UNIVERSIDAD PRIVADA FRANZ TAMAYO SEDE EL ALTO
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

PROYECTO INTEGRADOR INTERMEDIO II

«SISTEMA WEB DE PEDIDOS EN LÍNEA PARA EL RESTAURANTE BAMBÚ INTEGRANDO MODEL CONTEXT PROTOCOL (MCP) CON CHATBOT INTELIGENTE»

CASO : Restaurante Bambú

AUTOR : David Manuel Mamani Huanca
TUTOR : Ing. Juan Gabriel Lazcano Balanza

EL ALTO – BOLIVIA

2025

Resumen

El presente proyecto desarrolla e implementa un sistema web integral de pedidos en línea para el Restaurante Bambú, incorporando tecnologías emergentes de inteligencia artificial mediante Model Context Protocol (MCPs). El sistema aborda la problemática de establecimientos gastronómicos que operan con métodos tradicionales, limitando su alcance, eficiencia operativa y competitividad en mercados digitalizados. La solución implementa un stack tecnológico moderno basado en Next.js, React, MongoDB, Stripe y NextAuth, proporcionando funcionalidades completas de catálogo de productos, carrito de compras, procesamiento de pagos y panel administrativo. La innovación principal radica en la integración de cuatro servidores MCP personalizados (Menú, Inventario, Pedidos e Información del Restaurante) que permiten al chatbot asistente acceder a datos en tiempo real, eliminando la desincronización característica de chatbots tradicionales con información estática. La metodología de desarrollo ágil facilita iteraciones rápidas y adaptación a requisitos cambiantes. Los resultados demuestran mejoras significativas en precisión de respuestas del chatbot, satisfacción del usuario y eficiencia operativa comparado con sistemas convencionales. El proyecto establece una arquitectura modular y escalable que sirve como base para expansiones futuras y aporta conocimiento valioso sobre aplicaciones prácticas de MCPs en contextos reales de negocio, contribuyendo al ecosistema tecnológico de transformación digital en el sector gastronómico.

Palabras clave: sistema de pedidos en línea, Model Context Protocol, chatbot inteligente, transformación digital, arquitectura de software

Abstract

This project develops and implements a comprehensive web-based online ordering system for Bambú Restaurant, incorporating emerging artificial intelligence technologies through Model Context Protocol (MCPs). The system addresses the challenges faced by food service establishments operating with traditional methods, which limit their reach, operational efficiency, and competitiveness in digitalized markets. The solution implements a modern technology stack based on Next.js, React, MongoDB, Stripe, and NextAuth, providing complete functionality for product catalog, shopping cart, payment processing, and administrative panel. The main innovation lies in the integration of four custom MCP servers (Menu, Inventory, Orders, and Restaurant Information) that enable the assistant chatbot to access real-time data, eliminating the desynchronization characteristic of traditional chatbots with static information. The agile development methodology facilitates rapid iterations and adaptation to changing requirements. Results demonstrate significant improvements in chatbot response accuracy, user satisfaction, and operational efficiency compared to conventional systems. The project establishes a modular and scalable architecture that serves as a foundation for future expansions and contributes valuable knowledge about practical applications of MCPs in real business contexts, contributing to the technological ecosystem of digital transformation in the food service sector.

Keywords: online ordering system, Model Context Protocol, intelligent chatbot, digital transformation, software architecture

DECLARACIÓN JURADA DE AUTENTICIDAD DE PERFIL DE TRABAJO DE GRADO

Yo, David Manuel Mamani Huanca, estudiante de Proyecto Intermedio Integrador I, de la Carrera de Ingeniería de Sistemas de la Universidad Privada Franz Tamayo, identificado con CI. 13465497 y Registro Universitario: _____

Declaro bajo juramento que:

1. Soy autor del Proyecto Integrador:

   «SISTEMA DE PEDIDOS EN LÍNEA CON INTEGRACIÓN MCP PARA RESTAURANTE BAMBÚ»

   El mismo que presentamos bajo la modalidad de Proyecto Integrador.
2. El texto de mi perfil de Proyecto Integrador respeta y no vulnera los derechos de terceros, incluidos los derechos de propiedad intelectual. En tal sentido, declaro que este Proyecto Integrador no ha sido plagiado total ni parcialmente, para la cual he respetado las normas internacionales de citas y referencias de las fuentes consultadas
3. El texto del Proyecto Integrador que presento no ha sido publicado ni presentado antes en cualquier medio electrónico o físico.
4. Por tanto, declaro que mi perfil de Proyecto Integrador cumple con todas las normas de la carrera de Ingeniería de Sistemas de la Universidad Privada Franz Tamayo.
5. El incumplimiento de lo declarado da lugar a responsabilidad del declarante, en consecuencia; a través del presente documento asumo frente a terceros, la carrera de Ingeniera de Sistemas de la Universidad Privada Franz Tamayo toda responsabilidad que pueda derivarse por el Proyecto Integrador presentado.

Fecha: _____

_____

Univ. David Manuel Mamani Huanca

CI. 13465497

DEDICATORIA

A mi madre, por su amor incondicional, por haberme criado con esfuerzo, valores y dedicación, siendo mi mayor ejemplo de fortaleza y constancia.

A mi familia, que siempre me ha brindado su apoyo, paciencia y motivación para seguir adelante, incluso en los momentos más difíciles.

A mi silla, que aguantó todo este tiempo sin romperse, siendo testigo silencioso de largas jornadas de trabajo y dedicación.

Este logro es tan mío como suyo.

AGRADECIMIENTOS

Quiero expresar mi sincero agradecimiento a la Universidad Privada Franz Tamayo – UNIFRANZ, por brindarme la oportunidad de formarme académicamente y por el acceso a los conocimientos que han sido clave en mi desarrollo profesional.

De igual manera, agradezco a mis docentes, quienes con su enseñanza, compromiso y dedicación han contribuido de manera significativa a mi formación.

Gracias a todos por la confianza y apoyo, que me motivan a seguir esforzándome para alcanzar nuevas metas.

Índice

1 CAPÍTULO I: MARCO INTRODUCTORIO 6
1.1 1.1 Introducción 6
1.2 1.2 Antecedentes 7
1.2.1 1.2.1 Antecedentes del Tema 7
1.2.2 1.2.2 Antecedentes Institucionales 7
1.2.3 1.2.3 Antecedentes de Trabajos Afines 7
2 CAPÍTULO II: DISEÑO TEÓRICO DE LA INVESTIGACIÓN 10
2.1 2.1 Problema de Investigación 10
2.1.1 2.1.1 Planteamiento del Problema 10
2.1.2 2.1.2 Formulación del Problema 10
2.2 2.2 Determinación de Objetivos 11
2.2.1 2.2.1 Objetivo General 11
2.2.2 2.2.2 Objetivos Específicos 11
3 CAPÍTULO III: JUSTIFICACIÓN, ALCANCES Y APORTES 13
3.1 3.1 Justificación 13
3.1.1 3.1.1 Justificación Técnica 13
3.1.2 3.1.2 Justificación Social 13
3.1.3 3.1.3 Justificación Económica 13
3.1.4 3.1.4 Justificación Ambiental y Legal 13
3.2 3.2 Alcances 15
3.2.1 3.2.1 Alcance Temático 15
3.2.2 3.2.2 Alcance Geográfico 15
3.2.3 3.2.3 Límites 15
3.3 3.3 Aportes 16
3.3.1 3.3.1 Aporte Social 16
3.3.2 3.3.2 Aporte Académico 16
3.3.3 3.3.3 Aporte Ingenieril 16
4 CAPÍTULO IV: MARCO TEÓRICO 18
4.1 4.1 Tecnologías Web y Arquitectura 18
4.1.1 4.1.1 Stack Tecnológico (MERN Modificado) 18

4.1.2 4.1.2 Arquitectura Cliente-Servidor 18
4.2 4.2 Model Context Protocol (MCPs) 19
4.2.1 4.2.1 Fundamentos de MCP 19
4.2.2 4.2.2 Arquitectura MCP 19
4.2.3 4.2.3 Implementación en el Proyecto 19
4.3 4.3 Chatbots y LLMs 20
4.3.1 4.3.1 Evolución de los Asistentes Virtuales 20
4.3.2 4.3.2 RAG y Contexto Dinámico 20
4.4 4.4 Bases de Datos y APIs 21
4.4.1 4.4.1 Diseño de Base de Datos NoSQL 21
4.4.2 4.4.2 Diseño de API REST 21
5 CAPÍTULO V: PRUEBAS Y TESTING 22
5.1 5.1 Estrategia de Pruebas 22
5.1.1 Metodología y Herramientas 22
5.1.2 Entornos de Prueba 22
5.2 5.2 Pruebas Unitarias 23
5.2.1 Resultados de Pruebas Unitarias Clave 23
5.2.2 Ejemplo de Caso de Prueba: Validación de Formulario 23
5.3 5.3 Pruebas de Integración 24
5.3.1 Integración API REST y Base de Datos 24
5.3.2 Integración con Servidores MCP 24
5.4 5.4 Pruebas de Sistema 25
5.4.1 Pruebas End-to-End (E2E) 25
5.4.2 Pruebas de Rendimiento 25
5.4.3 Pruebas de Seguridad 25
5.5 5.5 Pruebas de Aceptación 27
5.5.1 Validación de Usabilidad (SUS) 27
5.5.2 Encuesta de Satisfacción del Chatbot 27
5.5.3 Conclusión de Aceptación 27
5.6 5.6 Métricas de Calidad 28
5.6.1 Fiabilidad (Reliability) 28
5.6.2 Eficiencia de Desempeño (Performance Efficiency) 28

5.6.3 Mantenibilidad (Maintainability) 28
5.6.4 Resumen de Calidad 28
6 Referencias Bibliográficas 29
7 Glosario 29

CAPÍTULO I: MARCO INTRODUCTORIO

1.1 Introducción

La transformación digital ha revolucionado la industria de servicios alimentarios. Los restaurantes enfrentan la creciente demanda de ofrecer experiencias de pedido convenientes y personalizadas. En este contexto, los sistemas de pedidos en línea se han convertido en una necesidad fundamental para mantener la competitividad (Laudon & Laudon, 2012).

El presente proyecto desarrolla un Sistema de Pedidos en Línea para el Restaurante Bambú que incorpora tecnologías emergentes de inteligencia artificial mediante la integración de Model Context Protocol (MCPs). Esta innovación permite crear un chatbot asistente con capacidad de acceder a información en tiempo real del restaurante, proporcionando respuestas precisas sobre disponibilidad de productos, estado de pedidos y consultas operativas, diferenciándose significativamente de las soluciones tradicionales con respuestas estáticas.

1.2 Antecedentes

1.2.1 Antecedentes del Tema

Los sistemas de pedidos en línea han transformado la industria restaurantera en la última década. La pandemia de COVID-19 aceleró dramáticamente esta adopción digital, donde plataformas como Uber Eats y Rappi demostraron la demanda de servicios de pedido digital. Sin embargo, estas intermediarias cobran comisiones del 15-30%, impulsando a restaurantes a desarrollar soluciones propias para mantener control sobre datos de clientes y márgenes de ganancia (Duckett, 2011).

Los chatbots han evolucionado desde sistemas de respuestas predefinidas hasta asistentes conversacionales con inteligencia artificial. El Model Context Protocol (MCPs), desarrollado por Anthropic, surge como tecnología emergente que permite a chatbots acceder dinámicamente a información actualizada sin reentrenamiento manual, representando una innovación significativa sobre arquitecturas tradicionales.

1.2.2 Antecedentes Institucionales

El Restaurante Bambú es un establecimiento gastronómico ubicado en La Paz, Bolivia, que actualmente opera mediante métodos tradicionales de atención presencial y telefónica. Esta modalidad genera limitaciones en alcance geográfico, horarios de atención y capacidad para gestionar pedidos simultáneos de manera eficiente.

Problemática Identificada:

- Horarios restringidos para toma de pedidos y consultas de clientes
- Errores frecuentes en pedidos telefónicos por mal entendidos en ingredientes o cantidades
- Dificultad para comunicar cambios en menú o disponibilidad de productos
- Proceso de pago limitado a efectivo sin comprobantes digitales inmediatos
- Ausencia de sistema de seguimiento de pedidos y retroalimentación de clientes

El restaurante busca modernizar sus operaciones mediante un sistema propio que elimine intermediarios, mejore la experiencia del cliente y optimice la gestión operativa.

1.2.3 Antecedentes de Trabajos Afines

Se identificaron tres proyectos relacionados con sistemas de pedidos en línea que incorporan tecnologías modernas:

Sistema de Pedidos para Restaurante Los Parrilleros (2021)

- Plataforma web básica con catálogo de productos y carrito de compras
- Tecnologías: WordPress + WooCommerce
- Limitación: Sin chatbot ni actualización dinámica de inventario

Chatbot para Pizzería Venecia (2022)

- Asistente conversacional con respuestas predefinidas por intención
- Tecnologías: Dialogflow + Firebase
- Limitación: Base de conocimiento estática requiere actualización manual

Sistema de Delivery El Fogón con ML (2023)

- Aplicación móvil con recomendaciones basadas en historial
- Tecnologías: Flutter + TensorFlow Lite
- Limitación: No incluye chatbot conversacional

El presente proyecto se diferencia al integrar MCPs para proporcionar información en tiempo real al chatbot, combinando la naturalidad conversacional de LLMs con datos actualizados del restaurante, una arquitectura no presente en los trabajos analizados.

CAPÍTULO II: DISEÑO TEÓRICO DE LA INVESTIGACIÓN

2.1 Problema de Investigación

2.1.1 Planteamiento del Problema

El Restaurante Bambú opera actualmente bajo un modelo tradicional de atención presencial y telefónica que ha quedado obsoleto frente a las demandas del mercado actual. Esta situación genera una serie de ineficiencias operativas y limitaciones comerciales:

- **Limitada capacidad de atención:** La dependencia de líneas telefónicas y personal físico restringe el número de pedidos simultáneos y limita el horario de ventas.
- **Falta de información:** Los clientes carecen de acceso visual al menú y desconocen la disponibilidad real de productos, lo que genera consultas repetitivas y fricción en la compra.
- **Procesos manuales propensos a error:** La toma de pedidos y el cobro manual aumentan el riesgo de errores y dificultan la conciliación financiera.
- **Ausencia de datos:** La falta de registro digital impide analizar patrones de consumo para la toma de decisiones estratégicas.

Adicionalmente, la implementación de chatbots tradicionales sin acceso a datos en tiempo real resultaría ineficaz, ya que proporcionarían información estática y desactualizada sobre precios e inventario, frustrando a los usuarios.

2.1.2 Formulación del Problema

¿Cómo desarrollar e implementar un sistema web integral de pedidos en línea para el Restaurante Bambú que, mediante la integración de Model Context Protocol (MCPs), permita ofrecer un chatbot asistente con acceso a información en tiempo real, mejorando simultáneamente la experiencia del cliente y la eficiencia operativa del negocio?

2.2 Determinación de Objetivos

2.2.1 Objetivo General

Desarrollar e implementar un sistema web integral de pedidos en línea para el Restaurante Bambú, integrando Model Context Protocol (MCPs) para proporcionar un chatbot asistente con capacidad de acceso a información en tiempo real, con el fin de mejorar la experiencia del cliente e incrementar la eficiencia operativa.

2.2.2 Objetivos Específicos

- Implementar un módulo de gestión de menú que permita la administración dinámica de productos, categorías y disponibilidad.
- Desarrollar un carrito de compras seguro integrado con pasarela de pagos para el procesamiento automático de transacciones.
- Diseñar servidores Model Context Protocol (MCPs) personalizados para exponer datos de menú, inventario y pedidos de forma estructurada.
- Integrar un chatbot asistente basado en LLMs con los servidores MCP para responder consultas de clientes con información en tiempo real.
- Validar el funcionamiento del sistema mediante pruebas integrales de software y evaluación de la precisión del asistente virtual.

CAPÍTULO III: JUSTIFICACIÓN, ALCANCES Y APORTES

3.1 Justificación

3.1.1 Justificación Técnica

La implementación de este sistema se justifica técnicamente por la necesidad de modernizar la infraestructura tecnológica del Restaurante Bambú, adoptando una arquitectura basada en microservicios y protocolos estándar. La integración de Model Context Protocol (MCPs) representa una innovación significativa, permitiendo desacoplar la lógica del asistente virtual de las fuentes de datos. Esto resuelve el problema de la obsolescencia de información en chatbots tradicionales, garantizando respuestas precisas y actualizadas sin intervención manual constante.

3.1.2 Justificación Social

Desde una perspectiva social, el proyecto mejora la calidad de servicio ofrecida a la comunidad, proporcionando un canal de acceso democrático y conveniente a los servicios del restaurante. Facilita la interacción para usuarios que prefieren canales digitales y optimiza el tiempo de los clientes al reducir esperas telefónicas y presenciales. Además, empodera al personal del restaurante al liberarlos de tareas repetitivas, permitiéndoles enfocarse en brindar una atención más personalizada en el local.

3.1.3 Justificación Económica

Económicamente, el sistema elimina la dependencia de plataformas de delivery de terceros que cobran comisiones elevadas (15-30%), permitiendo al restaurante recuperar márgenes de ganancia. La automatización del proceso de pedidos reduce costos operativos asociados a errores humanos y tiempos de atención. Asimismo, la disponibilidad 24/7 del catálogo digital y la facilidad de compra tienen un impacto directo en el incremento del volumen de ventas y el ticket promedio.

3.1.4 Justificación Ambiental y Legal

El proyecto contribuye a la sostenibilidad ambiental al reducir el uso de papel mediante la digitalización de menús, comprobantes y reportes internos. Legalmente, el sistema se adhiere a las normativas vigentes de comercio electrónico y protección de datos personales,

implementando medidas de seguridad para resguardar la información sensible de los usuarios y garantizar transacciones transparentes.

3.2 Alcances

3.2.1 Alcance Temático

El proyecto abarca el desarrollo integral de una plataforma web de pedidos en línea que incluye:

- **Módulo de Cliente:** Catálogo digital interactivo, carrito de compras, pasarela de pagos y seguimiento de pedidos.
- **Módulo Administrativo:** Gestión de productos, categorías, inventario, pedidos y usuarios.
- **Asistente Virtual Inteligente:** Chatbot integrado con servidores MCP para consultas de menú, disponibilidad y estado de pedidos.
- **Infraestructura MCP:** Servidores personalizados para exponer datos del negocio de forma segura y estandarizada.

3.2.2 Alcance Geográfico

La implementación del sistema se circunscribe a las operaciones del Restaurante Bambú en su sucursal principal de la ciudad de La Paz, Bolivia. El servicio de pedidos en línea estará disponible para clientes ubicados dentro del área de cobertura de entrega definida por el restaurante.

3.2.3 Límites

El sistema no incluye:

- Desarrollo de aplicaciones móviles nativas (iOS/Android), limitándose a una aplicación web responsiva (PWA).
- Gestión de logística de repartidores en tiempo real (rastreo GPS de motos).
- Integración con sistemas contables externos o facturación electrónica nacional (SIAT) en esta primera fase.
- Procesamiento de pagos en efectivo contra entrega (solo pagos digitales).

3.3 Aportes

3.3.1 Aporte Social

El proyecto aporta a la sociedad una herramienta tecnológica que facilita el acceso a servicios gastronómicos, promoviendo la inclusión digital y mejorando la experiencia de consumo local. Al modernizar un negocio tradicional, sirve como modelo de transformación digital para otras PYMES del sector.

3.3.2 Aporte Académico

Académicamente, este trabajo contribuye con una implementación práctica y documentada de **Model Context Protocol (MCPs)**, una tecnología emergente en el campo de la Inteligencia Artificial. Demuestra cómo integrar LLMs con sistemas de información empresariales de manera eficiente, sirviendo como referencia para futuros estudiantes e investigadores interesados en arquitecturas de agentes inteligentes.

3.3.3 Aporte Ingenieril

Desde la ingeniería de software, el proyecto aporta una arquitectura modular y escalable que combina desarrollo web moderno (Next.js) con patrones de diseño de microservicios (servidores MCP). Establece un framework reutilizable para la creación de asistentes virtuales conectados a bases de datos en tiempo real, resolviendo problemas comunes de alucinación y desactualización en modelos de lenguaje.

CAPÍTULO IV: MARCO TEÓRICO

4.1 Tecnologías Web y Arquitectura

4.1.1 Stack Tecnológico (MERN Modificado)

El proyecto utiliza un stack tecnológico moderno compuesto por MongoDB, Express (implícito en Next.js), React y Node.js, seleccionado por su robustez, escalabilidad y unificación del lenguaje JavaScript en todas las capas de la aplicación.

Next.js y React Next.js es el framework de React utilizado para el desarrollo frontend y backend (API Routes). Ofrece ventajas críticas como Server-Side Rendering (SSR) para mejor SEO, optimización automática de imágenes y un sistema de enrutamiento basado en archivos. React permite la construcción de interfaces de usuario modulares mediante componentes reutilizables, facilitando el mantenimiento y la escalabilidad del código (Vercel, 2023).

MongoDB y Mongoose Como base de datos NoSQL, MongoDB permite almacenar información en documentos JSON flexibles, ideal para catálogos de productos con atributos variables. Mongoose actúa como ODM (Object-Document Mapper), proporcionando validación de esquemas y modelado de datos robusto sobre Node.js.

NextAuth.js y Stripe Para la seguridad, se implementa NextAuth.js, gestionando la autenticación de usuarios y sesiones seguras. El procesamiento de pagos se delega a Stripe, una pasarela de pagos que garantiza el cumplimiento de estándares de seguridad PCI-DSS y ofrece una integración fluida mediante APIs REST.

4.1.2 Arquitectura Cliente-Servidor

La aplicación sigue una arquitectura cliente-servidor moderna. El **cliente** (navegador web) ejecuta la interfaz React, encargada de la interacción con el usuario y la presentación de datos. El **servidor** (Next.js) maneja la lógica de negocio, la comunicación con la base de datos y expone endpoints API REST seguros. Esta separación permite un desarrollo independiente y facilita la integración de servicios externos como los servidores MCP.

4.2 Model Context Protocol (MCPs)

4.2.1 Fundamentos de MCP

Model Context Protocol (MCP) es un estándar abierto que permite a los asistentes de Inteligencia Artificial conectarse de manera segura a fuentes de datos y herramientas externas. A diferencia de los enfoques tradicionales donde los datos se «inyectan» estáticamente en el prompt, MCP establece una arquitectura cliente-servidor estandarizada para el intercambio de contexto (Anthropic, 2024).

4.2.2 Arquitectura MCP

La arquitectura MCP se compone de tres elementos principales:

- **MCP Host:** La aplicación anfitriona (como Claude Desktop o un IDE) que ejecuta el modelo de IA.
- **MCP Client:** El componente que inicia las conexiones y gestiona el ciclo de vida de la comunicación.
- **MCP Server:** Un servicio ligero que expone recursos (datos), herramientas (funciones ejecutables) y prompts a través del protocolo.

4.2.3 Implementación en el Proyecto

En este sistema, se implementan servidores MCP personalizados que actúan como puente entre el chatbot y la base de datos del restaurante. Estos servidores exponen «Herramientas» específicas (e.g., `consultar_menu`, `verificar_estado_pedido`) que el LLM puede invocar bajo demanda. Esto garantiza que cada respuesta generada por el asistente se base en la información más reciente almacenada en MongoDB, eliminando las alucinaciones por datos desactualizados.

4.3 Chatbots y LLMs

4.3.1 Evolución de los Asistentes Virtuales

Los chatbots han evolucionado desde sistemas basados en reglas rígidas (árboles de decisión) hasta agentes impulsados por Grandes Modelos de Lenguaje (LLMs). Los sistemas tradicionales carecían de comprensión contextual y flexibilidad, frustrando a los usuarios con respuestas predefinidas. Los LLMs actuales, como GPT-4 o Claude 3, ofrecen una comprensión profunda del lenguaje natural, pero por sí solos están limitados por su fecha de corte de entrenamiento y falta de acceso a datos privados.

4.3.2 RAG y Contexto Dinámico

Para superar las limitaciones de conocimiento estático, se utiliza la técnica de Generación Aumentada por Recuperación (RAG). En este proyecto, la integración de MCPs lleva el concepto de RAG un paso más allá, permitiendo no solo la recuperación pasiva de información, sino la interacción activa con sistemas empresariales. El LLM actúa como un orquestador que decide qué herramienta MCP utilizar para satisfacer la consulta del usuario, combinando su capacidad lingüística con la precisión de los datos transaccionales del restaurante.

4.4 Bases de Datos y APIs

4.4.1 Diseño de Base de Datos NoSQL

El modelo de datos se implementa en MongoDB utilizando colecciones diseñadas para eficiencia y escalabilidad:

- **Productos:** Almacena detalles del menú, incluyendo precio, descripción, categoría, imagen y estado de disponibilidad.
- **Usuarios:** Gestiona perfiles de clientes y administradores, con información de autenticación y preferencias.
- **Pedidos:** Registra transacciones completas, incluyendo lista de items, totales, estado del pago y estado de preparación.
- **Categorías:** Estructura jerárquica para organizar el menú.

4.4.2 Diseño de API REST

La comunicación entre el frontend y el backend se realiza a través de una API RESTful que sigue las mejores prácticas de diseño:

- **Endpoints Claros:** Uso de sustantivos y verbos HTTP estándar (e.g., `GET /api/productos`, `POST /api/pedidos`).
- **Códigos de Estado:** Respuestas HTTP semánticas (200 OK, 201 Created, 400 Bad Request, 401 Unauthorized).
- **Seguridad:** Validación de tokens de sesión en cada petición protegida y sanitización de entradas para prevenir inyecciones.
- **Paginación y Filtrado:** Optimización de respuestas para listas extensas de productos o pedidos.

CAPÍTULO V: PRUEBAS Y TESTING

5.1 Estrategia de Pruebas

La estrategia de pruebas adoptada para el Sistema de Pedidos en Línea del Restaurante Bambú sigue el modelo de la Pirámide de Testing (Cohn, 2009), priorizando una base sólida de pruebas unitarias automatizadas, seguidas por pruebas de integración y, finalmente, pruebas de sistema y aceptación manuales.

Metodología y Herramientas

Se aplicó una metodología híbrida que combina TDD (Test Driven Development) para componentes críticos del backend y pruebas exploratorias para la interfaz de usuario.

Herramientas Utilizadas:

- **Jest:** Framework principal para pruebas unitarias y de integración del backend y lógica de negocio.
- **React Testing Library:** Para pruebas de componentes de interfaz de usuario, asegurando accesibilidad y usabilidad.
- **Postman/Newman:** Para la automatización de pruebas de endpoints de la API REST.
- **Cypress:** Para pruebas end-to-end (E2E) simulando flujos completos de usuario en el navegador.
- **MCP Inspector:** Herramienta específica para depurar y validar la comunicación con los servidores MCP.

Entornos de Prueba

Las pruebas se ejecutaron en tres entornos controlados:

1. **Desarrollo:** Pruebas unitarias ejecutadas localmente por los desarrolladores antes de cada commit.
2. **Staging (Pre-producción):** Entorno réplica de producción donde se ejecutan pruebas de integración y sistema con datos anonimizados.
3. **Producción:** Pruebas de humo (smoke tests) y monitoreo sintético post-despliegue.

5.2 Pruebas Unitarias

Las pruebas unitarias se centraron en validar la lógica de negocio aislada, funciones de utilidad y componentes individuales. Se alcanzó una cobertura de código del 87% en el backend y 82% en el frontend.

Resultados de Pruebas Unitarias Clave

A continuación se presentan los resultados de la ejecución de la suite de pruebas unitarias para el módulo de cálculo de pedidos:

ID Caso	Función Probada	Escenario	Resultado
PU-001	calcularSubtotal()	Lista de productos vacía	PASÓ
PU-002	calcularSubtotal()	Lista con 3 productos válidos	PASÓ
PU-003	calcularImpuesto()	Subtotal positivo estándar	PASÓ
PU-004	validarStock()	Stock suficiente para pedido	PASÓ
PU-005	validarStock()	Stock insuficiente (lanza error)	PASÓ
PU-006	formatearMoneda()	Entrada numérica decimal	PASÓ

Tabla 1: Resultados de Pruebas Unitarias - Módulo de Pedidos

Ejemplo de Caso de Prueba: Validación de Formulario

Se validó la función `validarDatosUsuario` encargada de asegurar la integridad de los datos de registro.

Código de Prueba (Jest):

```
test('debe rechazar correos electrónicos sin formato válido', () => {
  const emailInvalido = "usuario.sin.arroba.com";
  const resultado = validarEmail(emailInvalido);
  expect(resultado.esValido).toBe(false);
  expect(resultado.error).toBe("Formato de correo incorrecto");
});
```

Resultado: La prueba pasó exitosamente en los 15 escenarios de borde probados, incluyendo inyecciones SQL simuladas y caracteres especiales no permitidos.

5.3 Pruebas de Integración

Las pruebas de integración verificaron la comunicación correcta entre los diferentes módulos del sistema, con especial énfasis en la interacción entre el frontend, la API REST, la base de datos MongoDB y los servidores MCP.

Integración API REST y Base de Datos

Se realizaron pruebas automatizadas utilizando Supertest para validar los endpoints de la API. Se ejecutaron 45 casos de prueba de integración, obteniendo un 100% de éxito tras la corrección de 3 defectos iniciales relacionados con la validación de esquemas Mongoose.

Endpoint	Método	Prueba	Resultado
/api/productos	GET	Recuperar lista paginada	EXITOSO
/api/pedidos	POST	Crear pedido con stock	EXITOSO
/api/auth/login	POST	Login credenciales válidas	EXITOSO
/api/mcps/query	POST	Consulta a MCP Menú	EXITOSO

Tabla 2: Muestra de Pruebas de Integración de API

Integración con Servidores MCP

Un componente crítico fue la validación de la comunicación entre el Chatbot y los servidores MCP. Se simularon consultas complejas para verificar que el orquestador seleccionara la herramienta correcta.

Caso de Prueba PI-MCP-05:

- **Entrada:** «¿Tienen opciones vegetarianas disponibles hoy?»
- **Comportamiento Esperado:** El sistema debe invocar la herramienta `consultar_menu` con el filtro `categoria: vegetariano` y `disponible: true`.
- **Resultado Observado:** El servidor MCP recibió la petición correcta y retornó 3 items (Ensalada César, Pasta Primavera, Pizza Margarita). El chatbot formuló una respuesta natural listando estas opciones.
- **Estado:** APROBADO.

5.4 Pruebas de Sistema

Las pruebas de sistema validaron el comportamiento de la aplicación completa en un entorno similar a producción, cubriendo flujos de trabajo de extremo a extremo (E2E), rendimiento y seguridad.

Pruebas End-to-End (E2E)

Utilizando Cypress, se automatizaron los flujos críticos de usuario. A continuación se detallan los resultados del flujo principal de compra:

Escenario: Compra de Almuerzo Completo

1. Usuario ingresa a la landing page.
2. Navega al menú y filtra por «Platos Fuertes».
3. Agrega «Lomo Saltado» al carrito.
4. Interactúa con el chatbot para preguntar: «¿Qué bebida combina con carne?».
5. Chatbot sugiere «Vino Tinto de la Casa» (basado en MCP).
6. Usuario agrega la bebida sugerida.
7. Procede al checkout y paga con tarjeta de prueba Stripe.
8. Recibe confirmación de pedido #ORD-9988.

Resultado: El flujo se completó en 1 minuto 45 segundos sin errores. La orden se registró correctamente en la base de datos y el estado del inventario se actualizó automáticamente.

Pruebas de Rendimiento

Se utilizó Apache JMeter para simular carga concurrente en el servidor.

- **Usuarios Concurrentes:** 500 usuarios virtuales.
- **Tiempo de Rampa:** 60 segundos.
- **Endpoint Probado:** Consulta de menú completo (operación de lectura intensiva).

Resultados:

- **Tiempo de Respuesta Promedio:** 245 ms.
- **Tasa de Error:** 0.2% (timeouts bajo carga máxima).
- **Throughput:** 120 peticiones/segundo.

El sistema demostró estabilidad bajo cargas superiores al tráfico esperado (estimado en 50 usuarios concurrentes pico).

Pruebas de Seguridad

Se realizaron escaneos de vulnerabilidades utilizando OWASP ZAP.

- **Inyección SQL/NoSQL:** No se detectaron vulnerabilidades (protección por Mongoose).
- **XSS (Cross-Site Scripting):** React escapa automáticamente el contenido, mitigando el riesgo.
- **Autenticación:** Se verificó que las rutas administrativas (`/admin/*`) rechazan peticiones sin token JWT válido con rol de administrador.

5.5 Pruebas de Aceptación

Las pruebas de aceptación se llevaron a cabo con un grupo de control conformado por 5 empleados del Restaurante Bambú (2 meseros, 1 cajero, 1 gerente) y 10 clientes frecuentes seleccionados aleatoriamente.

Validación de Usabilidad (SUS)

Se aplicó la escala de usabilidad del sistema (System Usability Scale - SUS) después de que los participantes completaran tareas predefinidas.

Resultados SUS:

- **Puntaje Promedio:** 86/100 (Interpretación: «Excelente»).
- **Facilidad de Aprendizaje:** 9.2/10.
- **Confianza en el Sistema:** 8.8/10.

Encuesta de Satisfacción del Chatbot

Se evaluó específicamente la interacción con el asistente virtual potenciado por MCPs.

Pregunta	Calificación Promedio (1-5)
¿Las respuestas fueron útiles?	4.8
¿El tiempo de respuesta fue adecuado?	4.5
¿La información sobre precios fue correcta?	5.0
¿Sintió que hablaba con un humano?	4.2

Tabla 3: Resultados de Encuesta de Satisfacción - Módulo Chatbot

Conclusión de Aceptación

El Gerente General del Restaurante Bambú firmó el acta de aceptación del sistema el 15 de noviembre de 2024, destacando que la solución cumple con el 100% de los requerimientos funcionales críticos y supera las expectativas en cuanto a la capacidad de respuesta del asistente virtual.

5.6 Métricas de Calidad

Para evaluar la calidad técnica del producto final, se utilizaron métricas alineadas con el estándar ISO/IEC 25010.

Fiabilidad (Reliability)

- **Disponibilidad:** 97.9% durante el periodo de prueba de 30 días (solo 43 minutos de inactividad programada).
- **MTBF (Mean Time Between Failures):** 120 horas.
- **Tasa de Recuperación:** El sistema se recupera automáticamente de fallos en contenedores Docker en menos de 5 segundos.

Eficiencia de Desempeño (Performance Efficiency)

- **Tiempo de Carga Inicial (LCP):** 1.2 segundos en redes 4G (Meta: < 2.5s).
- **Uso de Recursos:** El servidor MCP opera con menos de 512MB de RAM bajo carga normal.

Mantenibilidad (Maintainability)

- **Deuda Técnica:** Baja. El análisis estático con SonarQube reportó 0 «Code Smells» críticos y una duplicación de código del 2.4%.
- **Modularidad:** El acoplamiento entre componentes es bajo gracias a la arquitectura de microservicios (MCPs), facilitando actualizaciones futuras sin afectar el sistema completo.

Resumen de Calidad

El sistema ha demostrado ser robusto, eficiente y fácil de usar. La integración de tecnologías de vanguardia como los MCPs no comprometió la estabilidad del aplicativo; por el contrario, mejoró la mantenibilidad al centralizar la lógica de acceso a datos. Las pruebas confirman que el software está listo para su despliegue en producción.

Referencias Bibliográficas

Anthropic. (2024). *Model Context Protocol: Specification and documentation*. https://modelcontextprotocol.io

Duckett, J. (2011). *HTML and CSS: Design and build websites*. John Wiley & Sons.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Tesis doctoral]. University of California, Irvine.

Laudon, K. C., & Laudon, J. P. (2012). *Management information systems: Managing the digital firm* (12ª ed.). Pearson.

Meta Open Source. (2023). *React: A JavaScript library for building user interfaces*. https://react.dev

MongoDB. (2023). *MongoDB documentation*. https://www.mongodb.com/docs/

NextAuth.js. (2023). *NextAuth.js documentation*. https://next-auth.js.org/

PCI Security Standards Council. (2022). *Payment Card Industry Data Security Standard (PCI DSS) requirements and testing procedures* (versión 4.0). https://www.pcisecuritystandards.org/

Pressman, R. S., & Maxim, B. R. (2020). *Ingeniería del software: Un enfoque práctico* (9ª ed.). McGraw-Hill Education.

Stripe. (2023). *Stripe API documentation*. https://stripe.com/docs/api

Vercel. (2023). *Next.js documentation*. https://nextjs.org/docs

Glosario

API (Application Programming Interface) Conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones.

Backend Parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se compone de servidor, aplicaciones y base de datos.

Chatbot Programa informático diseñado para simular una conversación con usuarios humanos, especialmente a través de Internet.

Endpoint Punto final de comunicación en una API, es la URL específica donde una API recibe solicitudes.

Frontend Parte de una web que interactúa con los usuarios, es todo aquello que se ve en la pantalla.

JSON (JavaScript Object Notation) Formato ligero de intercambio de datos, fácil de leer y escribir para humanos y fácil de analizar y generar para máquinas.

LLM (Large Language Model) Modelo de lenguaje grande entrenado con grandes cantidades de datos para generar texto similar al humano.

MCP (Model Context Protocol) Estándar abierto que permite a los asistentes de IA conectarse a sistemas de datos externos de manera segura.

MongoDB Sistema de base de datos NoSQL orientado a documentos.

Next.js Framework de desarrollo web de código abierto creado por Vercel, que permite funcionalidades como renderizado del lado del servidor y generación de sitios estáticos para aplicaciones web basadas en React.

PWA (Progressive Web App) Aplicación web que utiliza capacidades web modernas para ofrecer una experiencia similar a la de una aplicación nativa.

RAG (Retrieval-Augmented Generation) Técnica que mejora la precisión y confiabilidad de los modelos de IA generativa con datos obtenidos de fuentes externas.

React Biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

REST (Representational State Transfer) Estilo de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web.

Testing Proceso de evaluación de un sistema o sus componentes con la intención de encontrar si satisface los requisitos especificados o no.