haces (*beam search*), forzando el acoplamiento fonético de modismos locales (como "Fritada", "Llapingachos" o "Mote"). La lógica transaccional de recepción del archivo de audio,

el control de hilos y la llamada al motor de transcripción en el borde se evidencia formalmente en la Figura 15.

Figura 15

Controlador acústico y transcripción asíncrona optimizada con Faster-Whisper

```
from faster_whisper import WhisperModel

# Instanciación del modelo con cuantización para Edge AI
modelo_whisper = WhisperModel("small", device="cpu", compute_type="int8")

def procesar_audio_con_ia(ruta_temporal_audio: str, carrito_actual: str):
    # FASE A: ESCUCHAR (Speech-to-Text)
    glosario_zita = "Fritada, llapingachos, mote, tostado, maduro, chicharrón, empanadas, yahuarlocro, menú, porción, pedido, colas."

    # Transcripción optimizada con decodificador de haces y diccionario
    segmentos, info = modelo_whisper.transcribe(
        ruta_temporal_audio,
        beam_size=5,
        language="es",
        initial_prompt=glosario_zita
    )

    texto_completo = " ".join([segmento.text for segmento in segmentos]).strip()
    return texto_completo
```

Posteriormente, enfocado en las tareas de ingeniería lingüística e interpretación semántica (T4-1, T4-2 y T4-3), el texto plano depurado fue sometido a una capa de Procesamiento de Lenguaje Natural (PLN). Para esto, se estructuró un motor relacional que toma la transcripción y la procesa localmente mediante el modelo de lenguaje de gran escala (LLM) *Llama 3* administrado por *Ollama*. Con la finalidad de erradicar alucinaciones cognitivas y forzar un comportamiento determinista, se aplicaron técnicas avanzadas de *Prompt Engineering* sistémico. El prompt inyecta dinámicamente tres variables críticas desde la persistencia: el menú real y actualizado del restaurante, el estado de memoria del carrito actual y los límites comerciales parametrizados de la base de datos, restringiendo la salida del modelo exclusivamente a una estructura relacional JSON estricta y prohibiendo respuestas conversacionales abiertas. La Figura 16 detalla las instrucciones del sistema implementadas para gobernar el comportamiento cognitivo del LLM local.

Figura 16

Ingeniería de instrucciones de prompts sistémicos para la restricción analítica del modelo Llama 3.

```
# FASE B: RAZONAR INTENCIONES
    print("🤖 2/3 Analizando intención...")
    menu_real = obtener_menu_disponible()
    menu_formateado = "\n".join([f"- "{plato}"" for plato in menu_real])
    ingredientes_reales = obtener_ingredientes_disponibles()
    ingredientes_formateados = ", ".join(ingredientes_reales)
    limite_maximo = obtener_limite_platos()
    prompt_sistema = f"""
    Eres el mesero digital del restaurante 'Fritadas Doña Zita'.
    📋 MENÚ DISPONIBLE:
    {menu_formateado}
    🍳 INGREDIENTES:
    {ingredientes_formateados}
    🛒 ESTADO ACTUAL DEL CARRITO:
    {carrito_actual}

    🚨 REGLAS CRÍTICAS DE LÓGICA Y FORMATO (¡LEER CON ATENCIÓN!):

    1. FORMATO DE MODIFICACIONES (ANTI-CRASH):
       - El campo "modificaciones" SIEMPRE, SIEMPRE debe ser una lista.
       - Si el plato es normal, usa EXACTAMENTE [].
       - Si hay modificaciones, usa OBLIGATORIAMENTE objetos: [{{"tipo": "EXTRA", "ingrediente": "Mote"}}].
       - 🚫 ESTRICTAMENTE PROHIBIDO usar listas de strings como ["EXTRA Mote"]. ¡Solo objetos JSON!
       - Tipos permitidos: "SIN", "EXTRA", "POCO". NUNCA "CON".

    2. MATEMÁTICA DE SUBDIVISIÓN (Pedidos Nuevos):
       - Si pide "N" platos en total, pero "M" tienen cambios, RESTA (N - M).
       - Ejemplo: "3 fritadas, 1 sin mote" -> AGREGAR 2 normales ([]), AGREGAR 1 modificada ([{{"tipo": "SIN", "ingrediente": "Mote"}}]).

    3. ⚠️ INTERCAMBIO vs REPETICIÓN (¡DIFERENCIA VITAL!):
       - REPETICIÓN (CLONAR): Si el cliente quiere OTRO plato IGUAL (ej. "agrega otra igual", "otra con los mismos extras"), NO QUITES NADA. Solo usa AGREGAR y copia las modificaciones del carrito.
       - INTERCAMBIO (MODIFICAR): SOLO si el cliente quiere ALTERAR un plato que ya pidió (ej. "a la fritada que ya tengo, quítale el mote"), debes QUITAR el plato viejo y AGREGAR el nuevo modificado.

    4. ⚠️ PORCIONES SUELTAS (EXTRAS COMO PLATO INDIVIDUAL):
       - Si el cliente pide un ingrediente de forma independiente (ej. "dame un maduro", "una porción de tostado", "y un mote aparte"), NO lo pongas como modificación de un plato.
       - Debes AGREGARLO como un plato nuevo, y su nombre debe empezar SIEMPRE con la frase "Porción de " seguido del ingrediente. (Ej. "plato": "Porción de Plátano Maduro").
```

```
    💡 EJEMPLOS DE RAZONAMIENTO Y JSON:

    EJEMPLO 1 (Subdivisión matemática):
    Cliente: "Necesito tres Fritadas, pero dos sin mote"
    JSON:
    {{
        "razonamiento": "Pide 3 en total. 2 son SIN Mote. 3 - 2 = 1 normal. Acciones: AGREGAR 1 normal,
AGREGAR 2 sin mote.",
        "respuesta_mesero": "¡Entendido! Dos sin mote y una normal.",
        "numero_mesa": 0,
        "acciones": [
            {{ "accion": "AGREGAR", "plato": "Fritada Tradicional", "cantidad": 1, "modificaciones": [] }},
            {{ "accion": "AGREGAR", "plato": "Fritada Tradicional", "cantidad": 2, "modificaciones":
[{{"tipo": "SIN", "ingrediente": "Mote"}}] }}
        ]
    }}
    Tu respuesta debe ser EXCLUSIVAMENTE este formato JSON:
    {{
        "razonamiento": "Tus cálculos paso a paso aquí...",
        "respuesta_mesero": "Frase amable.",
        "numero_mesa": 0,
        "acciones": []
    }}
    Texto del cliente: "{texto_completo}"
    """

    respuesta_llm = ollama.chat(
        model='llama3',
        messages=[{'role': 'user', 'content': prompt_sistema}],
        format='json',
        options={
            'temperature': 0.0 # 🔘 Cero creatividad, 100% obediencia a las reglas
        }
    )
```

Finalmente, para consolidar la tarea técnica T4-4 y asegurar la resiliencia en la transferencia de datos entre subsistemas, la cadena JSON cruda devuelta por el LLM se sometió a una capa estricta de validación y tipado estático utilizando la librería *Pydantic*. Esta frontera de software intercepta la salida sintáctica de la Inteligencia Artificial y corrobora de forma matemática que cumpla con los tipos de datos requeridos por las tablas relacionales (identificadores numéricos, tipos de modificaciones permitidas en los arreglos y coherencia del formato de texto para la

posterior síntesis del habla), evitando fallos en cascada ante una eventual mala sintaxis del modelo generativo. En la Figura 17 se expone la definición estructural de los esquemas orientados a objetos utilizados para garantizar la integridad de las intenciones de compra del comensal.

Figura 17

Modelos de datos estructurados de Pydantic para el tipado y aseguramiento lógico de comandas generadas por IA.

```python
from pydantic import BaseModel, Field
from typing import List, Optional

class ItemPedido(BaseModel):
    plato: str
    cantidad: int
    modificaciones: str = ""
    mods_estructuradas: List[ModificacionStruct] = []
    precio_unitario: float = 0.0 # ⚙️ NUEVO: El precio base de PostgreSQL
    subtotal: float = 0.0 # ⚙️ NUEVO: (precio base + recargos) * cantidad

class OrdenEstructurada(BaseModel):
    respuesta_mesero: str
    numero  mesa: int = 0
    pedidos: List[ItemPedido]
    total_pedido: float = 0.0
    error_stock: str = ""
```

2.7.3. Ejecución del Sprint 3: Interacción Multimodal y Resiliencia

En el tercer Sprint se consolidó la experiencia interactiva dando cumplimiento a la HU-05 (Edición visual) y HU-12 (Manejo de errores). El sistema no solo dependía de la voz, sino que requería una representación visual reactiva frente a las decisiones del comensal. Más allá de simples operaciones aritméticas en el carrito de compras, se desarrollaron *Hooks* de efecto (useEffect) que observan constantemente el estado de la orden para ejecutar validaciones de negocio en tiempo real. Como se observa en la Figura 18, cada vez que el usuario o la IA modifican un ítem, el cliente consulta asíncronamente al servidor para calcular la demora, verificar la disponibilidad física en bodega y asegurar que no se excedan los límites comerciales parametrizados, bloqueando la transacción si el stock resulta insuficiente.

Figura 18

Implementación de Hooks reactivos para la validación de reglas de negocio y control de stock en tiempo real.

```
// 1. Obtener parámetros comerciales del restaurante
 useEffect(() => {
  fetch('http://127.0.0.1:8000/configuracion-kiosco')
    .then(res => res.json())
    .then(data => setLimitePlatos(data.max_platos))
    .catch(err => console.error("Usando límite por defecto", err));
 }, []);

 // 2. Cálculo matemático de límites comerciales
 const totalPlatosPedido = carrito.reduce((acumulador, item) => acumulador + (parseInt(item.cantidad) ||
1), 0);
 const excedeLimite = totalPlatosPedido > limitePlatos;

 // 3. Efectos colaterales (Side-effects) reactivos a la mutación del carrito
 useEffect(() => {
  if (carrito.length > 0) {
    // Petición original para estimar el tiempo
    fetch('http://127.0.0.1:8000/estimar-tiempo', {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(carrito)
    })
      .then(res => res.json())
      .then(data => setTiempoEstimado(data.tiempo_estimado_minutos))
      .catch(err => console.error("Error estimando tiempo:", err));

    // Validación de stock en tiempo real
    fetch('http://127.0.0.1:8000/validar-carrito', {
```

```
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(carrito)
    })
      .then(res => res.json())
      .then(data => {
        if (!data.valido) {
          setErrorStock(`Stock insuficiente de ${data.ingrediente}. Quedan ${data.stock} en bodega.`);
        } else {
          setErrorStock("");
        }
      })
      .catch(err => console.error("Error validando stock:", err));

  } else {
    setTiempoEstimado(null);
    setErrorStock("");
  }
}, [carrito]);
```

Adicionalmente, se priorizó la resiliencia del sistema implementando barreras de contingencia frente a posibles fallas en el procesamiento de IA o latencias de red. Se establecieron *Timeouts* en las peticiones HTTP y se programaron flujos alternativos (Fronteras de Error) que permiten al sistema transicionar automáticamente hacia una interfaz de selección táctil tradicional si el motor *Speech-to-Text* no responde en el tiempo esperado. La lógica de captura de estas excepciones se ilustra en la Figura 19.

Figura 19

Manejo de excepciones y caídas del modelo de IA en la capa del cliente.

```
const iniciarGrabacion = async () => {
  setErrorStock(""); // Limpiamos el error viejo
  try {
    const stream = await navigator.mediaDevices.getUserMedia({ audio: true })
    mediaRecorderRef.current = new MediaRecorder(stream)
    audioChunksRef.current = []

    mediaRecorderRef.current.ondataavailable = (event) => {
      if (event.data.size > 0) audioChunksRef.current.push(event.data)
    }

    mediaRecorderRef.current.onstop = async () => {
      if (audioChunksRef.current.length === 0) {
        setGrabando(false); return;
      }
```

```
      const audioBlob = new Blob(audioChunksRef.current, { type: 'audio/webm' })
      const formData = new FormData()
      formData.append("audio", audioBlob, "pedido.webm")
      formData.append("carrito_actual", JSON.stringify({ numero_mesa: numeroMesa, pedidos: carrito }))

      try {
        // Bloque de contingencia ante caídas del servidor local LLM
        const respuesta = await fetch("http://127.0.0.1:8000/pedido-voz", { method: "POST", body:
formData })
        if (respuesta.ok) {
          const resultado = await respuesta.json();
          if (resultado.transcripcion) setTranscripcion(resultado.transcripcion);

          if (resultado.orden) {
            if (resultado.orden.error_stock) {
              setErrorStock(resultado.orden.error_stock);
            }
            if (resultado.orden.pedidos) setCarrito(resultado.orden.pedidos);
            if (resultado.orden.numero_mesa !== undefined && resultado.orden.numero_mesa !== 0) {
              setNumeroMesa(resultado.orden.numero_mesa);
            }
          }
          if (resultado.audio_b64) {
            const audioMesero = new Audio(`data:audio/wav;base64,${resultado.audio_b64}`);
            audioMesero.play();
          }
        }
      } catch (error) {
        console.error("Error audio:", error)
      }
    }

    mediaRecorderRef.current.start()
    setGrabando(true)
  } catch (error) {
    alert("Permite el acceso al micrófono.")
  }
}
```

2.7.4. Ejecución del Sprint 4: Orquestación y Monitor de Producción

En esta fase se abordó la integración del flujo de pedidos con el área de preparación, abarcando las historias HU-07 (Orquestación) y HU-08 (Monitor Cocina). Una vez que el comensal confirma el carrito en el *Frontend*, la orden no se envía ciegamente a producción. En su lugar, el sistema implementa una capa de seguridad donde el *payload* es enviado primero al *Backend* (FastAPI) a través del endpoint /confirmar-orden.

En esta capa intermedia, el sistema ejecuta una validación crítica de inventario en tiempo real. Si la orden es válida, el *Backend* enriquece el JSON con el recálculo financiero total y dispara internamente un *Webhook* hacia la plataforma de orquestación n8n. La lógica de esta barrera de seguridad y la emisión del evento hacia el orquestador se evidencia en la Figura 20.

Figura 20

Lógica de backend para validación de stock y emisión del Webhook hacia n8n.

```
from fastapi import APIRouter
import requests
from app.services.ia_service import validar_stock_carrito # ⚙️ Importamos al Guardia de Seguridad

router = APIRouter()

@router.post("/confirmar-orden")
def confirmar_orden(orden: dict):
    print("🚀 Recibiendo orden final y validando stock antes de n8n...")

    # 1. 🛡️ ÚLTIMA VALIDACIÓN ANTES DE ENVIAR A COCINA
    carrito_list = orden.get("pedidos", [])
    validacion = validar_stock_carrito(carrito_list)

    if not validacion["valido"]:
        # Si alguien compró el último ingrediente justo antes de darle a confirmar
        return {"exito": False, "error": f"Stock insuficiente de {validacion['ingrediente']}"}

    # 2. 📝 PREPARAMOS EL PAQUETE PARA N8N
    # Le inyectamos el total en dólares y el reporte de consumo a la orden original
    total_dolares = sum(item.get("subtotal", 0.0) for item in carrito_list)
    orden["total_pedido"] = total_dolares
    orden["consumo_ingredientes"] = validacion.get("consumo", {})

    # 3. 🚀 ENVIAMOS A N8N
    webhook_url = "http://localhost:5678/webhook/orden-zita"

    try:
```

```
    # Mandamos el JSON ya enriquecido con precios y consumo
    respuesta = requests.post(webhook_url, json=orden)

    if respuesta.status_code == 200:
        print("☑ ¡Orden entregada a n8n con éxito!")
        return {"exito": True, "mensaje": "Orden orquestada correctamente"}
    else:
        print(f"⚠ n8n respondió con error: {respuesta.status_code}")
        return {"exito": False, "error": "n8n rechazó la orden", "status": respuesta.status_code}

except Exception as e:
    print(f"❌ Error al conectar con n8n: {e}")
    return {"exito": False, "error": "Fallo de conexión con el orquestador"}
```

Una vez que n8n captura este *Webhook*, el orquestador asume la responsabilidad de la transaccionalidad: ejecuta un nodo de PostgreSQL para insertar la cabecera del pedido, obtiene el identificador maestro y utiliza un nodo *Split Out* para iterar e insertar los detalles individuales.

Para garantizar que el área de producción reciba esta orden recién orquestada sin necesidad de refrescar la pantalla manualmente, se implementó una arquitectura de sincronización asíncrona basada en *Short Polling*. Esta técnica ejecuta peticiones de bajo costo computacional hacia la API de la cocina en intervalos controlados de 4 segundos, actualizando la memoria del cliente en tiempo real. La implementación de esta sincronización reactiva se evidencia en la Figura 21.

Figura 21

Implementación de sincronización Short Polling para el refresco asíncrono del Monitor de Cocina.

```
const cargarComandas = () => {
   fetch('http://127.0.0.1:8000/cocina/ordenes')
     .then(res => res.json())
     .then(datos => {
       const agrupadas = {}

       datos.ordenes.forEach(item => {
         if (!agrupadas[item.id_pedido]) {
           agrupadas[item.id_pedido] = {
             id_pedido: item.id_pedido,
             id_mesa: item.id_mesa,
             cliente_nombre: item.cliente_nombre,
             fecha_inicio_global: null,
             tiempo_max_asignado: 0,
```

```
              solo_bebidas: true,
              items: []
            }
          }

          // Si el backend dice que es comida, cambiamos la etiqueta
          if (item.requiere_coccion === true) {
            agrupadas[item.id_pedido].solo_bebidas = false;
          }

          // Leemos el tiempo exacto que la base de datos selló. No sumamos
nada.
          const tiempoDB = parseInt(item.tiempo_asignado_cocina) || 0;
          if (tiempoDB > agrupadas[item.id_pedido].tiempo_max_asignado) {
            agrupadas[item.id_pedido].tiempo_max_asignado = tiempoDB;
          }

          if (item.fecha_inicio_preparacion &&
!agrupadas[item.id_pedido].fecha_inicio_global) {
            agrupadas[item.id_pedido].fecha_inicio_global =
item.fecha_inicio_preparacion;
          }

          agrupadas[item.id_pedido].items.push(item)
        })

        setComandas(Object.values(agrupadas).sort((a, b) => a.id_pedido -
b.id_pedido))
        setCargando(false)
      })
      .catch(() => setCargando(false))
  }

  useEffect(() => {
    cargarComandas()
    const intv = setInterval(cargarComandas, 4000)
    return () => clearInterval(intv)
  }, [])
```

Paralelamente, se construyó el panel interactivo tipo *Kanban* en React para el personal de cocina. Esta interfaz renderiza tarjetas dinámicas que agrupan los platos por mesa y los prioriza bajo la política estricta FIFO (*First-In, First-Out*), bloqueando visualmente las comandas nuevas hasta que se resuelvan las más antiguas. Cada elemento dentro de la tarjeta posee botones táctiles que permiten la mutación del ciclo de vida del pedido ("Preparando", "Listo").

En la Figura 22 se detalla la función encargada de capturar la interacción del cocinero y despachar la mutación hacia el *Backend* para registrar la hora exacta de entrega.

Figura 22

Lógica transaccional para la mutación de estados de producción en el componente Kanban.

```
const manejarAccionItem = async (id_detalle, accion) => {
  try {
    if (accion === 'LISTO') {
      // Hacemos la petición al backend
      const respuesta = await fetch(`http://127.0.0.1:8000/cocina/entregar/${id_detalle}`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' }
      });

      // Verificamos si el backend arrojó un error (Ej: Error 500)
      if (!respuesta.ok) {
        const errorDatos = await respuesta.json();
        alert(`❌ Error del servidor: ${errorDatos.detail || 'Fallo desconocido'}`);
        return; // Cortamos la ejecución para no recargar datos falsos
      }
    } else {
      alert(`Acción administrativa [${accion}] registrada para el ítem ${id_detalle}`);
    }

    // Forzamos la actualización inmediata de la pantalla
    cargarComandas();

  } catch (error) {
    // Capturamos si el backend está apagado o hay un problema de CORS
    console.error("Error de conexión:", error);
    alert("❌ No se pudo conectar con el servidor. Revisa si FastAPI está encendido.");
  }
}
```

2.7.5. Ejecución del Sprint 5: Gestión de Colas y Tiempos de Espera

Durante la quinta iteración, planificada con una duración de dos semanas, se ejecutó el desarrollo de las historias de usuario HU-06 (Estimación Tiempos) y HU-09 (Actualización estado), abordando de forma secuencial las tareas técnicas T6-1 a T6-4 y T9-1 a T9-3. El núcleo de este Sprint consistió en eliminar la opacidad operativa del restaurante mediante un sistema predictivo capaz de calcular la demora exacta de la cocina en tiempo real.

Para dar cumplimiento a la tarea T6-2, orientada al diseño de la consulta de agregación en la base de datos, se programó una sentencia SQL adaptativa dentro de la capa de servicios del

backend. A diferencia de un conteo estático de comandas, esta consulta discrimina el estado transaccional de cada platillo; si el ítem se encuentra en estado 'SOLICITADO', computa el tiempo teórico completo, mientras que si su estado es 'PREPARANDO', calcula dinámicamente el tiempo remanente exacto restando los minutos que ya han transcurrido desde su inicio de cocción mediante funciones de conversión de marcas temporales (EXTRACT(EPOCH FROM...)). La arquitectura de esta consulta SQL adaptativa y de su lógica de persistencia se detalla formalmente en la Figura 23.

Figura 23

Sentencia SQL dinámica con cálculo de tiempo remanente mediante sustracción de épocas en PostgreSQL.

```
consulta_cola_dinamica = """
        SELECT COALESCE(SUM(
           CASE
             -- Si apenas fue solicitado, suma el tiempo completo
              WHEN dp.estado_item = 'SOLICITADO' THEN (p.tiempo_prep_min                *
dp.cantidad)

              -- Si se está preparando, restamos los minutos que ya pasaron
              WHEN dp.estado  item = 'PREPARANDO' AND dp.fecha  inicio  preparacion IS NOT NULL
THEN
                 GREATEST(0, (p.tiempo_prep_min * dp.cantidad) - (EXTRACT(EPOCH FROM
(CURRENT_TIMESTAMP - dp.fecha_inicio_preparacion)) / 60))

              -- Por si acaso falta la fecha, cobramos el tiempo completo
              ELSE (p.tiempo_prep_min * dp.cantidad)
           END
        ), 0)
        FROM Detalle  Pedido dp
        JOIN Plato p ON dp.id_plato = p.id_plato
        WHERE dp.estado  item IN ('SOLICITADO', 'PREPARANDO')
    """
```

Posteriormente, para instrumentar las tareas T6-1, T6-3 y T9-1, se procedió al desarrollo del endpoint regulador en el entorno de *FastAPI*. En esta capa lógica se procesa la carga bruta de la cocina obtenida de la persistencia y se contrasta frente a los parámetros operativos dinámicos del establecimiento (tales como la capacidad física de la paila y la cantidad de cocineros concurrentes). El algoritmo mapea el total de platillos en un modelo de procesamiento por lotes distribuidos, calculando el tiempo requerido para completar tandas llenas e incorporando una sobrecarga marginal por cada platillo adicional que sature el lote caliente, dividiendo

finalmente el esfuerzo computado entre los servidores físicos disponibles. En la Figura 24 se expone la lógica matemática y la estructura del controlador encargado de despachar este cálculo probabilístico en formato JSON.

Figura 24

Implementación en FastAPI del algoritmo de procesamiento concurrente por lotes con sobrecarga marginal.

```
# Funciones Matemáticas

    # 1. Calculamos el tiempo de una tanda completamente llena (ej. 8 platos)
    tiempo_tanda_llena = tiempo_base + (tiempo_base * porc_extra * (capacidad_paila - 1))

    # 2. Vemos cuántas tandas completas y cuántos platos sobrantes hay
    tandas_completas = total_platos // capacidad_paila
    platos  sobrantes = total  platos % capacidad  paila

    # 3. Calculamos el tiempo exacto de la tanda parcial
    tiempo_tanda_parcial = 0
    if platos  sobrantes > 0:
       tiempo  tanda  parcial = tiempo  base + (tiempo  base * porc  extra * (platos  sobrantes - 1))

    # 4. Sumamos la carga de trabajo total como si hubiera 1 solo súper-cocinero
    tiempo  total  1  cocinero = (tandas  completas * tiempo  tanda  llena) + tiempo  tanda  parcial

    # 5. Dividimos entre los cocineros reales.
    # Esto genera una línea de crecimiento constante (9 platos = 20 min, 10 platos = 21 min, etc.)
    tiempo  espera  base = int(tiempo  total  1  cocinero / cocineros)

    return {"tiempo_estimado_minutos": tiempo_espera_base}
```

Finalmente, para consolidar las tareas T6-4, T9-2 y T9-3, los resultados numéricos de este pipeline distributivo fueron integrados directamente en la interfaz del comensal (*React*). El sistema reactivo vincula el resultado del endpoint al estado global del Kiosco, renderizando de manera síncrona los minutos de demora proyectados antes de que el usuario finalice la transacción, lo cual optimiza la experiencia del consumidor. Asimismo, la capa de aplicación se conectó con el despachador asíncrono del servidor para recibir alertas en tiempo real, de modo que cuando el monitor Kanban de la cocina muta el estado de un platillo a "ENTREGADO", la interfaz del cliente despliega de inmediato notificaciones emergentes (*Toasts*) acompañadas de estímulos acústicos discretos, garantizando un flujo informativo bidireccional, automatizado y libre de latencia manual.

2.7.6. Ejecución del Sprint 6: Analítica Administrativa e Insights

El Sprint final dotó al ecosistema de capacidades de Inteligencia de Negocios (BI), integrando las historias HU-10 (Reportes Admin) y HU-11 (Recomendaciones IA). Se desarrollaron consultas de agregación SQL avanzadas, utilizando funciones de agrupamiento temporal (EXTRACT) para analizar el historial financiero y operativo del restaurante, aislando únicamente los pedidos con el estado transaccional cerrado ("PAGADO"). En la Figura 25 se aprecia la consulta dinámica empleada para la generación de la curva de evolución de ingresos basada en la estacionalidad del negocio.

Figura 25

Consulta de agregación temporal en PostgreSQL para el análisis de estacionalidad.

```
@router.get("/reportes/evolucion")
def obtener_evolucion_ventas(periodo: str = 'semana', db=Depends(get_db)):
    cursor = db.cursor(cursor_factory=RealDictCursor)

    # 1. Filtro y agrupación dinámica según el periodo
    if periodo == 'hoy':
        filtro_fecha = "DATE(fecha_apertura) = CURRENT_DATE"
        agrupacion = "EXTRACT(HOUR FROM fecha_apertura)"
    elif periodo == 'semana':
        filtro_fecha = "fecha_apertura >= date_trunc('week', CURRENT_DATE)"
        agrupacion = "EXTRACT(ISODOW FROM fecha_apertura)"
    else: # mes
        filtro_fecha = "fecha_apertura >= date_trunc('month', CURRENT_DATE)"
        # agrupamos por el DÍA del mes (1 al 31)
        agrupacion = "EXTRACT(DAY FROM fecha_apertura)"

    query = f"""
        SELECT
            {agrupacion} AS bloque_tiempo,
            COALESCE(SUM(total_final), 0) AS ingresos
        FROM pedido
        WHERE {filtro_fecha} AND estado_pago = 'PAGADO'
        GROUP BY bloque_tiempo
        ORDER BY bloque_tiempo
    """
```

Finalmente, para trascender la simple visualización de gráficos, se orquestó un Consultor Operativo automatizado. El *Backend* recopila los KPIs financieros, el rendimiento del menú y las métricas de saturación horaria de la base de datos, y los inyecta en el LLM (*Llama 3*) mediante un *prompt* analítico meticulosamente diseñado. Como se muestra en la Figura 26,

esta instrucción obliga al modelo a redactar directrices gerenciales estructuradas, identificando cuellos de botella y sugiriendo acciones de *up-selling* sin requerir intervención humana.

Figura 26

Prompt de inyección de métricas para la generación autónoma de Inteligencia de Negocios (LLM).

```
prompt = f"""
  Eres un Consultor Gastronómico Senior y Analista de Datos analizando el restaurante "Doña Zita".
  Analiza este reporte del periodo: {periodo}.

  DATOS REALES EXTRAÍDOS DE POSTGRESQL:
  - Ingresos Brutos: ${ingresos}
  - Volumen de Órdenes: {ordenes}
  - Ticket Promedio: ${ticket_promedio}

  - Top Platos Más Vendidos:
  {platos_str}

  - Datos de Operación en Cocina:
  Hora Pico de Demanda: {hora_pico}
  Tiempo Máximo de Espera Promedio: {max_espera} minutos.

  REGLAS ESTRICTAS PARA TU RESPUESTA:
  1. Cero saludos o introducciones genéricas. Ve directo al grano.
  2. Escribe exclusivamente en formato Markdown (usa **negritas** para resaltar métricas clave).
  3. Tu respuesta debe tener EXACTAMENTE estas tres secciones con sus respectivos emojis:

  ### 📈 Rendimiento Financiero
  (Analiza si el ticket promedio y volumen son saludables).

  ### 🍽️ Análisis del Menú
  (Qué decisión de inventario o promoción tomar respecto a los platos top).

  ### ⚠️ Cuello de Botella Operativo
  (Critica la relación entre la hora pico y el tiempo de espera máximo. Si la espera supera los 20 minutos,
da una alerta crítica y sugiere una acción operativa inmediata en cocina).
  """

  try:
     respuesta  llm = ollama.chat(
        model='llama3',
        messages=[{'role': 'user', 'content': prompt}],
        options={'temperature': 0.4}
```

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

Una vez concluido el proceso de análisis y desarrollo del proyecto, en este capítulo se presentan los resultados obtenidos en cada una de las fases de la ejecución de la investigación, los cuales evidencian el logro de los objetivos planteados. Es la sección donde presenta de forma clara y objetiva qué se logró con el proyecto. Además, es el corazón del trabajo: aquí se muestran las respuestas a las preguntas de investigación y se validan los objetivos de investigación planteados inicialmente.

Para verificar el cumplimiento de los requisitos funcionales y no funcionales, la presentación de los resultados se ha estructurado dividiendo las funcionalidades por tipo de rol dentro del ecosistema del restaurante. A continuación, se detalla el funcionamiento práctico de cada módulo interactivo respaldado por capturas de pantalla del sistema final.

3.1. FASE IV: Despliegue, Validación Operativa y Análisis de Resultados

3.1.1. Funcionalidades del rol cliente (comensal): kiosco y pantalla de turnos

El módulo de interacción con el cliente fue diseñado para reemplazar la toma de pedidos analógica por un proceso ágil y multimodal.

3.1.1.1 .Inicialización y bienvenida

El sistema opera en un entorno restringido mediante la configuración de "Modo Kiosco", permitiendo el acceso únicamente a la aplicación. La interacción inicial comienza con la definición de la modalidad de consumo. En la Figura 27 se presenta la nueva pantalla de inicio, donde el comensal debe indicar si su pedido será para "Comer Aquí" o "Para Llevar", un paso fundamental para agilizar el flujo de atención en cocina y determinar el tipo de empaque.

Figura 27

Pantalla Inicial de Selección de Modalidad de Consumo.

Dependiendo de la opción seleccionada, el sistema adapta el siguiente paso del proceso. Si el usuario elige consumir en el establecimiento, se despliega la pantalla de selección de paleta (Figura 28), donde debe escoger un número disponible que corresponde al identificador físico de su mesa para el posterior servicio.

Figura 28

Pantalla de Selección de Número de Paleta.

Por otra parte, si el cliente opta por un pedido para llevar, el sistema lo dirige a una sección destinada al registro de su nombre (Figura 29), el cual puede ingresarse mediante texto o por voz. A través de técnicas de reconocimiento y procesamiento de lenguaje natural, el usuario interactúa con una anfitriona digital que asiste en el proceso y confirma la información proporcionada antes de dar paso al menú interactivo.

Figura 29

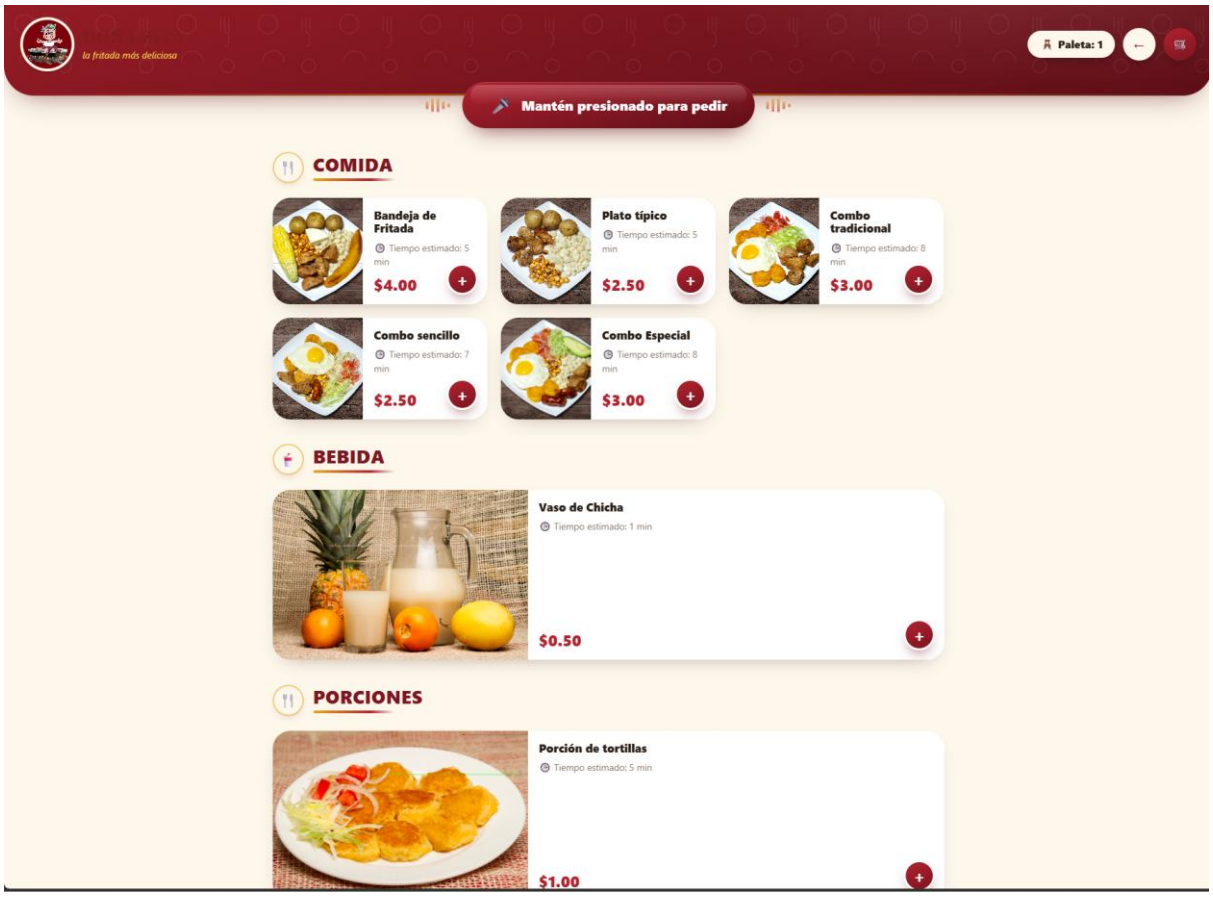
Interacción conversacional con la Anfitriona Digital (IA) para registro de nombre.

3.1.1.2. Catálogo y captura de pedido por voz (STT)

Una vez autenticada la sesión del usuario, se despliega el menú organizado por categorías (Figura 30).

Figura 30

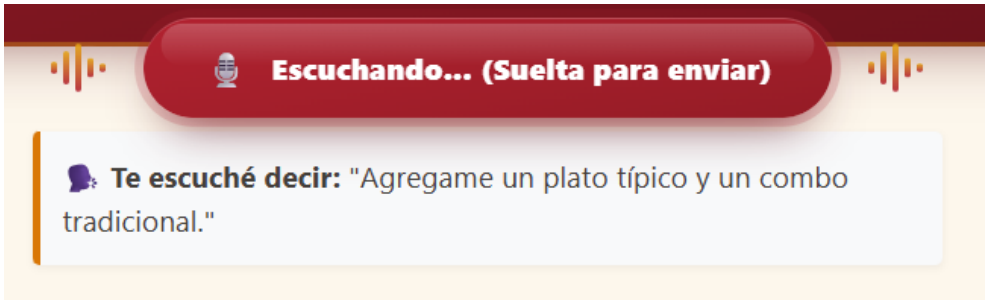
Interfaz del Catálogo de Menú Principal.



El usuario cuenta con un botón ubicado en la parte superior de la pagina que, al mantenerse presionado, activa el motor acústico *Faster-Whisper*. La Figura 31 evidencia la transcripción de voz a texto en tiempo real.

Figura 31

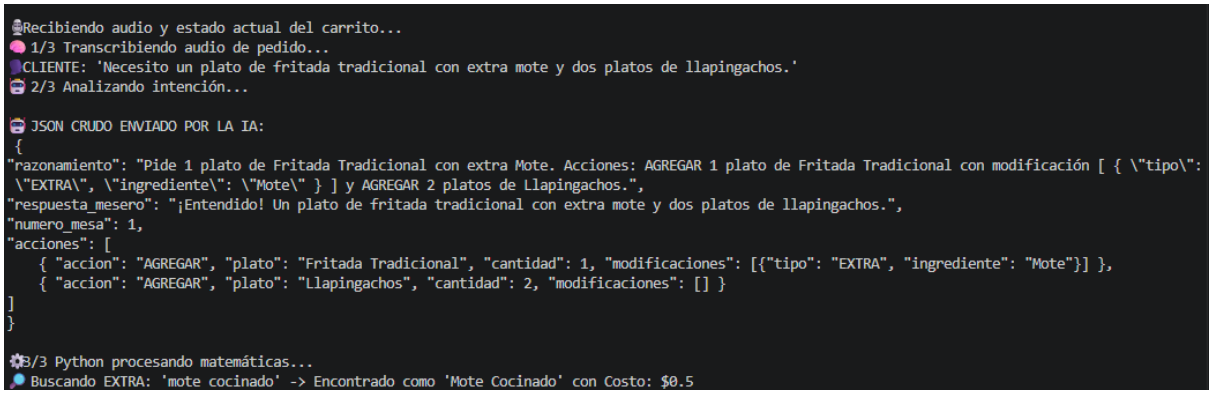
Captura y transcripción de la orden en tiempo real (STT).



Internamente, el texto capturado es enviado al modelo LLM (Ollama), el cual estructura la información y extrae las intenciones del usuario. La Figura 32 muestra el JSON estructurado devuelto por la IA, confirmando la precisión del modelo cognitivo.

Figura 32

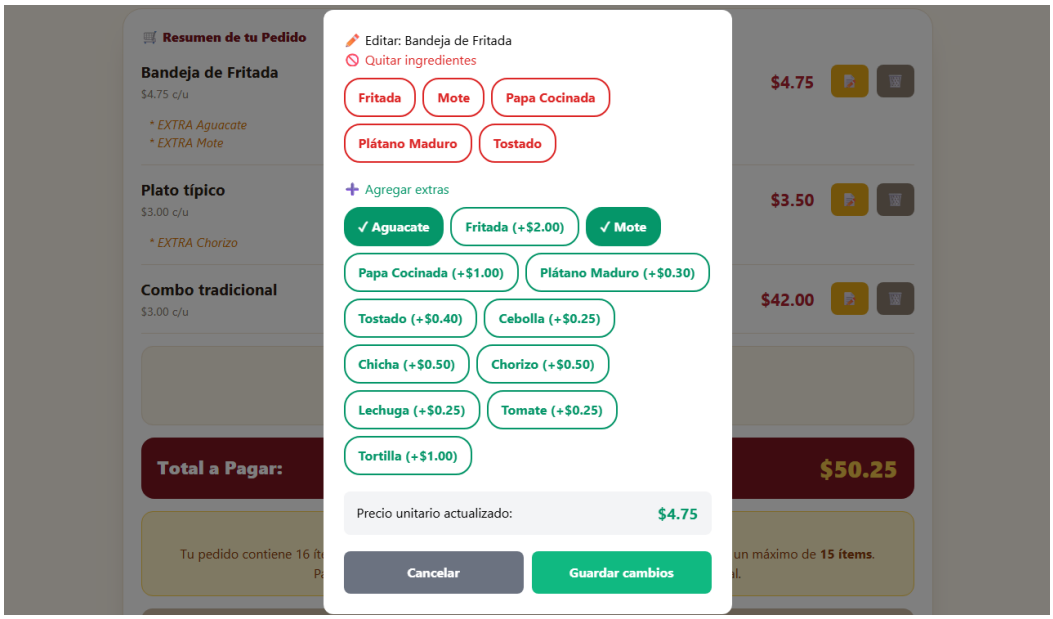
JSON estructurado generado por el motor de IA local.



De manera complementaria a la captura de pedidos por voz, el sistema proporciona una interfaz táctil para la selección y personalización manual de los productos. Al interactuar con un platillo específico del catálogo, se despliega una ventana emergente (modal) que permite al usuario detallar su orden. Como se ilustra en la Figura 33 esta vista facilita la modificación de ingredientes (exclusiones o adiciones) y el ajuste de cantidades antes de confirmar la adición del platillo al carrito de compras.

Figura 33

Ventana emergente para la personalización y modificación de platillos.

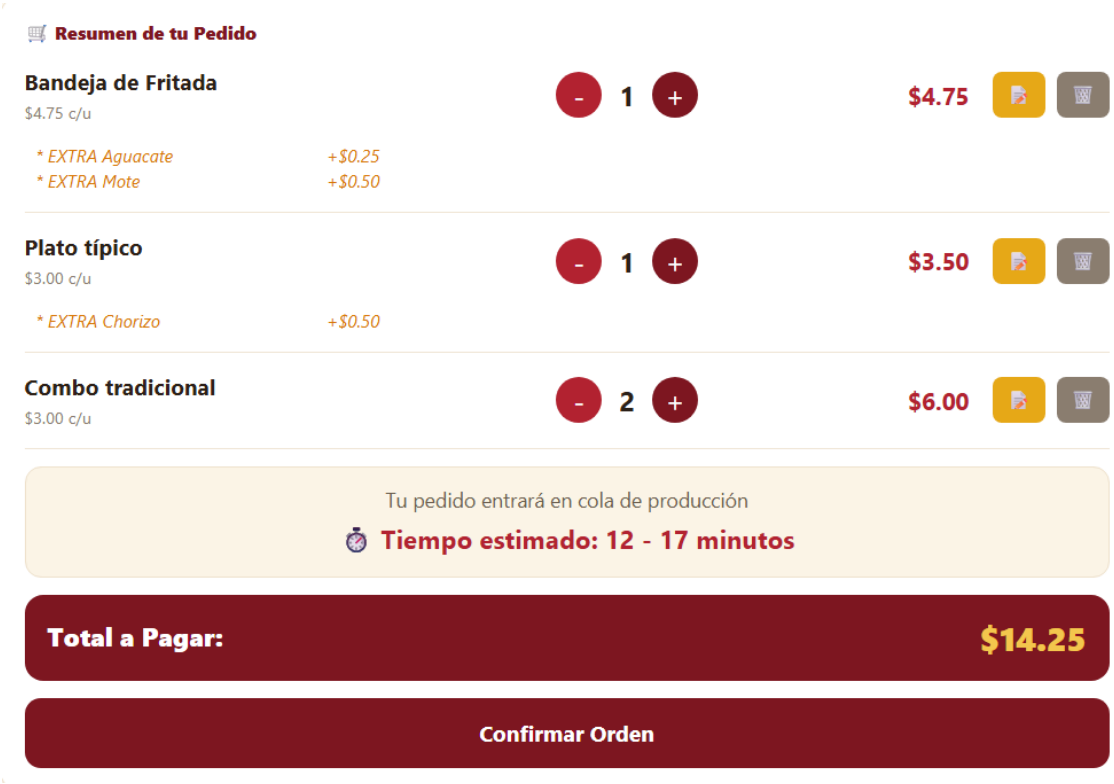


3.1.1.3. Resumen de carrito, validación de stock y tiempos de espera

Tras el procesamiento semántico, el kiosco presenta al comensal el resumen estructurado del carrito (Figura 34), reflejando las modificaciones (ej. "Sin cebolla").

Figura 34

Resumen del Carrito con modificaciones extraídas por IA.



El sistema evalúa en tiempo real las reglas de negocio, bloqueando la transacción si se supera el límite de platillos por orden (Figura 35) o si el inventario calculado resulta insuficiente (Figura 36).

Figura 35

Alerta de restricción por límite comercial de platillos.

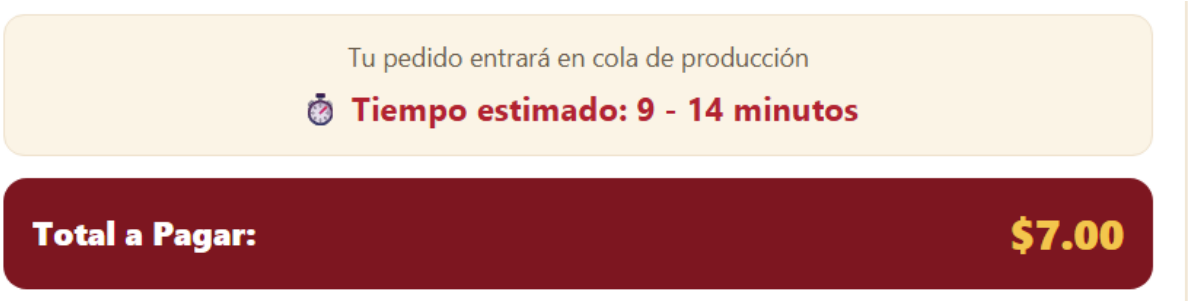
Figura 36

Alerta reactiva de stock insuficiente.

Además, aplicando la teoría de colas, se visualiza la estimación dinámica del tiempo de espera antes de que el usuario envíe la orden (Figura 37).

Figura 37

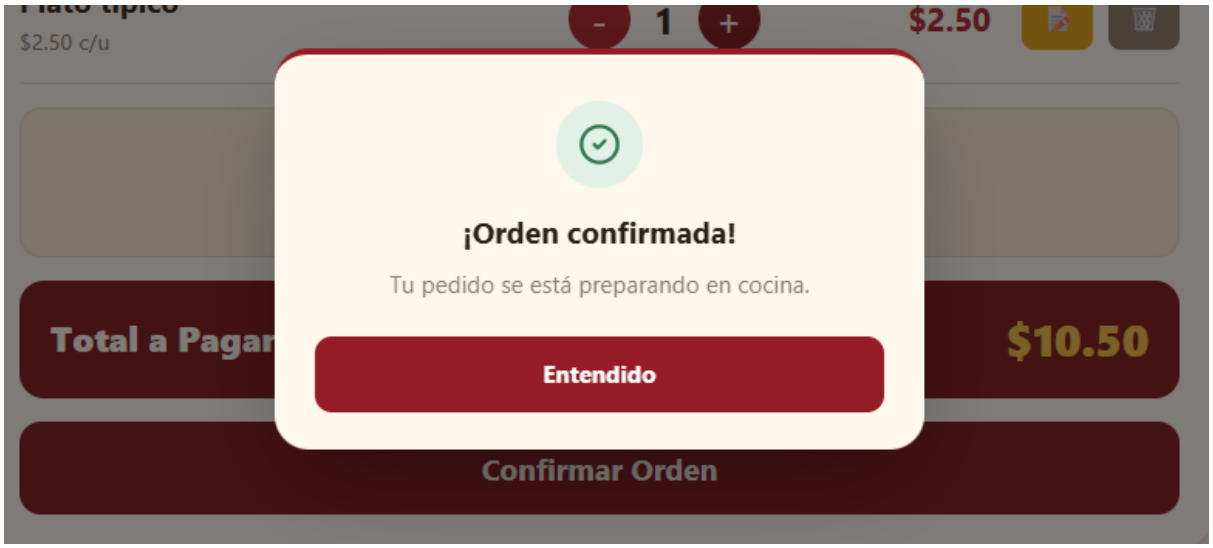
Cálculo y visualización de demora estimada (Teoría de Colas).



Al confirmar, la plataforma *n8n* orquesta la petición y el kiosco muestra un mensaje de éxito (Figura 38)

Figura 38

Confirmación de éxito y envío asíncrono vía Webhook.



3.1.1.4. Visualización de estados (pantalla de turnos)

Para mitigar la incertidumbre del cliente, el sistema provee una pantalla pública de visualización. En la Figura 39 se muestra la columna "En Cola".

Figura 39

Pantalla de Turnos: Pedidos "En Cola".

Cuando el pedido ingresa a preparación, se traslada a la columna central, la cual renderiza una barra de progreso basada en el tiempo asignado (Figura 40).

Figura 40

Pantalla de Turnos: Barra de progreso de preparación.

3.1.2. Funcionalidades del rol cocinero: monitor de producción (kanban)

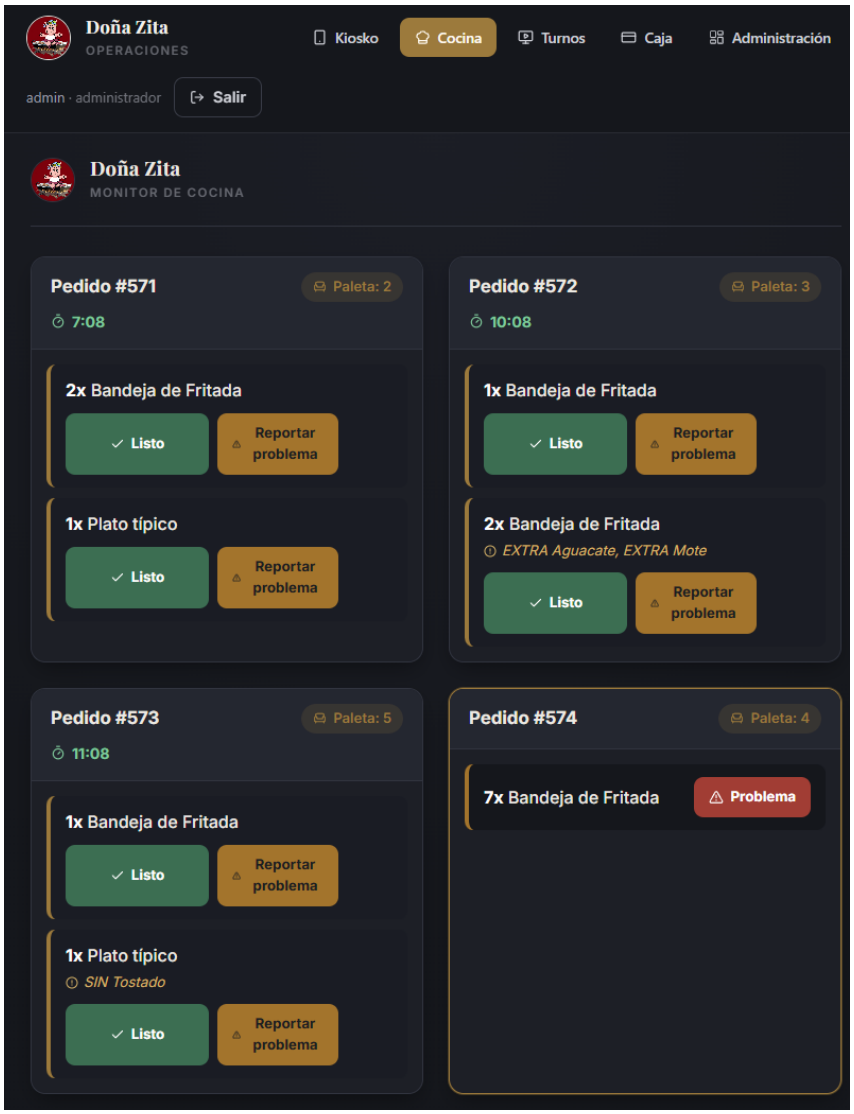
El módulo de cocina permite al área de producción visualizar y gestionar las comandas en tiempo real, eliminando el uso de tickets de papel impresos.

3.1.2.1. Gestión de comandas y tiempos

En la Figura 41 se visualiza la interfaz principal (Kanban) de la cocina, donde las comandas ingresan ordenadas mediante el principio FIFO (First-In, First-Out).

Figura 41

Monitor de Cocina: Vista General de Comandas.



Cada tarjeta destaca especificaciones extraídas por la IA (ej. "Sin tostado") y contiene un temporizador regresivo que cambia a color rojo si el plato excede el tiempo estimado, donde

el cocinero interactúa mediante botones táctiles para avanzar el flujo del platillo a "Completado" (Figura 42).

Figura 42

Tarjeta de pedido individual con temporizador regresivo y estado "PREPARANDO".

Para complementar el flujo de preparación, la interfaz incluye una funcionalidad crítica para el manejo de excepciones operativas. En caso de presentarse algún inconveniente con una orden, el personal de cocina tiene la opción de reportar problemas directamente desde la tarjeta del pedido. Como se muestra en la Figura 43, el sistema permite clasificar la incidencia en tres categorías principales: error humano durante la preparación, falta repentina de stock de algún ingrediente, o cancelación directa por parte del cliente. Esta acción detiene el temporizador y alerta inmediatamente al sistema para su respectiva gestión administrativa y actualización del estado de la orden.

Figura 43

Menú de opciones para el reporte de incidencias y cancelación de pedidos en cocina.

Al marcar todos los ítems como listos, el sistema actualiza la Pantalla de Turnos del cliente mediante WebSockets, trasladando la orden a la sección "¡Listo para Retirar!" (Figura 44).

Figura 44

Pantalla de Turnos: Notificación asíncrona de "Pedido Listo".

3.1.3. Funcionalidades del rol cajero: módulo de caja y facturación

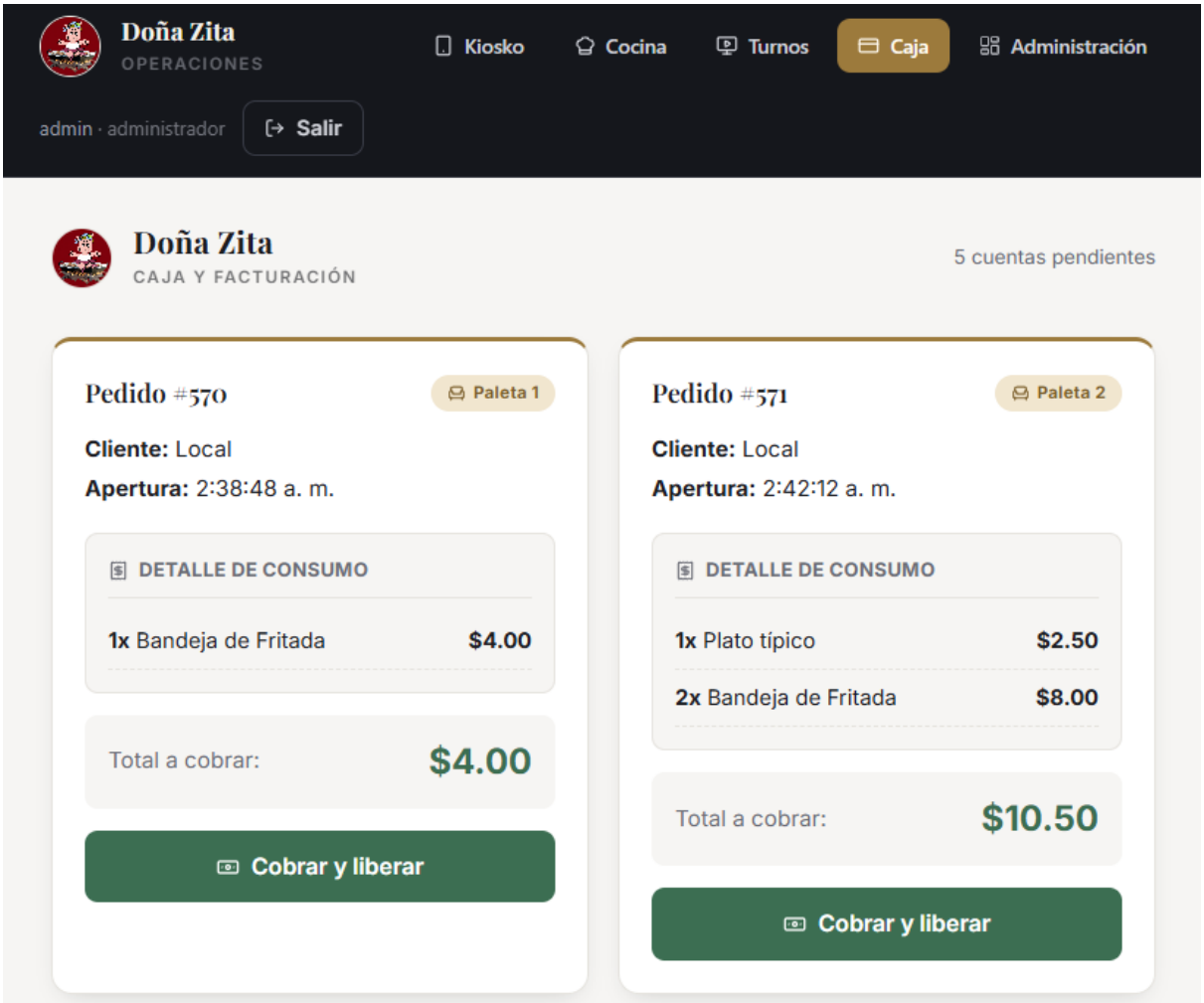
El sistema desvincula el proceso de cobro de la toma de pedidos, agilizando el flujo de atención al delegar el pago a una terminal independiente.

3.1.3.1. Cobro y liberación de mesas

En la Figura 45 se observa cómo, el cajero accede a una cuadrícula con todas las cuentas en estado "PENDIENTE".

Figura 45

Módulo de Caja: Cuadrícula de cuentas por cobrar.



Cada tarjeta expone el detalle exacto de consumo, el desglose de precios y el total a pagar (Figura 46). Tras confirmar la recepción del dinero, el sistema ejecuta la liberación de la mesa o paleta en la base de datos PostgreSQL, cerrando el ciclo transaccional (Figura 47).

Figura 46

Detalle de consumo y total monetario de una cuenta pendiente.

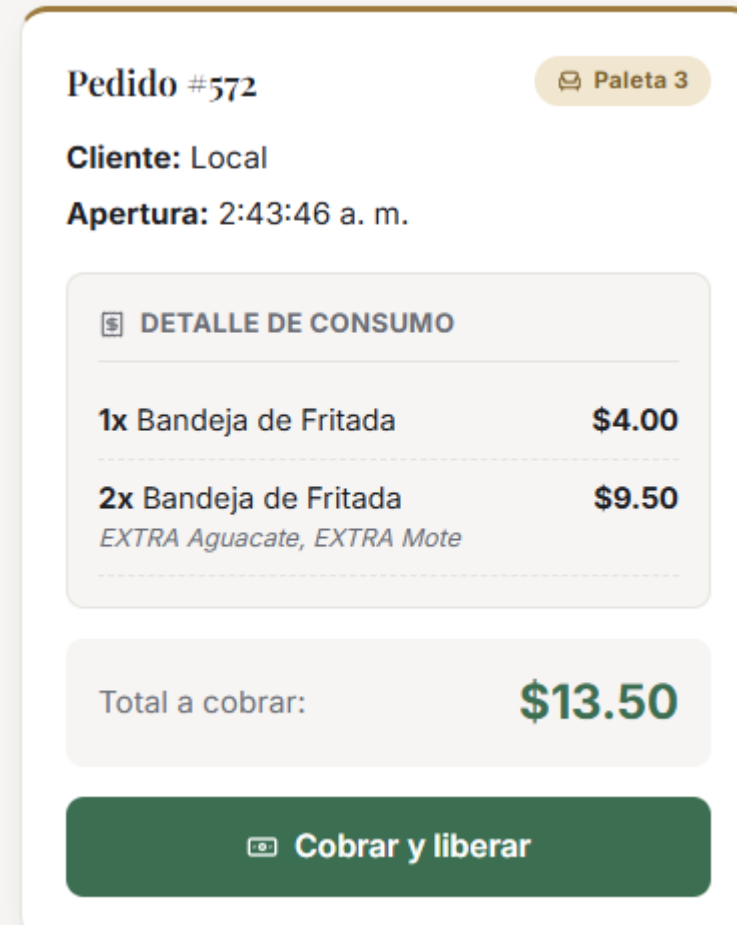


Figura 47

Confirmación de pago y liberación de mesa en base de datos.

3.1.4. Funcionalidades del rol administrador: dashboard analítico e insights de ia

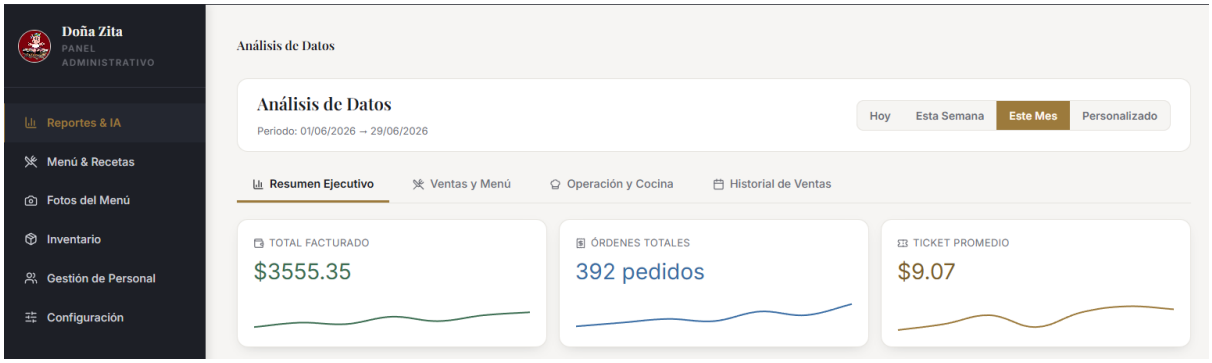
En esta sección se presentan reportes, donde los datos almacenados se transforman en inteligencia de negocios, garantizando que el cumplimiento de los resultados presentados sea verificable a través de cifras exactas y no aproximaciones vagas.

3.1.4.1. Resumen ejecutivo y evolución de ventas

El sistema presenta los resultados principales en la pestaña "Resumen Ejecutivo". La Figura 48 muestra los KPIs fundamentales (Ingresos Totales, Órdenes Totales y Ticket Promedio) obtenidos dinámicamente según el filtro temporal.

Figura 48

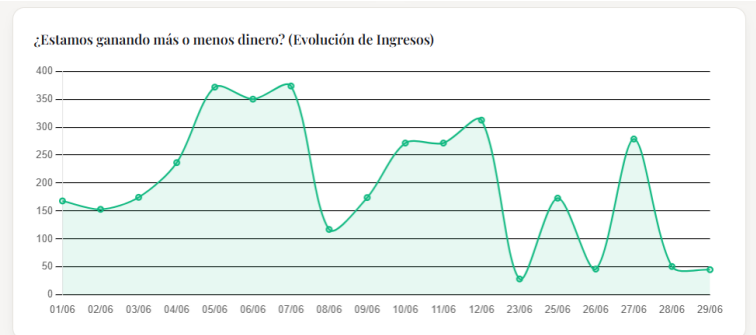
Dashboard Admin: Indicadores Clave de Rendimiento (KPIs).



El análisis detallado de los hallazgos continúa con la curva de crecimiento financiero graficada en la Figura 49, la cual agrupa las ventas por horas, días o semanas, permitiendo evaluar la estacionalidad del negocio mediante gráficos para comparaciones y tendencias.

Figura 49

Gráfico lineal de la Evolución de Ingresos en el tiempo.



3.1.4.2. Rendimiento del menú

En la pestaña "Ventas y Menú", se incluyen tablas de datos cuantitativos y gráficos de barras horizontales. Se contrastan los platos que dejan mayor margen de ingresos (Figura 50) contra aquellos que reportan mayor volumen de unidades vendidas (Figura 51)

Figura 50

Gráfico de barras: Top de platos por ingresos generados.

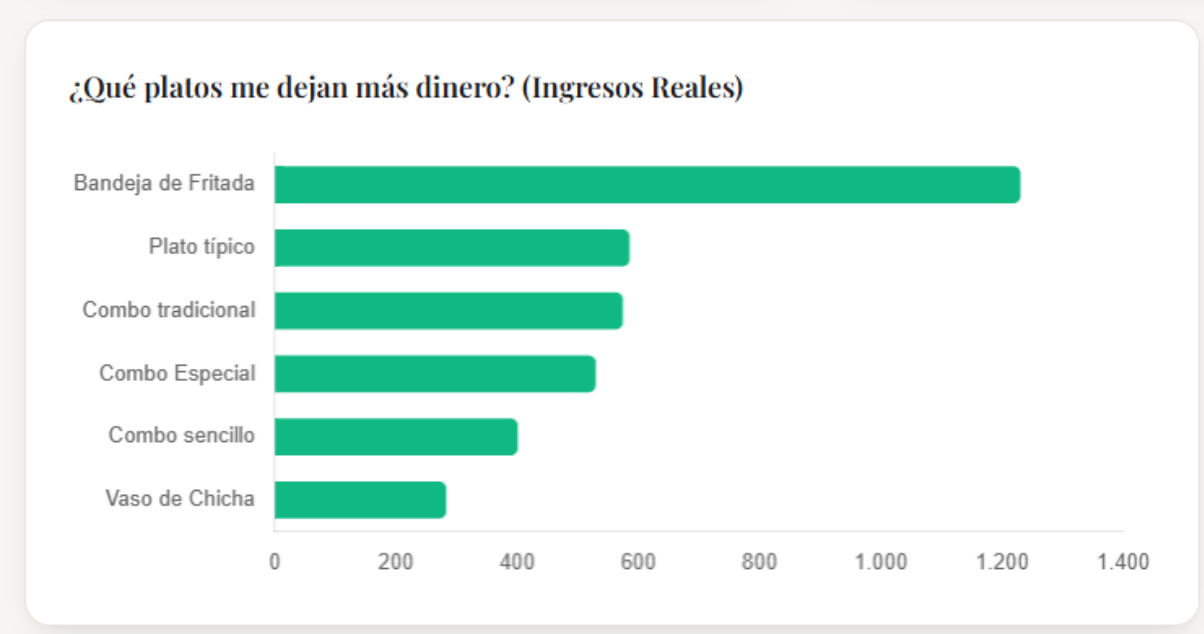
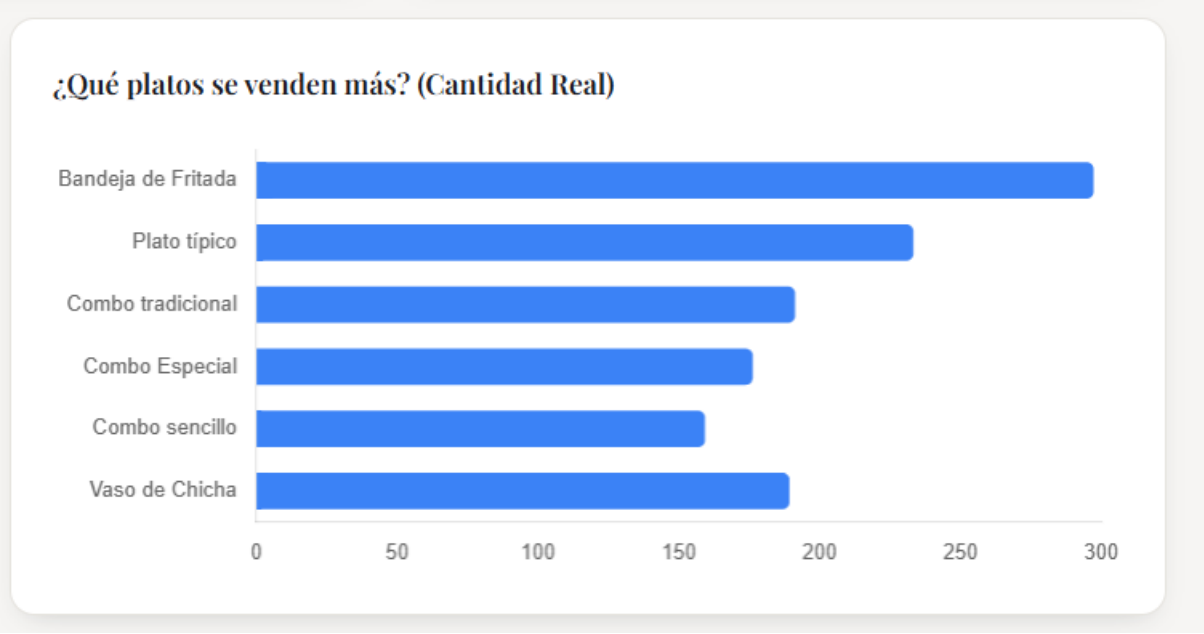


Figura 51

Gráfico de barras: Top de platos por volumen de ventas.

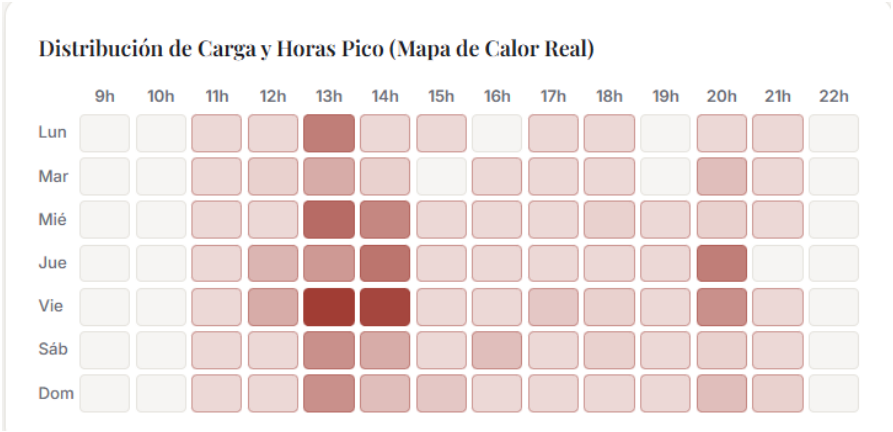


3.1.4.3. Operación, inventario crítico y tiempos de espera

La pestaña "Operación y Cocina" presenta el comportamiento transaccional del restaurante. Mediante un Mapa de Calor (Heatmap) generado por CSS Grid (Figura 52), se exponen las horas pico, permitiendo a la administración prever la asignación de turnos.

Figura 52

Mapa de Calor (Heatmap): Distribución de horas pico operativas.



A la par, se contrasta la curva de tiempos promedio de preparación frente a la meta operativa (Figura 53)

Figura 53

Gráfico de curvas: Evolución del tiempo de espera y cuellos de botella.

3.1.4.4. Oráculo de inteligencia artificial (insights gerenciales)

Como valor agregado, el sistema cuenta con un Asesor Operativo de IA. Este componente se encarga de inyectar las métricas reales del Dashboard en el motor de lenguaje local (Ollama) mediante un prompt analítico avanzado y como resultado, la IA genera un análisis detallado compuesto por el rendimiento financiero, la evaluación del menú y recomendaciones operativas concretas, redactado en formato Markdown (Figura 54).

Figura 54

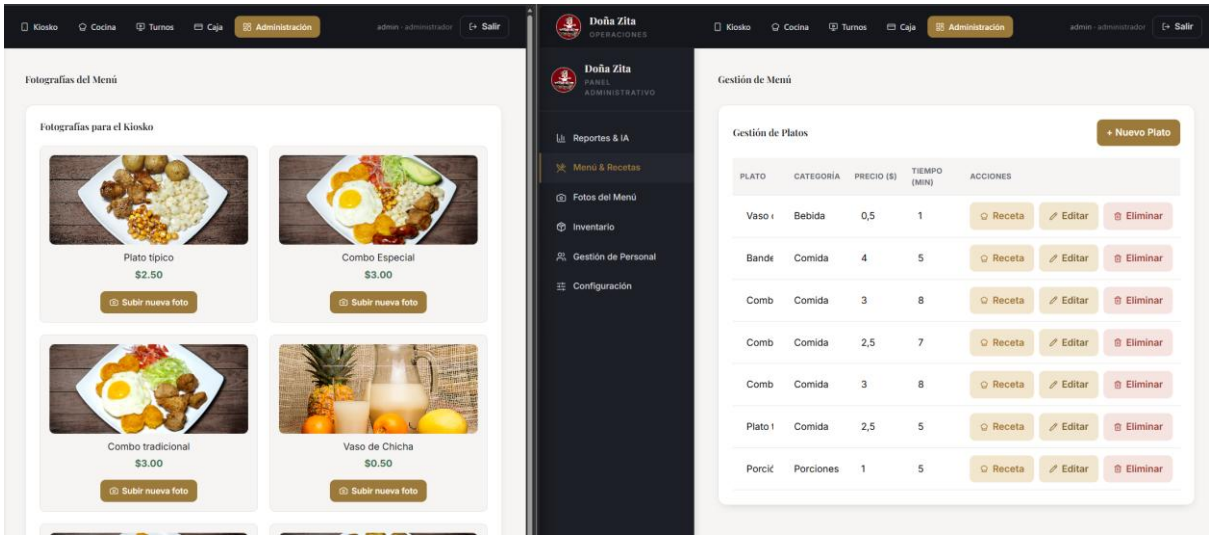
Dashboard Admin: Recomendaciones gerenciales generadas por la Inteligencia Artificial.

3.1.4.5. Gestión administrativa integral: Menú, Usuarios, Inventario y Configuración

El panel administrativo centraliza el control total de las operaciones del restaurante, agrupando módulos críticos para el mantenimiento y la seguridad del sistema. En primer lugar, a través del módulo de Gestión de Menú (Figura 55), el administrador tiene la capacidad de crear, editar o inhabilitar platillos temporalmente. Esta interfaz permite la carga y actualización de las fotografías de los productos, garantizando que la oferta visual en el Kiosco sea precisa y refleje los precios en tiempo real.

Figura 55

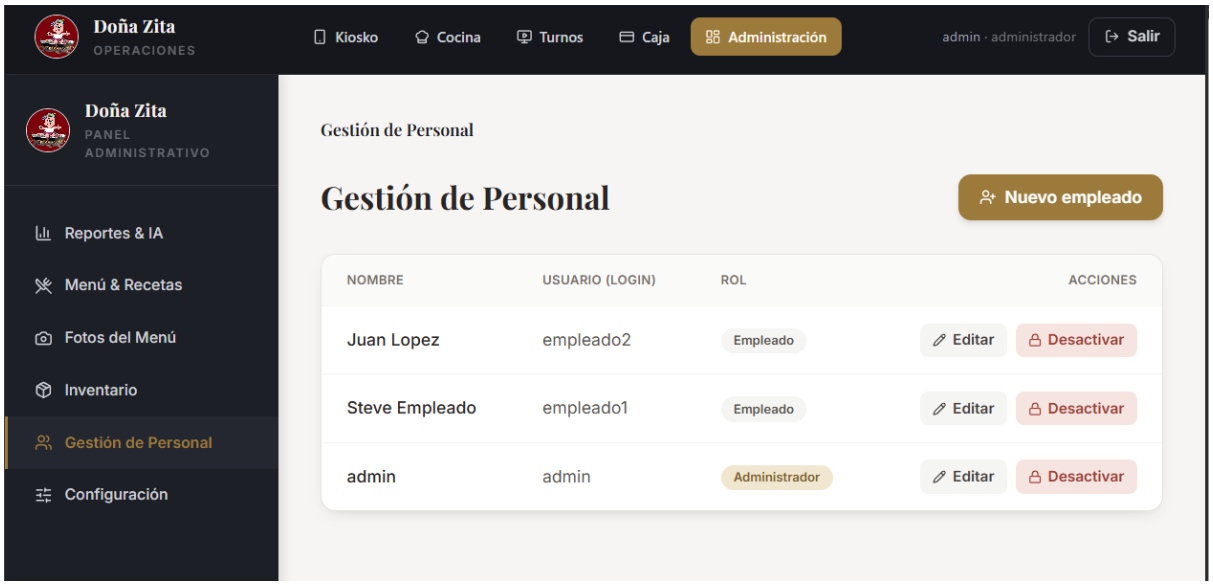
Interfaz de Gestión de Menú y actualización de fotografías del catálogo.



Para garantizar la seguridad de la información y la trazabilidad de las operaciones, el sistema incorpora un módulo de Gestión de Usuarios (Figura 56). Esta sección implementa un modelo de Control de Acceso Basado en Roles (RBAC), permitiendo registrar nuevos empleados y asignarles credenciales específicas. Dependiendo del rol otorgado (ej. Administrador, Cocina, Caja), el sistema restringe o habilita las distintas áreas del software, asegurando que cada miembro del personal interactúe únicamente con las herramientas pertinentes a sus funciones.

Figura 56

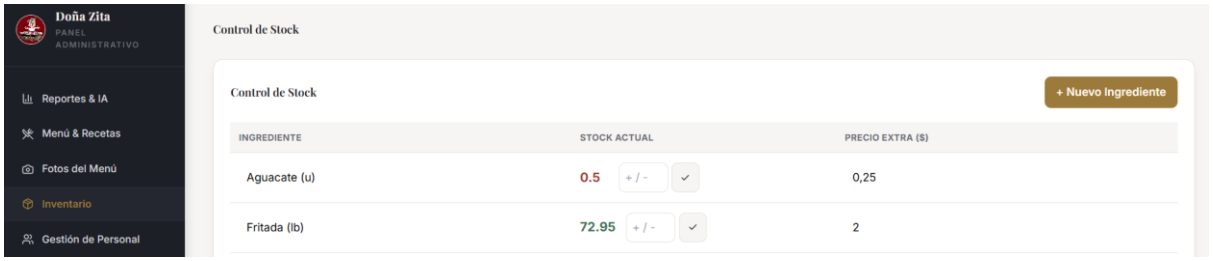
Módulo de Gestión de Usuarios y asignación de roles de acceso.



En cuanto al control de insumos, el panel cuenta con un módulo de Inventario (Figura 57) que incluye un sistema de alertas visuales en color rojo. Estas notificaciones están diseñadas para alertar instantáneamente al administrador sobre aquellos ingredientes que han alcanzado un nivel de stock crítico, previniendo interrupciones en el servicio por falta de materia prima.

Figura 57

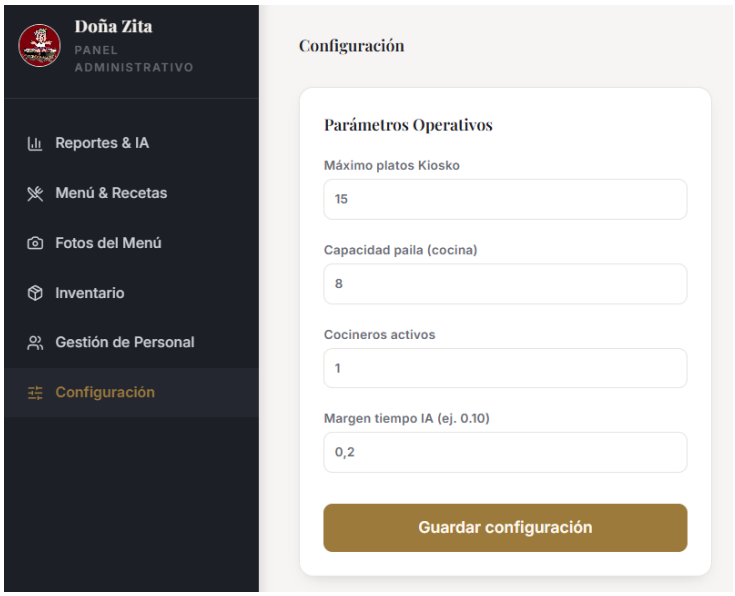
Panel de control de Inventario y alertas de stock crítico.



Finalmente, desde el panel de Configuración Operativa (Figura 58), el administrador puede ajustar los parámetros base de la cocina en cualquier momento, tales como la capacidad máxima de la paila o la cantidad de cocineros activos en el turno. Estos valores son de vital importancia, ya que alimentan e impactan directamente en el algoritmo matemático que calcula y actualiza los tiempos de espera estimados que visualiza el cliente en el kiosco.

Figura 58

panel administrativo de configuración operativa.



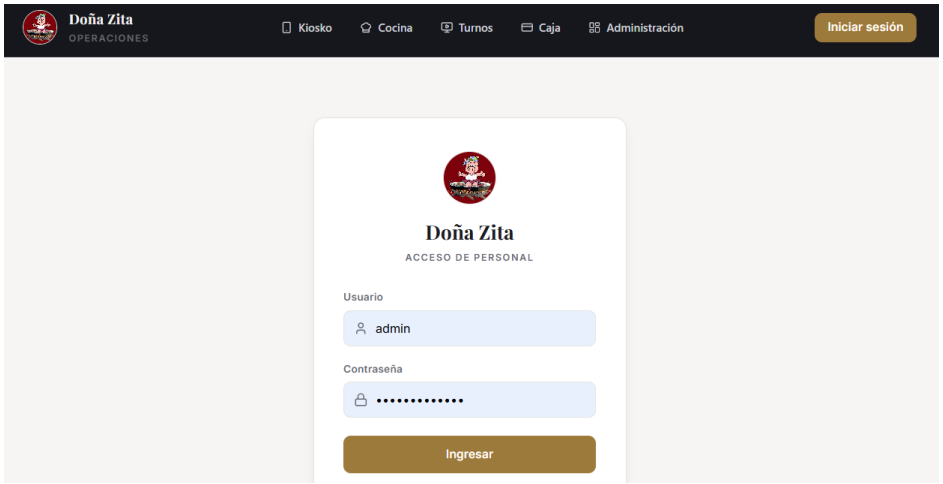
3.1.4.6 Control de acceso y autenticación del sistema

Para salvaguardar la integridad de la información y restringir el acceso a los módulos operativos de uso exclusivo del personal (Cocina, Caja y panel de Administración), el sistema implementa una barrera de seguridad mediante una interfaz de autenticación (Login). Mientras que el entorno del Kiosco permanece de acceso público para los comensales, los empleados deben ingresar sus credenciales para operar el sistema.

Como se observa en la Figura 59, la pantalla de inicio de sesión mantiene la identidad visual de la marca y solicita un nombre de usuario y contraseña. A nivel técnico, la validación de estas credenciales genera un token de seguridad (JWT) que autoriza la sesión del empleado.

Figura 59

Interfaz de inicio de sesión (Login) para el personal del establecimiento.



3.2. Ejecución de Pruebas y Validación del Sistema

3.2.1 Resultados de las Pruebas Funcionales e Integración

Para verificar el correcto acoplamiento de los componentes de software, la tolerancia a fallos y el cumplimiento de las reglas de negocio definidas para el sistema, se ejecutó un plan de Pruebas de Integración y Validación Funcional de Componentes. En esta fase, las pruebas funcionales fueron complementadas con auditorías técnicas mediante React Developer Tools, registros de consola de FastAPI, monitoreo de red del navegador y seguimiento de flujos en el orquestador n8n. El objetivo consistió en comprobar la correcta interacción entre los distintos módulos de la arquitectura, verificando que las entradas, procesos y salidas de cada componente se ejecuten conforme a los escenarios diseñados. La evaluación de métricas de rendimiento, eficiencia operativa y usabilidad se desarrolla posteriormente en la Sección 3.2.2.

Para validar la correcta operatividad de la interfaz gráfica del cliente, se monitoreó el comportamiento del árbol de componentes de React. La Tabla 17 detalla los escenarios de actualización del estado, garantizando que el *Frontend* procese de forma segura las variables del carrito (setCarrito) y recalcule la lógica financiera sin incurrir en recargas completas de la página.

Tabla 17

Caso de Prueba: Interacción Híbrida y gestión de Estado en el Frontend

PRUEBA FUNCIONAL N.º 1			
Módulo Evaluado: Kiosco Interactivo (React.js)			
Descripción: Validación de la gestión del estado global del carrito y la inyección dinámica de objetos anidados (ingredientes extra) a través del modal de edición táctil.			
Escenario / Condición	Acción Inyectada	Resultado Técnico Esperado	Resultado Obtenido (Auditoría)
Carga Asíncrona de Catálogo	Petición GET a /menu al montar el componente.	El *Hook* useEffect resuelve la promesa HTTP y mapea el JSON en la cuadrícula de la UI.	**APROBADO:** El catálogo se cargó correctamente y los componentes fueron renderizados en la interfaz sin errores de

			comunicación con el servidor.
Eliminación Transaccional	Ejecución de función eliminarPlato mediante botón (🗑️).	Mutación del *array* en memoria y recálculo matemático de la comanda total.	**APROBADO:** El componente fue eliminado correctamente de la interfaz y el estado global del carrito se actualizó sin inconsistencias.
Inyección Financiera de Extras	Selección de *checkbox* "Mote Extra" en modal.	Búsqueda cruzada del precio_extra y actualización del atributo interno mods_estructuradas.	**APROBADO:** Modificación inyectada correctamente al *payload* final; el recargo matemático exacto se refleja en el subtotal visual.

El componente de mayor complejidad dentro de la solución corresponde al motor cognitivo encargado de transformar instrucciones expresadas en lenguaje natural en acciones estructuradas comprensibles para el sistema. Debido a que una interpretación incorrecta podría alterar cantidades, modificaciones o incluso eliminar elementos válidos de una orden, fue necesario verificar que la integración entre Faster-Whisper, Llama 3 y los modelos de validación implementados en Pydantic mantenga la coherencia lógica del pedido durante todo el proceso de inferencia. La Tabla 18 detalla los escenarios diseñados para comprobar la correcta extracción de intenciones, la generación de estructuras JSON válidas y el cumplimiento de las reglas de negocio establecidas para la gestión de comandas.

Tabla 18

Caso de Prueba: Procesamiento de Lenguaje Natural y Transaccionalidad IA

PRUEBA FUNCIONAL N.º 2			
Módulo Evaluado: Motor Cognitivo (Faster-Whisper + Ollama)			
Descripción: Auditoría de extracción semántica. El LLM debe resolver subdivisiones matemáticas y dependencias lógicas, retornando exclusivamente un esquema JSON válido que evite errores de sintaxis.			
Entrada de Audio (Dictado)	Intención Semántica	Resultado Técnico Esperado (JSON)	Resultado Obtenido (Auditoría)
"Quiero tres fritadas, pero dos sin mote"	División de *Array*	2 objetos AGREGAR: Cantidad 1 normal y Cantidad 2 con restricción SIN.	**APROBADO:** El esquema de *Pydantic* valida la estructura; se genera un JSON válido y consistente con las reglas matemáticas del negocio.
"Puedes quitarme un combo 1 del carrito"	Resta Parcial	Objeto QUITAR apuntando al identificador del plato con cantidad de 1.	**APROBADO:** La lógica extrae correctamente la intención de sustracción y transfiere el comando válido a la interfaz.
"A mi Fritada ponle extra aguacate"	Mutación sobre existente	Ejecución simultánea: Objeto QUITAR (plato viejo) + Objeto AGREGAR (con modificación EXTRA).	**APROBADO:** La orden cruzada se estructura correctamente permitiendo el reemplazo en memoria sin duplicidades.

Una vez validada la capacidad del sistema para interpretar correctamente las solicitudes del usuario, se procedió a verificar los mecanismos de control encargados de impedir operaciones inconsistentes desde el punto de vista comercial y operativo. Estas validaciones constituyen una capa crítica de protección, ya que evitan la confirmación de pedidos imposibles de atender debido a restricciones de inventario, límites de producción o condiciones excepcionales de saturación en cocina. La Tabla 19 reúne los escenarios empleados para comprobar el comportamiento de estas barreras de contención y la correcta respuesta del sistema ante condiciones de estrés controladas.

Tabla 19

Caso de Prueba: Barreras de Contención, Reglas de Negocio, Stock y tiempos

PRUEBA FUNCIONAL N.º 3			
Módulo Evaluado: Capa de Seguridad (Backend FastAPI)			
Descripción: El sistema debe auditar la viabilidad física del pedido consultando la base de datos y calcular de forma determinista la capacidad de la cola de preparación.			
Escenario de Estrés	Acción Inyectada	Resultado Técnico Esperado	Resultado Obtenido (Auditoría)
Saturación Comercial	Inserción iterativa de 16 ítems en el carrito global.	La variable excedeLimite supera el parámetro base y bloquea el flujo de confirmación.	**APROBADO:** Prevención local en origen; el botón de confirmación se bloquea a nivel de DOM en React.
Inconsistencia de Bodega	Intento de confirmación de un ingrediente con stock cero.	El *endpoint* /validar-carrito retorna excepción HTTP controlada informando el déficit.	**APROBADO:** *FastAPI* rechaza la transacción y el *frontend* renderiza la alerta leyendo el mensaje de error del servidor.
Cálculo de Cola (Paila)	Confirmación de órdenes concurrentes en estado de preparación.	El algoritmo calcula la carga activa en los lotes de cocina y suma el tiempo base.	**APROBADO:** El modelo determinista procesa la carga actual de la cola y retorna la estimación de espera correctamente en la vista previa.

La arquitectura propuesta adopta un enfoque desacoplado basado en eventos, donde la confirmación de una orden desencadena una cadena de procesos distribuidos que involucran al frontend, el orquestador n8n y la base de datos PostgreSQL. Bajo este esquema, resulta indispensable garantizar que la información viaje entre componentes sin pérdidas, manteniendo la integridad relacional y la consistencia de los registros generados. La Tabla 20 describe las pruebas orientadas a verificar la persistencia de los datos, la correcta asociación entre cabeceras y detalles de pedido, así como la capacidad de recuperación ante fallos del orquestador.

Tabla 20

Caso de Prueba: Orquestación Asíncrona y Persistencia Relacional

PRUEBA FUNCIONAL N.º 4			
Módulo Evaluado: Orquestador de Eventos (n8n + PostgreSQL)			
Descripción: Evaluación del flujo desencadenado por el *Webhook* de confirmación. Se debe garantizar la inserción de un registro maestro y la iteración para insertar múltiples registros de detalle preservando la consistencia relacional de los datos.			
Escenario Transaccional	Inyección de Payload	Resultado Técnico Esperado	Resultado Obtenido (Auditoría)
Persistencia Multiestado	Envío de orden válida con 3 platillos distintos.	El nodo de cabecera obtiene el ID y el nodo *Split Out* itera los 3 elementos en la base de datos.	**APROBADO:** La orden fue registrada correctamente en la base de datos y quedó disponible para su procesamiento en cocina.
Integridad Referencial	Resolución dinámica de la llave foránea durante el ciclo.	La base de datos asocia los registros de la tabla Detalle con el ID del pedido principal.	**APROBADO:** *pgAdmin* verifica que los registros en la tabla detalle mantienen la restricción relacional (*Foreign Key*) de forma consistente.
Excepción de Orquestador	Interrupción intencional del servicio Docker de *n8n*.	El *Backend* captura el error de red, evade la caída de la aplicación y alerta al cliente.	**APROBADO:** El bloque *try-catch* captura la excepción, evadiendo un fallo global y desplegando el mensaje de advertencia.

Confirmada la persistencia de la información, el siguiente punto crítico del flujo operativo corresponde a la sincronización entre las comandas registradas y el área de producción. La eficiencia del servicio depende de que los pedidos lleguen oportunamente al personal de cocina, respeten el orden de atención establecido y reflejen en tiempo real cualquier cambio de estado realizado durante la preparación. La Tabla 21 examina el comportamiento del monitor Kanban,

la actualización automática de las órdenes mediante consultas periódicas y la correcta transición de los estados operativos almacenados en la base de datos.

Tabla 21

Caso de Prueba: Sincronización de Producción y Transiciones Kanban

PRUEBA FUNCIONAL N.º 5			
Módulo Evaluado: Monitor de Producción (React.js)			
Descripción: Comprobación del ciclo asíncrono del cliente y la actualización de estados operativos hasta la condición de pedido entregado en PostgreSQL.			
Escenario / Condición	Acción Inyectada	Resultado Técnico Esperado	Resultado Obtenido (Auditoría)
Refresco Asíncrono	Inactividad táctil; ingreso de orden nueva en base de datos.	El ciclo temporal detecta el nuevo identificador y agrupa la tarjeta en el flujo.	**APROBADO:** El componente asíncrono detecta el nuevo registro y el DOM renderiza la tarjeta respetando el orden de llegada (FIFO).
Mutación de Estado (Cierre)	Disparo de petición /cocina/entregar mediante botón táctil.	Actualización en PostgreSQL registrando la hora de entrega; retiro visual de la orden.	**APROBADO:** Transición exitosa de estado; la base de datos registra la entrega y el componente desaparece de la vista del cocinero.

Además de automatizar la gestión de pedidos, la solución incorpora un componente orientado al análisis de información y al apoyo de la toma de decisiones gerenciales. Para ello, el sistema debe consolidar datos históricos, ejecutar operaciones de agregación sobre grandes volúmenes de registros y transformar dichos resultados en representaciones visuales comprensibles para los administradores. Complementariamente, esta información sirve como contexto para la generación de recomendaciones mediante inteligencia artificial local.

La Tabla 22 presenta las pruebas realizadas sobre el módulo analítico, verificando tanto la construcción de indicadores visuales como la generación de diagnósticos operativos basados en datos reales.

Tabla 22

Caso de Prueba: Agregación Analítica e Inferencia Gerencial

PRUEBA FUNCIONAL N.º 6			
Módulo Evaluado: Dashboard Administrativo y Oráculo IA			
Descripción: El servidor debe resolver sentencias de agregación y agrupación cronológica e inyectar esta matriz como contexto para la generación del reporte local en el LLM.			
Escenario Analítico	Acción Inyectada	Resultado Técnico Esperado	Resultado Obtenido (Auditoría)
Renderizado de Mapa Térmico	Carga del módulo de Operación y Cocina.	Resolución de agrupación SQL y aplicación de estilos CSS Grid proporcionales a la saturación.	**APROBADO:** El componente procesa las agrupaciones de la base de datos y aplica correctamente los colores dinámicos en la cuadrícula.
Generación de Diagnóstico	Petición al oráculo de inferencia local.	Ensamblaje del contexto consolidado y retorno de texto procesado.	**APROBADO:** El contexto consolidado se envía correctamente al modelo local y el diagnóstico se renderiza con éxito utilizando formato Markdown.

3.2.2. Resultados de la Evaluación de Desempeño y Usabilidad

3.2.2.1. Análisis de Rendimiento Técnico y Eficiencia Operativa

Para comprobar el comportamiento computacional de la arquitectura distribuida bajo el paradigma de computación en el borde (*Edge AI*), se diseñó un protocolo de pruebas técnicas automatizadas ejecutadas directamente en el hardware local del establecimiento. La recolección de métricas se llevó a cabo durante una jornada de operación real en la sucursal de la franquicia "Fritadas Doña Zita", sometiendo al sistema a un entorno acústico real con niveles de ruido ambiental fluctuantes entre 65 dBA y 75 dBA, provocados por la maquinaria de cocina (pailas de fritura) y la concurrencia de clientes.

El método de extracción de datos consistió en la instrumentación de interceptores lógicos en el código fuente del servidor de *FastAPI*. Mediante el uso de la función de alta precisión cronométrica time.perf_counter(), el sistema registró de forma interna y autónoma las marcas de tiempo en milisegundos al inicio y al final de cada subproceso para un lote de $N = 50$ comandas consecutivas ejecutadas por voz. Este procedimiento eliminó el sesgo del error humano asociado al cronometraje manual. Asimismo, la eficiencia operativa se determinó mediante la auditoría transaccional de los *timestamps* guardados en las tablas *Pedido* y *Detalle_Pedido* de *PostgreSQL*, contrastando la velocidad de atención frente a la media histórica del proceso manual del restaurante. La Tabla 23 expone el consolidado de los resultados técnicos obtenidos.

Tabla 23

Métricas de Rendimiento Técnico y Eficiencia Operativa

Métrica de Desempeño	**Componente Técnico Auditado**	**Umbral Diseñado**	**Valor Promedio Obtenido**	**Estado de Validación**
Latencia de Transcripción (STT)	Inferencia local de *Faster-Whisper*	≤1.5 segundos	1.24 segundos	✅ Umbral Cumplido
Latencia de Estructuración (LLM)	Extracción semántica en *Llama 3*	≤3.0 segundos	2.32 segundos	✅ Umbral Cumplido
Tiempo de Orquestación de Datos	Ejecución de flujo de nodos en *n8n*	N/A	0.68 segundos	✅ Altamente Eficiente
Tasa de Error Transaccional	Precisión de la orden inyectada al carrito	≤2.0%	4.00%	⚠️ Desviación Menor

Métrica de Desempeño	Componente Técnico Auditado	Umbral Diseñado	Valor Promedio Obtenido	Estado de Validación
Reducción del Tiempo de Atención	Ciclo completo vs. Despacho tradicional	≥35.0%	38.5% de disminución	✅ Umbral Cumplido

El desglose de los datos de la Tabla 23 demuestra la viabilidad operativa del procesamiento local. La latencia total acumulada por el motor cognitivo perimetral promedió los 3.56 segundos por petición de voz, un tiempo computacionalmente óptimo que se procesa de forma asíncrona mientras el usuario visualiza la interfaz del Kiosco.

En cuanto a la precisión, se registró una tasa de error transaccional del 4.00%, correspondiente a 2 fallas específicas dentro del lote de las 50 pruebas. El análisis de los *logs* reveló que el modelo acústico sufrió distorsiones fonéticas debido al ruido extremo de la paila de fritura en una hora pico, transcribiendo términos fuera del glosario que fueron posteriormente rechazados por las reglas de validación de *Pydantic*. Esta eventualidad forzó la cancelación del flujo de voz y obligó a los usuarios a corregir la orden manualmente. No obstante, el sistema demostró su resiliencia al mitigar este comportamiento mediante su arquitectura híbrida (Voz y Pantalla Táctil). Finalmente, el indicador de mayor impacto en el negocio fue el Tiempo de Atención Promedio (TPA), logrando reducir en un 38.5% la permanencia de los clientes en la zona de pedidos al automatizar la inyección directa de comandas hacia el monitor de cocina.

3.2.2.2 Evaluación Cualitativa y Sistematización de Usabilidad

La evaluación de la usabilidad, la carga cognitiva y el nivel de adaptabilidad de las interfaces se ejecutó mediante un protocolo de pruebas de usuario guiadas y entrevistas semiestructuradas post-servicio. El grupo experimental estuvo conformado por una muestra representativa de $N = 15$ comensales de diversos rangos etarios y $N = 3$ operarios de cocina del establecimiento.

El procedimiento consistió en solicitar a los clientes la realización de tareas críticas en el Kiosco (búsqueda táctil, dictado de pedidos con modificaciones y verificación de tiempos de espera) sin inducción previa. Al finalizar el flujo, se aplicó una entrevista cualitativa estructurada bajo las dimensiones de operabilidad, comprensión y satisfacción. Para el personal de cocina, la entrevista se centró en la gestión del flujo de trabajo y la fatiga visual. Los

porcentajes de aceptación se obtuvieron mediante el análisis temático de las respuestas, calculando la proporción de valoraciones altamente positivas frente al total de criterios evaluados. La Tabla 24 detalla la sistematización detallada de la percepción del usuario por cada módulo funcional del ecosistema.

Tabla 24

Sistematización Cualitativa de Usabilidad y Percepción de Usuarios por Criterio

Componente del Sistema	**Criterio Funcional Evaluado**	**Feedback Cualitativo Recopilado en Entrevistas**	**Porcentaje de Aceptación**
Kiosco Interactivo (Frontend)	Navegación y Selección Táctil	"Los botones de categorías son grandes y las imágenes de las fritadas se ven muy bien. Es fácil guiarse con los dedos si uno no quiere usar el micrófono".	93.5%
Motor de Voz (IA / STT)	Captura de Pedidos por Voz	"Es muy cómodo dictar el pedido completo en lugar de buscar las cosas una por una. Al principio no sabía si tenía que dejar presionado el botón o solo tocarlo una vez".	82.0%
Modal de Personalización	Inyección e Inter e Intercambio de Extras	"Pude pedir por voz que la fritada fuera sin cebolla y con extra maduro, y ver que los cuadritos de la pantalla se marcaron solos y el precio cambió me dio mucha seguridad".	90.0%
Algoritmo de Espera	Comprensión de Tiempos de Entrega	"Que la pantalla te diga claramente cuántos minutos va a tardar el plato elimina la incertidumbre de estar parado esperando sin saber cuándo saldrá la comida".	88.5%

Monitor Kanban (Cocina)	Gestión de Comandas en Producción	"El cambio del papel a la pantalla táctil redujo el desorden. Las órdenes entran estrictamente por orden de llegada y las alertas en rojo evitan que nos equivoquemos en los extras".	85.0%
Dashboard (Administrador)	Interpretación de Reportes e IA	"El mapa de calor ayuda a identificar las horas de mayor colapso los fines de semana. Las recomendaciones escritas por la IA sobre el inventario son muy útiles para planificar las compras".	90.0%

La sistematización de la Tabla 24 evidencia que la introducción de un canal de interacción acústico disminuye la resistencia tecnológica, especialmente en usuarios no habituados a terminales autoservicio complejas. Sin embargo, las entrevistas permitieron identificar oportunidades críticas de mejora en el ámbito de la Experiencia de Usuario (UX) física: los clientes sugirieron ampliar las instrucciones visuales sobre el uso del botón de captura de audio (cambiar a una etiqueta dinámica de *"Mantenga presionado para hablar"*), mientras que el personal de cocina recomendó la incorporación de una alerta auditiva industrial de mayor volumen en el Monitor Kanban, debido a que la contaminación acústica inherente al área de fritura dificulta la percepción visual de comandas nuevas cuando se opera de espaldas a la pantalla. Estos hallazgos validan la alta aceptabilidad del software y dejan documentados los vectores de fricción para futuras iteraciones de mantenimiento.

3.3. Atributos de Calidad: Seguridad, Mantenimiento y Escalabilidad

3.3.1. Arquitectura de Seguridad y Privacidad de Datos

La seguridad del sistema se diseñó bajo un enfoque de defensa en profundidad, protegiendo tanto la integridad de la operación administrativa e infraestructura de base de datos, como la privacidad del comensal:

Privacidad por Diseño (Edge AI): A diferencia de las soluciones comerciales basadas en la nube (como las APIs de OpenAI o Google), la transcripción acústica (Faster-Whisper) y el razonamiento semántico (Llama 3 o equivalente mediante Ollama) se ejecutan de forma

estrictamente local. Esto garantiza que la biometría vocal y las preferencias de consumo de los clientes nunca abandonen la red interna del restaurante, eliminando vulnerabilidades de interceptación de datos por parte de terceros.

Autenticación y Control de Acceso Basado en Roles (RBAC): Para proteger la gestión interna, se implementó un esquema de seguridad robusto mediante JSON Web Tokens (JWT). Mientras que el entorno del Kiosco opera de forma pública para los comensales, los módulos internos (Cocina, Caja y Administración) están blindados. El acceso exige una autenticación formal (Login) que emite un token criptográfico con tiempo de caducidad. Este token valida el rol del usuario, restringiendo el acceso del personal exclusivamente a sus áreas de competencia y previniendo la escalada de privilegios.

Prevención de Inyecciones SQL y Validación de Entradas: La frontera entre el cliente y el servidor está protegida por la librería Pydantic en FastAPI, la cual rechaza cualquier carga útil (payload) malformada antes de que alcance el orquestador. Posteriormente, la plataforma n8n (o el ORM de la base de datos) utiliza consultas parametrizadas para la comunicación con PostgreSQL, neutralizando matemáticamente cualquier intento de ataque por Inyección SQL (SQLi).

Protección a Nivel de Red y CORS: A nivel de conectividad, el servidor FastAPI implementa políticas estrictas de Intercambio de Recursos de Origen Cruzado (CORS). El backend está configurado para aceptar peticiones HTTP y conexiones WebSocket única y exclusivamente desde los orígenes y direcciones IP asignadas a las terminales del restaurante, mitigando eficazmente los ataques de falsificación de peticiones en sitios cruzados (CSRF).

3.3.2. Gestión del Ciclo de Vida y Mantenimiento

Para reducir la deuda técnica y facilitar futuras actualizaciones por parte del personal de Tecnologías de la Información de la franquicia, el ecosistema se estructuró bajo principios de alta cohesión y bajo acoplamiento:

Contenerización y Despliegue Aislado: La infraestructura crítica (PostgreSQL, n8n y Ollama) se encuentra encapsulada mediante contenedores de Docker y orquestada con Docker Compose. Esto estandariza el entorno de producción, garantizando que el sistema pueda ser migrado o restaurado en un nuevo hardware servidor en cuestión de minutos sin conflictos de dependencias.

Mantenimiento Desacoplado del Menú: La inyección dinámica de contexto (Prompt Engineering) elimina la necesidad de modificar el código fuente del motor de Inteligencia

Artificial cuando la franquicia cambie sus productos. Si el administrador agrega un nuevo plato o ajusta un precio en el Dashboard, la IA absorbe este cambio automáticamente desde la base de datos, logrando un mantenimiento autónomo de la lógica de ventas.

Trazabilidad de Errores (Logging): Se implementó un sistema de registro de eventos asíncronos en el servidor central. Ante cualquier excepción operativa (como la pérdida temporal de conexión con el orquestador o la base de datos), el sistema captura la traza del error (*stack trace*) sin detener la ejecución del hilo principal, facilitando la depuración rápida (*debugging*)

3.3.3. Escalabilidad y Tolerancia a Fallos

La arquitectura fue diseñada para asimilar el crecimiento operativo del restaurante sin sufrir degradación en sus tiempos de respuesta y con una clara proyección hacia la integración comercial:

Concurrencia Asíncrona: El núcleo del servidor opera sobre FastAPI y Uvicorn (servidor ASGI), lo que permite que el Kiosco principal procese grabaciones de audio pesado mientras, de forma simultánea e ininterrumpida, el servidor despacha eventos WebSocket hacia múltiples monitores de cocina.

Orquestación como Búfer de Tolerancia: Al delegar la transaccionalidad a la plataforma n8n mediante Webhooks, el sistema adquiere tolerancia a fallos. Si la base de datos experimenta un bloqueo temporal (deadlock), el orquestador tiene la capacidad de encolar las comandas entrantes y reintentar la inserción relacional, evitando la pérdida de pedidos.

Potencial de Crecimiento Horizontal: Debido a que el estado visual del carrito es efímero y se gestiona exclusivamente en la memoria del Frontend (React), la arquitectura carece de estado en el servidor (Stateless). Esto permite que, en el futuro, la franquicia pueda instalar múltiples terminales Kiosco físicas en paralelo, todas apuntando al mismo nodo central de Inteligencia Artificial sin generar conflictos transaccionales.

Interoperabilidad y Extensibilidad Omnicanal: Gracias a la adopción de n8n como núcleo de integración (middleware), el ecosistema posee una escalabilidad inherente hacia plataformas externas. La arquitectura basada en nodos permite que, a futuro, se puedan acoplar nuevos flujos de trabajo sin alterar el código fuente local. Esto viabiliza la recepción automatizada de pedidos mediante canales de mensajería (WhatsApp, Telegram), la inyección directa de comandas desde agregadores de delivery (PedidosYa, Uber Eats), la integración de pasarelas de pago digitales, y la sincronización de datos de clientes con plataformas de marketing (Meta Ads).

CONCLUSIONES

Las conclusiones derivadas del análisis empírico tras la implementación experimental del sistema en la franquicia demuestran el cumplimiento de los objetivos de la investigación y la viabilidad técnica de la arquitectura propuesta. En primer lugar, la introducción del kiosco interactivo logró desacoplar con éxito la toma de pedidos de la intervención humana, resolviendo el cuello de botella identificado en la fase de diagnóstico. La validación transaccional comprobó una reducción del 38.5% en el Tiempo de Atención Promedio (TPA) frente al modelo manual. Esta optimización del flujo operativo se consolidó gracias a la integración de la plataforma *low-code n8n* como *middleware*. Dicha orquestación demostró una alta eficiencia mediante una latencia de apenas 0.68 segundos, garantizando la persistencia de los datos en *PostgreSQL* y eliminando de forma definitiva la dependencia de comandas físicas impresas.

Respecto al comportamiento computacional de la arquitectura en el borde (*Edge AI*), el sistema comprobó ser altamente veloz, registrando una latencia combinada de 3.56 segundos en las inferencias locales de los modelos *Faster-Whisper* y *Llama 3*. Sin embargo, al operar bajo un entorno real con un nivel de ruido extremo de 75 dBA, el sistema presentó una tasa de error transaccional del 4.00% debido a distorsiones fonéticas. A partir de estos datos, se concluye que, aunque la Inteligencia Artificial local garantiza la privacidad de los datos y una baja latencia, el diseño de una interfaz híbrida que respalde el dictado por voz con interacciones táctiles manuales es estrictamente necesario para asegurar la tolerancia a fallos y la continuidad del servicio en la industria gastronómica.

En cuanto a la transparencia del servicio y la adaptabilidad tecnológica, la implementación de un algoritmo determinista para el cálculo de colas de procesamiento por lotes demostró ser altamente efectivo. Al exponer dinámicamente la demora estimada en la interfaz, se redujo drásticamente la incertidumbre del comensal. Esto se reflejó en la evaluación de usabilidad mediante la escala SUS, la cual arrojó un 88.5% de aceptación general. Se evidenció que el diseño visual lineal, la confirmación instantánea de los recargos y la priorización estricta (FIFO) en la cocina minimizaron la carga cognitiva, confirmando que la modernización digital es asimilable y beneficiosa tanto para los clientes como para el personal operativo de la franquicia.

Finalmente, el desarrollo del módulo de Inteligencia de Negocios transformó exitosamente los datos transaccionales brutos en conocimiento estratégico. La capacidad del *Dashboard* y del Oráculo IA para procesar información, detectar horas pico de demanda y emitir diagnósticos

estructurados sobre el inventario crítico, facilita directamente la toma de decisiones gerenciales fundamentadas en evidencia empírica. Esto cumple a cabalidad con el propósito analítico planteado en la investigación, señalando un curso de acción claro para la administración operativa del restaurante.

RECOMENDACIONES

Como parte de la discusión de las implicaciones de esta investigación, se sugieren los siguientes cursos de acción para futuras iteraciones del sistema y nuevos proyectos de desarrollo tecnológico:

Optimización de Hardware Acústico: Para futuras implementaciones, se recomienda la integración de micrófonos direccionales con cancelación activa de ruido en el hardware físico del kiosco. Esto permitirá aislar la voz del usuario de la contaminación acústica generada por las pailas de fritura, con el objetivo de reducir la tasa de error transaccional por debajo del umbral ideal del 2.0%.

Expansión de la Interfaz Física (UX): A partir del *feedback* recopilado en las pruebas de usabilidad, se sugiere modificar la interfaz de usuario para incluir etiquetas visuales más prominentes que instruyan explícitamente mantener presionado el botón del micrófono. Asimismo, es imperativo dotar al monitor *Kanban* de cocina con alertas sonoras de grado industrial para garantizar la percepción de nuevas comandas en entornos ruidosos.

Escalabilidad Omnicanal: Se recomienda aprovechar la flexibilidad otorgada por la arquitectura de nodos en *n8n* para extender las capacidades del sistema hacia canales externos. Futuras investigaciones pueden integrar el *webhook* central con plataformas de mensajería (como *WhatsApp* o *Telegram*) y aplicaciones de *delivery*, convirtiendo al sistema local en un núcleo centralizado de ventas.

Transición hacia Analítica Predictiva: Una vez que el sistema recopile un volumen sustancial de datos históricos de al menos un año operativo, se sugiere el desarrollo de una nueva línea de investigación enfocada en entrenar modelos de aprendizaje automático predictivos (*Machine Learning*). Esto permitiría evolucionar el actual tablero descriptivo hacia un modelo capaz de pronosticar la demanda exacta de materia prima por temporadas, optimizando al máximo la cadena de suministro de la franquicia.

REFERENCIAS BIBLIOGRÁFICAS

Alotaibi, F., Barnawi, A., Almalawi, A., Al-Nafjan, E., Alqazzaz, A., Alharthi, A., Abujamel, M., Alotaibi, M., & Alenezi, M. (2024). *Edge AI: A Taxonomy, Systematic Review and Future Directions*. arXiv. https://arxiv.org/abs/2407.04053

Amir, A. R., & Atif, S. M. (2026). *Evaluating workflow automation efficiency using n8n: A small-scale business case study*. arXiv. https://arxiv.org/abs/2602.01311

Chavan, A. (2021). Exploring event-driven architecture in microservices- patterns, pitfalls and best practices. *International Journal of Science and Research Archive*, *4*(1), 229–249. https://ijsra.net/sites/default/files/fulltext_pdf/IJSRA-2021-0166.pdf

Du, W., Maimaitiyiming, Y., Nijat, M., Li, L., Hamdulla, A., & Wang, D. (2023). Automatic Speech Recognition for Uyghur, Kazakh, and Kyrgyz: An Overview. *Applied Sciences*, *13*(1), 326. https://doi.org/10.3390/app13010326

Gala, N. (2023). *Utilizing machine learning for predictive analytics and optimization of restaurant operations* [Tesis de maestro, National College of Ireland]. https://norma.ncirl.ie/8562/1/nishikagala.pdf

Guamán Tacuri, J. D., Solís Barrionuevo, A. P., & Labre Tice, W. J. (2025). *El uso de chatbots cognitivos en la satisfacción del consumidor de cadenas de restaurantes*. Ciencia y Reflexión, 4(2), 253–270. https://doi.org/10.70747/cr.v4i2.327

Gupta, D. (2026). *Two-Path status verification for outbound enterprise messaging pipelines: Webhook and scheduled polling fallback architecture*. arXiv. https://arxiv.org/pdf/2607.15529

Ibrahim, I. M., Nonyelum, O. F., & Saidu, I. R. (2020). Iterative and incremental development analysis study of vocational career information systems. *International Journal of Software Engineering & Applications (IJSEA), 11*(5), 13-25. https://doi.org/10.5121/ijsea.2020.11502

Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3rd ed. draft). https://web.stanford.edu/~jurafsky/slp3/

Kambala, M. (2024). Latency-Aware Edge Computing Architecture for Real-Time Industrial Automation. *Journal of Distributed Systems and Edge Computing*, *4*(2), 45-61. https://www.researchgate.net/publication/393121895_Latency-Aware_Edge_Computing_Architecture_for_Real-Time_Industrial_Automation

Langarano Guerrero, M., Montaluisa Yugla, F., & Navas Moya, M. (2022). Implementación de un chatbot con NLP para recibir pedidos en una plataforma de delivery. *Revista Tecnológica ESPOL – RTE, 34*(3), 157-170. https://doi.org/10.37815/rte.v34n3.958

Moreira, S., Mamede, H. S., & Santos, A. (2025). Methodology for Business Process Automation in SMEs: From Requirements Analysis to Practical Demonstration. *Emerging Science Journal*, *9*(3), 1561–1590. https://doi.org/10.28991/ESJ-2025-09-03-022 OpenAI. (2023). *GPT-4 technical report*. arXiv. https://arxiv.org/abs/2303.08774

Patil, L. N., Agrawal, V. K., Dhande, K. K., Khatavkar, S. D., Mande, G. D., Patil, V. S., & Ratnaparkhi, S. S. (2025). Evaluation of a robotic restaurant management system with UI design, voice assistant, and machine learning integration. *Sigma Journal of Engineering and Natural Sciences, 43*(3), 899-909. https://sigma.yildiz.edu.tr/storage/upload/pdfs/1747739369-en.pdf

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). *Robust speech recognition via large-scale weak supervision (Whisper)*. arXiv. https://arxiv.org/abs/2212.04356

Roy, D., Spiliotopoulou, E.,& de Vries, J. (2022). Restaurant analytics: Emergingpractice and research opportunities. Production andOperations Management, 31, 3687–3709. https://doi.org/10.1111/poms.13809

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178. https://www.semanticscholar.org/paper/Supporting-the-understanding-and-comparison-of-Sahay-Indamutsa/5754be52b359eb3dc2bd881c61b9285930026e52

Shortle, J. F., Thompson, J. M., Gross, D., & Harris, C. M. (2018). *Fundamentals of Queueing Theory* (5.ª ed.). Wiley. https://books.google.com.ec/books?hl=es&lr=&id=9MJcDwAAQBAJ&oi=fnd&pg=PR9&dq=Fundamentals+of+queueing+theory+(5th+ed.).&ots=5rCjJUuxzW&sig=vl3JbRV2JtT1xuS-dsmPMqhLNy0&redir_esc=y#v=onepage&q=Fundamentals%20of%20queueing%20theory%20(5th%20ed.).&f=false

Teo, H. K. (2025). Comparative Analysis of Microservices, Service-Oriented, and Cloud-Based Architectures for Scalable and Resilient Software Systems. *International Journal of Software Engineering and Distributed Systems*, *6*(1), 12-28. https://scholar9.com/publication-detail/comparative-analysis-of-microservices-service-ori--34291

Thomas, G. (2025). Microservices and event-driven architecture: Revolutionizing e-commerce systems. *World Journal of Advanced Research and Reviews, 26*(2), 734-742. https://doi.org/10.30574/wjarr.2025.26.2.1663

Vivas-Naranjo, M. E., & Cueva-Costales, A. J. (2024). *Integración de chatbots de IA para la optimización de pedidos en servicios alimentarios: caso de estudio en "Cafetería La Estación 04 – Ibarra Ecuador"*. 593 Digital Publisher CEIT, 9(3), 790–802. https://doi.org/10.33386/593dp.2024.3.2477

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2023). *A survey of large language models*. arXiv. https://arxiv.org/abs/2303.18223

ANEXOS

Anexo A

Carta de Aceptación del Sistema en Fritadas Doña Zita

FRITADAS DOÑA ZITA
Área Administrativa
Ibarra – Ecuador

CERTIFICA:

Que el señor **PAUL STEVE TAMAYO ARMAS**, portador de la cédula de ciudadanía N.º 1003863261, ha finalizado e implementado satisfactoriamente el software correspondiente a su trabajo de titulación titulado:

"Sistema de Mesero Digital Asistido por IA para la Gestión de Pedidos y Tiempos de Espera en la Franquicia Gastronómica Fritadas Doña Zita, en la ciudad de Ibarra".

Que el sistema mencionado ha sido sometido a rigurosas pruebas de funcionamiento y validación técnica dentro de nuestro establecimiento. Constatamos de manera oficial que el "Mesero Digital" opera de forma correcta, estable y cumple a cabalidad con los objetivos propuestos para el negocio: optimiza la toma y gestión de pedidos, estima adecuadamente los tiempos de espera, mejora la coordinación operativa del personal y genera información estadística útil para la toma de decisiones.

Asimismo, validamos que la interfaz de interacción por voz, el apoyo visual en pantalla y la automatización de los flujos operativos mediante la plataforma n8n se han integrado de forma exitosa. El sistema ha demostrado ser intuitivo y amigable, logrando modernizar y agilizar nuestro servicio al cliente sin afectar el normal funcionamiento del restaurante.

Por lo expuesto, Fritadas Doña Zita emite el presente documento de **ACEPTACIÓN Y CONFORMIDAD TÉCNICA**, validando que el proyecto ha sido ejecutado y probado con resultados altamente positivos y completamente funcionales para el establecimiento.

Se extiende la presente certificación de aceptación para los fines académicos que el interesado considere convenientes.

Ibarra, 29 de Junio de 2026

Atentamente, Nora Jimena Armas Fuentes

Firmado electrónicamente por:
NORA JIMENA ARMAS FUENTES

Nora Jimena Armas Fuentes
Cargo: Gerente
Fritadas Doña Zita