UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

Facultad de Ciencias Sociales y Jurídicas de Orihuela

Área de Administración y Dirección de Empresas

ChatGPT para Reservas de Restaurantes- Guía Práctica

Trabajo Fin de Grado

Autor/a: Francisco José Tovar Martínez

Tutor/a: Pedro Campillo

Orihuela, 06 2025

Curso académico 2024/2025

Resumen

El presente Trabajo de Fin de Grado (TFG) aborda el desarrollo de una guía práctica y exhaustiva que sirva como manual para la creación e implementación de un agente de IA basado en ChatGPT, específicamente diseñado para la gestión de reservas de mesas en restaurantes. Para alcanzar este objetivo, se ha realizado una revisión del estado del arte sobre la IA en la hostelería y los sistemas de reserva, seguido del diseño conceptual del agente. El núcleo del trabajo se centra en la guía de implementación, detallando la configuración inicial, la interacción con la API de ChatGPT, la ingeniería de prompts, la gestión del diálogo y la crucial integración con sistemas externos mediante la funcionalidad de function calling. Finalmente, se analizan los desafíos técnicos, operativos y éticos inherentes, y se exponen las conclusiones del estudio, reconociendo la viabilidad de la solución propuesta y su potencial para optimizar operaciones y mejorar la satisfacción del cliente, al tiempo que se identifican limitaciones y se proponen futuras líneas de investigación.

Abstract

This Final Degree Project (TFG) addresses the development of a comprehensive and practical guide to serve as a manual for the creation and implementation of a ChatGPT-based AI agent, specifically designed for managing restaurant table reservations. To achieve this objective, a state-of-the-art review of AI in the hospitality industry and reservation systems was conducted, followed by the agent's conceptual design. The core of the project focuses on the implementation guide, detailing the initial setup, interaction with the ChatGPT API, prompt engineering, dialogue management, and the crucial integration with external systems using the function calling feature. Finally, the inherent technical, operational, and ethical challenges are analyzed, and the study's conclusions are presented, acknowledging the viability of the proposed solution and its potential to optimize operations and improve customer satisfaction, while also identifying limitations and proposing future lines of research.

Palabras Clave: Inteligencia Artificial, ChatGPT, Reservas de Restaurantes, Guía Práctica, API, Function Calling, Procesamiento del Lenguaje Natural, Hostelería, Automatización, Experiencia del Cliente.

Índice General

1. Introducción.

1.1. Motivación de la investigación.

Este Trabajo de Fin de Grado (TFG) nace de la convicción de que la inteligencia artificial (IA) puede revolucionar el sector de la restauración. Inicialmente, la idea era general, pero rápidamente evolucionó hacia la identificación de los retos y dificultades específicos que enfrentan las empresas de hostelería, y cómo la IA podría ofrecer soluciones. Es innegable que los métodos tradicionales de gestión de reservas, como las llamadas telefónicas y el seguimiento manual, presentan serias limitaciones. Estos enfoques no solo consumen una cantidad excesiva de tiempo del personal, sino que también son propensos a errores y demoras en la atención al cliente. La consecuencia directa es una pérdida de clientes y un incremento de los costes operativos para el empresario.

En un mundo donde el tiempo es un recurso escaso y valioso, los consumidores recurren cada vez más a la tecnología para agilizar sus tareas diarias, y la reserva de mesas en restaurantes no es una excepción (Davis, 2024). Precisamente, la implementación de la IA en los negocios de hostelería ofrece un abanico de ventajas competitivas. Estas incluyen la posibilidad de que los clientes reserven online, recibir recomendaciones personalizadas basadas en sus preferencias, posibles alergias o la asistencia de niños, y la actualización de disponibilidad en tiempo real. Además, la IA facilita la gestión predictiva de las reservas y la automatización de tareas operativas, lo que se traduce directamente en una reducción significativa de los costes. A pesar de este potencial, es importante señalar que la adopción de la IA en la industria de la hostelería aún se encuentra en sus etapas iniciales. Sin embargo, su capacidad para mejorar la eficiencia operativa, personalizar las experiencias, reducir costes y aumentar la satisfacción y fidelidad del cliente es inmensa (Alawami et al., 2025). Esta brecha entre el potencial y la implementación actual es precisamente lo que impulsa nuestra investigación.

1.2. Objetivos

Partiendo de la necesidad identificada y el vasto potencial de la IA, el objetivo principal de este TFG es desarrollar una guía práctica y exhaustiva, concebida como un manual, para la creación e implementación de un agente de inteligencia artificial basado en ChatGPT. Este agente estará diseñado específicamente para la gestión de reservas de mesas en restaurantes. La guía abordará de forma integral todo el proceso, desde la conceptualización hasta la puesta en marcha. Se detallará cómo aprovechar la API de ChatGPT para manejar las solicitudes de reserva de manera eficiente, capturando información clave como la fecha, hora y número de comensales. Más allá de lo básico, se profundizará en la gestión de detalles cruciales del cliente, como alergias, preferencias o la presencia de niños, garantizando una experiencia personalizada. Además, se explorará la interacción necesaria con posibles sistemas *back-end* del restaurante, todo ello con el fin último de asegurar una experiencia de usuario fluida, eficiente y sin fricciones

1.3. Impacto esperado

La relevancia de este TFG radica en su capacidad para acercar el potencial de los LLMs avanzados como ChatGPT a las empresas, y en particular al sector de la hostelería. Al proporcionar una guía práctica y accesible, este trabajo busca suplir la carencia actual de recursos que faciliten la aplicación de estas tecnologías en el proceso de reservas de restaurantes. El impacto esperado es doble: por un lado, empoderar a los hosteleros con herramientas que optimicen sus operaciones; por otro, mejorar la experiencia del cliente, fomentando una mayor satisfacción y lealtad.

1.4. Estructura de la memoria

Este documento se organiza de la siguiente manera: En el Capítulo 1 se exponen los motivos, objetivos e impacto esperado de este trabajo. El Capítulo 2 presenta una revisión del Estado del Arte sobre la IA en hostelería y sistemas de reserva. El Capítulo 3 trata el diseño conceptual del agente de reservas. El Capítulo 4 constituye la guía práctica de implementación. El Capítulo 5 analiza los desafíos técnicos, operativos y éticos inherentes a la implementación de dicho agente. Finalmente, el Capítulo 6 ofrece las conclusiones del trabajo y sugiere posibles líneas de investigación futura.

2. Estado del Arte

2.1. Chatbots de Inteligencia artificial (IA) y tecnologías que los componen.

Para comprender el diseño de nuestro agente, es fundamental primero definir los chatbots de IA. Estos son programas informáticos diseñados para interactuar con humanos utilizando lenguaje natural, simulando una conversación. Su sofisticación se basa en la integración de diversas tecnologías de IA. Principalmente, el Procesamiento del Lenguaje Natural (PNL), cuyo objetivo es identificar y extraer el contenido subjetivo y emocional del texto, llamado análisis de sentimientos (Celi-Parraga et al., 2021). Esto permite a los chatbots no solo comprender las intenciones y matices de las consultas de los usuarios, sino también generar respuestas coherentes y relevantes. Complementando la PNL, el Aprendizaje Automático (*Machine Learning*) y, en particular, su subcampo el Aprendizaje Profundo (*Deep Learning*), son cruciales. Estas tecnologías capacitan a los sistemas para aprender y mejorar continuamente a partir de los datos de las interacciones, identificando patrones lingüísticos complejos y refinando su capacidad para ofrecer respuestas cada vez más precisas y naturales, brindando así soporte y servicios automatizados en línea (Mei et al., 2024).

Por último, analizaremos algunas de las ventajas que tienen los Chatbots a la hora de atender a clientes; Una ventaja clave es su disponibilidad 24/7, eliminando las limitaciones horarias y permitiendo a los clientes obtener asistencia inmediata (Bushra, 2024), Además, pueden manejar múltiples conversaciones en diferentes idiomas simultáneamente, asegurando respuestas eficientes incluso en períodos de alta demanda (Aslam et al., 2025)

2.2. Tendencias de Investigación en Chatbots

A pesar de que la investigación sobre chatbots en hostelería ha crecido significativamente desde 2023. Los hosteleros necesitan saber cómo se comparan los chatbots con los humanos en el proceso de reserva. Hasta donde sabemos, esto no se ha investigado hasta la fecha (Wüst & Bremser, 2025).

Las principales tendencias en la investigación sobre chatbots para reservas de restaurantes se centran en el creciente uso de la inteligencia artificial (IA) y el procesamiento del lenguaje natural (PNL) para desarrollar asistentes virtuales más sofisticados, capaces de comprender el lenguaje humano y personalizar las interacciones. Estos sistemas no solo están diseñados para imitar el diálogo humano o comprender a los usuarios, sino que también tienen muchas aplicaciones prácticas(Celi-Parraga et al., 2021). Otra tendencia de investigación son las tecnologías emergentes como el reconocimiento de voz y la interacción multimodal para mejorar la experiencia del usuario, mientras que también se abordan desafíos como la fatiga del usuario y la necesidad de una transición fluida a agentes humanos para consultas complejas.
El mercado de Chatbots para restaurantes está en expansión, impulsado por la búsqueda de eficiencia operativa y unos consumidores cada vez más exigentes (Davis, 2024). Esto dirige la investigación futura hacia la optimización del flujo conversacional, la adaptabilidad a diferentes tipos de restaurantes y la consideración de aspectos éticos como la transparencia y la privacidad.

En cuanto la investigación académica ha experimentado un notable aumento de interés por la IA que se traducen en un significativo aumento de las investigaciones sobre este tema. A pesar de ello la adopción generalizada de soluciones de IA sofisticadas en la industria de la hostelería todavía está en desarrollo. Muchas empresas aún están explorando cómo integrar eficazmente estas tecnologías.

Las investigaciones recientes también reflejan un cambio en su enfoque, que ha pasado de la previsión de la demanda y el análisis de sentimientos, al uso de bots de servicio y la aplicación de inteligencia artificial para mejorar la calidad del servicio, con un énfasis reciente en herramientas de IA generativa como ChatGPT (To & Yu, 2025).

2.3. ChatGPT y sus posibles aplicaciones en la hostelería

ChatGPT es un chatbot basado en inteligencia artificial lanzado en noviembre de 2022 por OpenAI. Su versatilidad lo ha convertido en una de las principales tendencias en hostelería (Lacalle, 2023). Basado en la arquitectura Generativa, ChatGPT se entrena con grandes cantidades de datos, lo que le permite comprender matices del lenguaje, responder preguntas, redactar contenido y mantener conversaciones. Su capacidad para procesar y generar lenguaje natural lo convierte en una herramienta potencialmente poderosa para aplicaciones de servicio al cliente, incluido el tema de este trabajo, la gestión de reservas en restaurantes.

Los beneficios de la aplicación de la inteligencia artificial en hostelería son numerosos entre ellos: una mayor eficiencia operativa, reducción de costes, mejora de la satisfacción y lealtad del cliente y la capacidad de tomar decisiones más informadas basadas en datos por parte de los hosteleros (Alawami et al., 2025).

2.4. ChatGPT como LLM, fortalezas y limitaciones.

Los Modelos de Lenguaje Grandes (LLMs) como los de la serie GPT (Generative Pre-trained Transformer), en la que se basa ChatGPT, representan un avance significativo en IA. Estos modelos se entrenan con cantidades masivas de datos textuales, lo que les permite aprender patrones lingüísticos complejos y generar texto notablemente fluido, coherente e informativo (Alipour et al., 2024). ChatGPT, en particular, destaca por su capacidad para mantener conversaciones, comprender el contexto, responder preguntas, traducir idiomas, resumir textos y realizar diversas tareas de procesamiento de lenguaje natural (Choi & Kim, 2024).

Modelos más recientes como GPT-4 han introducido mejoras como ventanas de contexto más amplias (permitiendo procesar conversaciones más largas) y habilidades conversacionales más afinadas, haciendo las interacciones más naturales y coherentes. Estas capacidades los hacen versátiles para una amplia gama de aplicaciones, incluyendo asistentes virtuales y creación de contenido (Alipour et al., 2024).

En cuanto a las fortalezas cabe destacar su fluidez y flexibilidad conversacional. Pueden adaptarse a diversos temas y estilos, generando respuestas que a menudo son indistinguibles de las escritas por humanos. Sin embargo, esta fortaleza viene acompañada de limitaciones importantes. Una de las más notorias es la tendencia a "alucinar", es decir, generar información que suena plausible, pero es objetivamente incorrecta o inventada.

Otras limitaciones incluyen la sensibilidad a la formulación exacta del prompt (pequeños cambios pueden alterar significativamente la respuesta), la posibilidad de heredar y amplificar sesgos presentes en los datos de entrenamiento, y dificultades para mantener la coherencia o seguir objetivos complejos a lo largo de interacciones muy largas.

2.5. La API de ChatGPT y function Calling.

OpenAI proporciona una Interfaz de Programación de Aplicaciones (API) que permite a los desarrolladores integrar las capacidades de ChatGPT en sus propias aplicaciones y servicios (Pavlo, 2024). Es un conjunto de reglas y herramientas que nos permitirá que diferentes software se comuniquen y se integren con con nuestro agente de inteligencia artificial.

Una capacidad clave ofrecida recientemente por ChatGPT es el "Function Calling" (llamada a funciones). Esta le da la capacidad a los chatbots de interactuar con otros sistemas e interpretar el lenguaje natural. Lo que permite a los modelos GPT responder a preguntas que de otro modo no podrían, como aquellas que requieren información en tiempo real o datos no incluidos en su conjunto de entrenamiento. En otras palabras, la llamada a funciones proporciona otra forma de enseñar a los modelos de IA a interactuar con el mundo externo (Ackerson, 2023).

2.6. Ejemplos y Enfoques de Sistemas de reservas de restaurantes basados en IA

Ya existen diversos sistemas de reserva online para restaurantes, algunos de los cuales incorporan elementos de IA. Las plataformas impulsadas por IA pueden ofrecer características avanzadas como procesos de reserva fluidos a través de interfaces conversacionales, actualizaciones de disponibilidad de mesas en tiempo real, recomendaciones personalizadas basadas en el historial o preferencias del cliente, y gestión predictiva de listas de espera.

Investigaciones académicas han explorado sistemas específicos, como AutoConcierge, que combina un LLM (GPT-3) para la comprensión del lenguaje natural con un sistema de razonamiento lógico (Answer Set Programming) para ofrecer recomendaciones de restaurantes basadas en preferencias detalladas del usuario obtenidas a través del diálogo (Zeng et al., 2023). Otro ejemplo es KITA, que utiliza un LLM para parsear consultas complejas y componer llamadas a APIs (como BookRestaurant) y consultas a bases de

conocimiento (Joshi et al., 2024) Estos enfoques demuestran la viabilidad de usar IA avanzada para tareas relacionadas con la búsqueda y reserva de restaurantes.

2.6.1. Funcionalidades Esenciales

Un sistema de reservas de restaurante basado en IA eficaz debe manejar un conjunto de funcionalidades esenciales. Estas incluyen: comprender las solicitudes de reserva del usuario (fecha, hora, número de comensales), verificar la disponibilidad en tiempo real (requiere integración), presentar opciones y confirmar la selección, recopilar detalles del cliente (nombre, contacto), y gestionar información específica como alergias, restricciones dietéticas, presencia de niños y preferencias de asiento o peticiones especiales. Además, el sistema debe ser capaz de gestionar reservas existentes, permitiendo modificaciones o cancelaciones y responder a preguntas frecuentes sobre el restaurante (Davis, 2024).

2.7. Síntesis y Oportunidad del TFM

La revisión del estado del arte revela un creciente interés y aplicación de la IA, ML y chatbots en la hostelería, con beneficios demostrados en eficiencia y experiencia del cliente. ChatGPT ofrecen capacidades conversacionales sin precedentes, y mecanismos como la API y el function calling permiten su integración en aplicaciones prácticas. Sin embargo, también existen desafíos como, la precisión y las alucinaciones, que requieren estrategias de mitigación cuidadosas.

Aunque existen sistemas de reserva con IA y trabajos de investigación sobre enfoques avanzados, parece haber una oportunidad para un TFM que se centre específicamente en proporcionar una guía práctica y detallada para construir un agente de reservas de restaurante utilizando específicamente la API de ChatGPT. Esta guía debe abordar no solo el flujo básico de reserva, sino también el manejo de requisitos complejos (alergias, preferencias múltiples) a través de técnicas como la ingeniería de prompts y el function calling, y considerar los desafíos prácticos y éticos de su implementación. La naturaleza flexible, pero a veces impredecible de ChatGPT. Hace que una guía detallada sobre cómo controlarlo y conectarlo a sistemas reales (a través de function calling) sea particularmente valiosa para cerrar la brecha entre el potencial de la tecnología y su aplicación efectiva en el dominio específico de las reservas de restaurantes.

3. Diseño del Agente de IA para Reservas de Restaurantes con ChatGPT

En el presente capítulo se abordará el diseño conceptual y funcional de un agente conversacional destinado a la gestión de reservas en el modelo de lenguaje ChatGPT. El desarrollo se estructura en dos fases principales: la primera consiste en la definición de los requisitos funcionales que el agente debe satisfacer, y la segunda, en el análisis de cómo las capacidades de ChatGPT y su API pueden ser aprovechadas para la obtención de dichos requisitos.

3.1. Definición de Requisitos Funcionales

La creación de un agente de reservas eficiente y robusto basado en ChatGPT, exige definir una serie de requisitos funcionales que englobar la totalidad del ciclo de vida de una reserva, desde la interacción inicial con el usuario pasando por la gestión de la misma y finalmente el tratamiento de la información particular del cliente.

3.1.1. Flujo Principal de Reserva

El proceso para la formalización de una reserva se desglosa en las siguientes etapas:

1. Identificación de la Intención de Reserva: El agente debe ser capaz de distinguir con fiabilidad la intención explícita o implícita del usuario de iniciar un proceso de reserva.

2. Recopilación de Parámetros Esenciales de la Reserva: El sistema debe solicitar y extraer de manera inequívoca la fecha, la hora y el número de comensales para los cuales se desea efectuar la reserva.

3. Verificación de Disponibilidad: Se requiere que el agente interactúe con un sistema externo (ej. base de datos, API de un sistema de gestión de restaurantes – RMS/POS) con el fin de constatar la existencia de mesas disponibles que se ajusten a los criterios proporcionados por el usuario. Esta etapa es de naturaleza crítica y supone la integración entre el agente y los diferentes sistemas de gestión del restaurante.

4. Presentación de Alternativas y/o Confirmación Provisional: En caso de existir disponibilidad, el agente deberá presentar las posibles opciones para reservar y el horario disponible. Si el agente encuentra que no hay disponibilidad ofrecerá otras opciones de reserva (ej. horarios o días próximos).

5. Obtención de Datos del Cliente: Una vez el cliente escoja el horario entre las disponibles, el agente procederá a solicitar y almacenar el nombre y la información de contacto (número de teléfono y/o dirección de correo electrónico) del titular de la reserva. Se debe aplicar especial atención e a la privacidad y protección de estos datos.

6. Confirmación Definitiva de la Reserva: Previo a la conclusión del proceso, el agente deberá recapitular la totalidad de los detalles de la reserva (fecha, hora, número de comensales, nombre y datos de contacto del cliente, así como cualquier anotación especial) y recibir una confirmación explícita por parte del usuario.

7. Emisión de Confirmación Formal: Tras la validación por el usuario, el agente deberá expedir una notificación de confirmación al cliente (ej. mediante SMS o correo electrónico), lo cual podría implicar una integración adicional con servicios de mensajería.

3.1.2. Gestión de Requerimientos Específicos

El agente debe tener la capacidad para atender y registrar necesidades particulares expresadas por los usuarios:

- Alergias y Restricciones Alimentarias: El agente deberá ser capaz de identificar, ya sea mediante pregunta activa o por reconocimiento de expresiones del usuario la presencia de alergias o restricciones dietéticas (ej. "alergia al cacahuete", "intolerancia al gluten", "vegetariano", "vegano"). Esta información deberá registrarse con precisión para su comunicación al personal de cocina y sala.

- Presencia de Niños: Se deberá registrar si la reserva incluye niños y, opcionalmente, capturar información relevante como la necesidad de sillas altas (tronas) o consultas sobre la disponibilidad de menús infantiles.

- Preferencias de Ubicación: El agente permitirá al usuario manifestar sus preferencias en cuanto a la ubicación de la mesa (ej. "cerca de la ventana", "en una zona tranquila", "terraza"), registrándolas como una solicitud sujeta a la disponibilidad del establecimiento.

- Ocasiones Especiales: El agente deberá posibilitar la anotación de circunstancias especiales asociadas a la reserva (ej. "celebración de cumpleaños", "aniversario"), permitiendo al restaurante considerar detalles o atenciones particulares.

- Solicitudes Particulares Adicionales: Se deberá contemplar la capacidad de registrar otras peticiones razonables y específicas (ej. "necesidad de acceso para silla de ruedas", "espacio para un cochecito de bebé").

3.1.3. Modificación y Cancelación de Reservas

El agente debe proveer mecanismos para la gestión de reservas existentes:

- Localización de Reservas Preexistentes: El sistema permitirá la recuperación de una reserva activa; mediante el nombre del titular, su número de teléfono o un identificador único de confirmación.

- Modificación de Parámetros de Reserva: Deberá facilitar a los usuarios la modificación de su reserva (ej. cambio de fecha, hora, o número de comensales), incluyendo la verificación de disponibilidad para la nueva solicitud y la actualización de la reserva si esta fuera posible.

- Cancelación de Reservas: Se permitirá a los usuarios solicitar la anulación de su reserva y dar confirmación de que dicha cancelación ha sido procesada exitosamente.

3.1.4. Suministro de Información General

El agente deberá estar capacitado para responder a consultas frecuentes relativas al establecimiento, tales como horarios de funcionamiento, dirección, opciones de estacionamiento, tipo de cocina ofrecida, y políticas internas (ej. código de vestimenta, admisión de mascotas), basándose en una base de conocimiento previa.

3.1.5. Tratamiento de Errores y Ambigüedades Conversacionales

Es imperativo que el agente gestione adecuadamente las situaciones anómalas o ambiguas:

- Solicitud de Clarificación: En escenarios donde la petición del usuario resulte ambigua o incompleta (ej. "quiero reservar para cenar" sin especificar un horario), el agente deberá solicitar la información adicional necesaria para resolve la consulta.

- Comunicación de Indisponibilidad: El sistema deberá informar de manera clara y cortés cuando una solicitud no pueda ser satisfecha (ej. horario completo), y gestionar la situación ofreciendo alternativas o, si la operativa del restaurante lo contempla, la opción de inscribirse en una lista de espera.

- Manejo de Fallos Sistémicos: El agente deberá gestionar los errores de naturaleza técnica (ej. fallo en la comunicación con la API del restaurante), informando al usuario de la incidencia y proponiendo, en la medida de lo posible, cursos de acción alternativos (ej. contactar telefónicamente con el restaurante, reintentar la operación más tarde, o escalar la interacción a un agente humano).

3.2. Correspondencia entre Requisitos y Capacidades de ChatGPT

Las funcionalidades delineadas en la sección anterior pueden ser implementadas mediante el aprovechamiento de las capacidades y características ofrecidas por la API de ChatGPT:

1. Comprensión del Lenguaje Natural (NLU): La aptitud central de ChatGPT para procesar e interpretar el lenguaje natural constituye el pilar para la comprensión de la variada gama de formulaciones que los usuarios emplean al expresar sus solicitudes, preguntas, preferencias y restricciones.

2. Gestión del Contexto Conversacional (messages): La API de ChatGPT facilita el envío del historial completo de la conversación en cada interacción mediante el *array messages*. Esta característica es fundamental para que el agente mantenga la coherencia contextual, recordando la información recopilada en turnos previos (como la fecha y hora ya acordadas) y sosteniendo un diálogo fluido durante los procesos de reserva, modificación o cancelación. La arquitectura de la API de OpenAI está diseñada para soportar esta funcionalidad.

3. Adherencia a Instrucciones y Directrices (system prompt): El mensaje inicial con el rol "system" permite configurar el comportamiento general del agente, incluyendo su propósito (agente de reservas), su tono comunicacional (ej. amable, eficiente), las normativas que debe seguir (ej. solicitar siempre confirmación, qué datos específicos recopilar) y sus limitaciones operativas (ej. no aceptar reservas para un número de comensales superior a X).

4. Extracción de Información Estructurada (Ingeniería de *Prompts*): Es factible diseñar *prompts* específicos para instruir a ChatGPT en la tarea de extraer información clave de las respuestas del usuario y devolverla en un formato estructurado, como JSON. Esta capacidad es crucial para la obtención fiable de datos como fechas, horas, número de comensales y detalles sobre alergias, para su posterior procesamiento por el sistema *backend* ("detrás de escena", es el motor lógico que procesa la información y se conecta con otros sistemas, pero que el usuario final no ve directamente). El empleo de técnicas como la especificación explícita del formato de salida deseado o la provisión de ejemplos (*few-shot prompting*) contribuye a mejorar la precisión de esta extracción.

5. Orquestación del Diálogo (*Prompting* y Lógica Externa): Aunque ChatGPT posee capacidades para mantener un flujo conversacional, la lógica principal que gobierna la gestión del diálogo (e.g., la secuencia de preguntas, el momento de verificar la disponibilidad, la solicitud de confirmaciones) reside en la aplicación *backend* que orquesta las llamadas a la API. Los *prompts* se utilizan para guiar a ChatGPT en cada etapa del diálogo, en función del estado actual de la conversación y los objetivos de la interacción. Se pueden emplear *prompts* para solicitar aclaraciones, confirmar información o presentar opciones al usuario.

6. Interacción con Sistemas Externos (*Function Calling*): Para la ejecución de tareas que requieren información o la realización de acciones que trascienden el conocimiento intrínseco del LLM (tales como la verificación de disponibilidad en tiempo real o la creación efectiva de la reserva en el sistema de gestión del restaurante), se recurre a la funcionalidad de *Function Calling*. El LLM identifica la necesidad de invocar una función predefinida por el desarrollador (ej. check_availability), y la aplicación *backend* se encarga de ejecutar la llamada a la API correspondiente del restaurante, devolviendo el resultado al LLM para que este, a su vez, informe al usuario.

Tabla 1: Resumen Mapeo de Requisitos Funcionales a Capacidades de ChatGP

Categoría de Requisito	Requisito Específico	Capacidad Clave de ChatGPT
Flujo Principal de Reserva	Interpretar solicitud inicial	NLU, Comprensión del Lenguaje Natural (entrada de mensaje)
	Recopilar Fecha, Hora, Comensales	NLU, Ingeniería de Prompts (Extracción)
	Verificar Disponibilidad	Function Calling (check_availability)
	Presentar Opciones / Confirmar	Generación de Lenguaje Natural (NLG), Gestión del Diálogo
	Recopilar Datos Cliente (Nombre, Contacto)	NLU, Ingeniería de Prompts (Extracción)
	Confirmar Detalles Reserva	NLG, Gestión de lenguaje natural (salida de mensaje)
	Finalizar Reserva (Acción Externa)	Function Calling (create_reservation)
Necesidades Específicas	Capturar Alergias / Restricciones	NLU, Ingeniería de Prompts (Extracción Específica)
	Registrar Niños	NLU, Ingeniería de Prompts (Extracción)
	Capturar Preferencias (Asiento, Ocasión)	NLU, Ingeniería de Prompts (Extracción)
	Manejar Peticiones Especiales	NLU, Ingeniería de Prompts
Gestión de Reservas	Recuperar Reserva Existente	Function Calling (get_booking_details)
	Modificar Reserva	NLU, Gestión del Diálogo, Function Calling (modify_reservation)
	Cancelar Reserva	NLU, Gestión del Diálogo, Function Calling (cancel_reservation)
Provisión de Información	Responder FAQs (Horario, Ubicación, etc.)	NLU, NLG (basado en contexto/prompt)
Manejo Errores	Solicitar Clarificación	Ingeniería de Prompts, Gestión del Diálogo
	Gestionar Indisponibilidad	Gestión del Diálogo, NLG
	Manejar Errores Técnicos	Lógica Backend (Fallback), NLG para informar

3.3. Arquitectura Conceptual del Sistema

Para la implementación del agente conversacional de reservas propuesto, se define una arquitectura conceptual multicomponente, diseñada para integrar las distintas tecnologías y flujos de datos necesarios para su operación. Esta arquitectura, cuya representación visual se detalla en la Figura 1, se fundamenta en la interacción coordinada de los siguientes elementos clave:

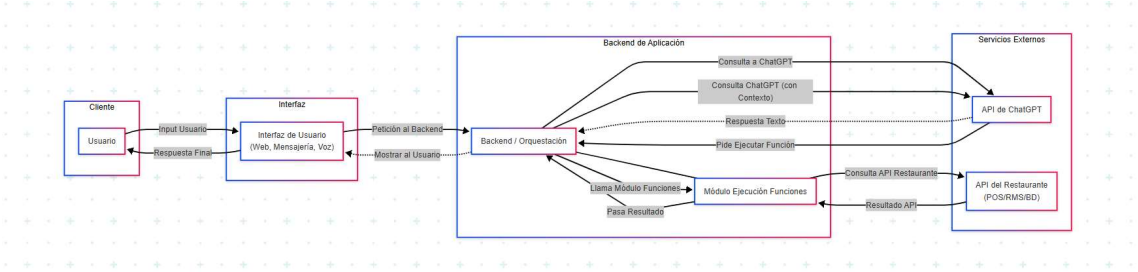

Figura 1: Diagrama de Arquitectura Conceptual

1. **Interfaz de Usuario (UI):** Constituye el canal de comunicación directo entre el cliente y el agente conversacional. Esta interfaz puede materializarse de diversas formas, tales como un *widget* de chat incorporado en el sitio web del restaurante, una integración con plataformas de mensajería instantánea (ej. WhatsApp, Facebook Messenger), o incluso mediante una interfaz basada en voz.

2. **Backend de Aplicación / Capa de Orquestación:** Actúa como el núcleo central del sistema, asumiendo la responsabilidad de la lógica de negocio y la coordinación general. Sus funciones principales incluyen: la gestión del flujo conversacional, el mantenimiento del estado del diálogo (información recopilada, etapa actual del proceso), la interacción programática con la API de ChatGPT, el procesamiento de las respuestas generadas por el modelo de lenguaje (LLM), la toma de decisiones sobre la invocación de funciones externas, y la comunicación bidireccional con la Interfaz de Usuario.

3. **API de ChatGPT:** Representa la interfaz de programación de aplicaciones (API) proporcionada por OpenAI, que facilita el acceso al modelo de lenguaje grande (LLM). Esta API recibe como entrada el historial de la conversación y las instrucciones contextuales (*prompts*), y devuelve como salida la respuesta generada por el LLM o, alternativamente, una solicitud estructurada para invocar una función externa predefinida.

4. **Módulo de Ejecución de Funciones:** Es un componente especializado dentro del *backend*, cuya finalidad es albergar y ejecutar la lógica de negocio asociada a las funciones específicas que ChatGPT puede solicitar (ej. check_availability, create_booking). Este módulo es el responsable último de interactuar con los sistemas internos del restaurante.

5. **API del Restaurante (POS/RMS/Base de Datos):** Corresponde a la interfaz programática que expone los datos y las funcionalidades del sistema de gestión propio del restaurante. A través de esta API, el Módulo de Ejecución de Funciones puede acceder a información crítica, como la disponibilidad de mesas en tiempo real, y ejecutar operaciones de creación, modificación o cancelación de reservas directamente en el sistema de origen.

Flujo Típico de Información

El proceso de interacción estándar dentro de esta arquitectura sigue una secuencia definida:

1. El usuario inicia la interacción enviando un mensaje a través de la UI.

2. La UI transmite dicho mensaje al Backend de Aplicación.

3. El Backend incorpora el mensaje del usuario al historial conversacional y prepara la solicitud para la API de ChatGPT. Esta solicitud incluye el historial completo, el *prompt* de sistema (que define el comportamiento del agente) y las definiciones de las funciones externas pertinentes para el contexto actual.

4. El Backend cursa la solicitud a la API de ChatGPT.

5. La API de ChatGPT procesa la entrada y genera una respuesta, que puede adoptar una de las siguientes formas:

    - Opción A (Respuesta Textual): Devuelve un mensaje en lenguaje natural generado por el LLM (e.g., una pregunta para aclarar una ambigüedad, un mensaje de confirmación).

    - Opción B (Solicitud de Invocación de Función): Retorna una estructura de datos (generalmente JSON) que especifica la necesidad de invocar una función externa concreta, junto con los argumentos requeridos (ej. check_availability con los parámetros fecha, hora y comensales).

6. El Backend recibe y procesa la respuesta de ChatGPT:

    - Si es Opción A: El mensaje textual se envía a la UI para ser mostrado al usuario. El Backend actualiza el estado interno de la conversación.

    - Si es Opción B:
        - El Backend invoca al Módulo de Ejecución de Funciones, pasándole los detalles de la función a ejecutar.

        - El Módulo de Ejecución interactúa con la API del Restaurante (ej. para consultar la disponibilidad).

- La API del Restaurante retorna el resultado de la operación (ej. lista de horarios disponibles, código de confirmación de reserva).
- El Módulo de Ejecución devuelve este resultado al Backend.
- El Backend añade tanto la solicitud de función original como su resultado al historial de la conversación.
- El Backend realiza una segunda llamada a la API de ChatGPT, esta vez con el historial actualizado, para que el LLM genere una respuesta en lenguaje natural basada en el resultado de la función ejecutada (ej. "Sí, disponemos de una mesa a las 20:00 h. ¿Desea que la confirme?").
- El Backend recibe esta nueva respuesta textual y la envía a la UI, actualizando nuevamente el estado de la conversación.

7. Este ciclo iterativo se repite con cada nueva entrada del usuario hasta la finalización de la interacción.

Este diseño arquitectónico promueve una separación clara de responsabilidades, disociando la inteligencia conversacional (residente en ChatGPT) de la lógica de negocio y el acceso a los datos específicos del restaurante (manejados por el Backend y la API del Restaurante). El mecanismo de *function calling* actúa como el puente esencial que conecta y posibilita la colaboración entre estos dos dominios.

4. Guía Práctica de Implementación

Este capítulo es el corazón del TFM, donde se detallan los pasos prácticos seguidos y propuestos para implementar el agente de reservas de restaurante utilizando la API de ChatGPT. El foco estará en Python como lenguaje de programación, dada su popularidad en el desarrollo de aplicaciones de IA y la disponibilidad de librerías robustas como la oficial de OpenAI.

4.1. Configuración Inicial

Antes de escribir la primera línea de código para el agente, es necesario realizar una configuración inicial del entorno y obtener las credenciales necesarias.

4.1.1. Obtención y Gestión Segura de Claves API (OpenAI)

El primer paso ineludible es obtener una clave API de OpenAI. Esta clave es la que autenticará las solicitudes de la aplicación a sus servicios.

- **Crear Cuenta/Iniciar Sesión:** Para ello, se visitó el sitio web oficial de OpenAI (platform.openai.com). Allí, se creó una cuenta (o se inició sesión, si ya se disponía de una). Este proceso requiere una verificación por correo electrónico.

- **Navegar a Claves API:** Una vez dentro del panel de control de OpenAI, se localizó la sección de (Claves API) que generalmente se encuentra en un menú lateral.

- **Generar Nueva Clave:** Se hizo clic en el botón destinado a crear una nueva clave secreta (Crear nueva clave secreta).

- **Copiar y Guardar de Forma Segura:** Es crucial destacar que la clave API se muestra una sola vez al generarse. Por tanto, se copió inmediatamente y se guardó en un lugar seguro y privado. Nunca se debe incrustar la clave API directamente en el código fuente de la aplicación. Para gestionarla de forma segura, se han considerado y se recomiendan los siguientes métodos:

    - **Variables de Entorno:** Almacenar la clave como una variable de entorno del sistema operativo. Durante el desarrollo, esta es una práctica común.

    - **Archivos .env:** Crear un archivo llamado .env en la raíz del proyecto. Dentro de este archivo, se añadiría una línea como: `OPENAI_API_KEY="tu_clave_aqui"`. Es fundamental asegurarse de añadir el archivo .env al archivo .gitignore del proyecto para evitar subir accidentalmente la clave a repositorios de código públicos. Para cargar esta variable en la aplicación Python, se utilizará una librería como python-dotenv.

- **Gestores de Secretos:** Para entornos de producción más robustos, lo ideal sería utilizar servicios especializados en la gestión de secretos, como AWS Secrets Manager, Azure Key Vault o Google Cloud Secret Manager.

Aunque esto es recomendable en las pruebas realizadas para la implementación de esta guía no fue necesaria.

4.1.2. Preparación del Entorno de Desarrollo (Python, venv, librerías)

Para mantener un entorno de desarrollo limpio y aislado, se recomienda encarecidamente utilizar un entorno virtual para cada proyecto.

- **Instalar Python:** Es necesario asegurarse de tener una versión reciente de Python instalada en el sistema (la documentación de OpenAI suele recomendar la versión 3.7 o superior). Python se puede descargar desde su sitio web oficial, python.org. Se verificó la instalación ejecutando python --version en la terminal.

- **Crear Entorno Virtual:** Desde la terminal, se navegó al directorio del proyecto y se ejecutó el siguiente comando para crear un entorno virtual llamado venv:
  `Python -m venv venv`

  Esto crea un subdirectorio venv dentro del proyecto.

- **Activar Entorno Virtual:** Para empezar a usarlo, se activó con:

  - En Windows: `venv\Scripts\activate`

  - En macOS/Linux: `source venv/bin/activate`

Tras la activación, el prompt de la terminal suele mostrar (venv) al principio, indicando que el entorno virtual está activo.

- **Instalar Librerías Necesarias:** Con el entorno virtual activo, se procedió a instalar las librerías necesarias utilizando pip, el gestor de paquetes de Python:
  `pip install openai python-dotenv requests`

  Las librerías clave son:

  - openai: Es la librería oficial de OpenAI para interactuar con su API. Es importante asegurarse de instalar una versión reciente que soporte las últimas características, como el function calling que es esencial para el proyecto.

  - python-dotenv: Esta librería permitirá cargar fácilmente la clave API (y otras configuraciones) desde el archivo .env mencionado antes.

    - requests: Aunque la librería openai maneja las llamadas a su API, requests es una librería estándar muy útil para realizar otras llamadas HTTP, por ejemplo, si se necesita que las funciones (function calling) interactúen con otras APIs externas (como la del sistema POS/RMS del restaurante).

- **Verificar Instalación:** Para confirmar que todo se instaló correctamente, se abrió un intérprete de Python (escribiendo python en la terminal activada) y se intentó importar las librerías recién instaladas (p. ej., `import openai, import dotenv, import requests`). Si no aparecen errores, la instalación fue exitosa.

4.1.3. Selección del Modelo ChatGPT

OpenAI ofrece varios modelos a través de su API, principalmente de las familias GPT-3.5 y GPT-4. La elección del modelo es una decisión importante que afectará tanto al coste como a las capacidades del agente.

- **gpt-3.5-turbo:** Según la investigación realizada, este modelo es generalmente más rápido y económico. Es adecuado para muchas tareas de chatbot y extracción de datos, y podría ser un buen punto de partida.

- **gpt-4 / gpt-4o:** Estos son los modelos más capaces que ofrece OpenAI actualmente. Tienen un mejor razonamiento, siguen instrucciones complejas con mayor precisión y, a menudo, ofrecen un mejor rendimiento en tareas como el function calling y la generación de respuestas más matizadas. Sin embargo, su coste por token (la unidad de medida del texto procesado) es más elevado. gpt-4o es el modelo más reciente en el momento de redactar esto, optimizado para ofrecer alta capacidad con mejor velocidad y coste.

La estrategia será comenzar el desarrollo y las pruebas iniciales con un modelo más económico como gpt-3.5-turbo o el balanceado gpt-4o, y evaluar su rendimiento. Si las tareas de extracción de información detallada (como alergias complejas) o la gestión de diálogos muy matizados resultan ser un desafío para este modelo, se consideraría pasar a un modelo superior de la familia GPT-4, asumiendo el incremento en el coste. La elección final dependerá del equilibrio entre el presupuesto disponible y la complejidad de las funcionalidades que el agente de reservas necesite implementar.

4.2. Interacción Básica con la API de ChatGPT

Una vez configurado el entorno, el siguiente paso es establecer la comunicación básica con la API de ChatGPT.

4.2.1. Estructura de Llamadas API (Python)

Utilizando la librería openai en Python, una llamada típica para obtener una respuesta del modelo de chat se estructura de la siguiente manera. Este es un ejemplo fundamental adaptado para el proyecto.

```
import os
import openai
from dotenv import load_dotenv
import json # Importante para manejar argumentos de funciones

# Cargar la clave API desde el archivo .env
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")

if not api_key:
    raise ValueError("No se encontró la clave API de OpenAI. Asegúrate de que está configurada
en el archivo .env o como variable de entorno.")

# Inicializar el cliente de OpenAI
# Para versiones más antiguas de la librería openai podría ser: openai.api_key = api_key
# Para versiones v1.0.0+ de la librería openai:
client = openai.OpenAI(api_key=api_key)

# Definir el modelo a utilizar (ej. "gpt-4o", "gpt-3.5-turbo")
MODELO_CHATGPT = "gpt-4o"

# Lista de mensajes que representa el historial de la conversación
# El primer mensaje suele ser el 'system prompt'
mensajes = [
    {"role": "system", "content": "Eres un asistente virtual muy útil especializado en
reservas de restaurantes."},
    {"role": "user", "content": "Hola, ¿podrías ayudarme a reservar una mesa?"}
]
try:
    # Realizar la llamada a la API de Chat Completions
    respuesta_api = client.chat.completions.create(
        model=MODELO_CHATGPT,
        messages=mensajes
        # Aquí se pueden añadir otros parámetros opcionales como:
        # temperature=0.7, (controla la aleatoriedad, más bajo es más determinista)
        # max_tokens=150, (límite de tokens en la respuesta)
        # tools=[...], (definición de funciones para function calling)
        # tool_choice="auto", (o forzar una función específica)
    )
    # Extraer el contenido de la respuesta del asistente
    # La estructura de la respuesta puede variar ligeramente, es bueno inspeccionarla
    mensaje_del_asistente = respuesta_api.choices[0].message


    if mensaje_del_asistente.content:
        contenido_respuesta = mensaje_del_asistente.content
        print("Respuesta del Asistente:", contenido_respuesta)
        # Aquí se añadiría la respuesta del asistente al historial 'mensajes'
        mensajes.append({"role": "assistant", "content": contenido_respuesta})
```

```
    # Manejo de 'function calling' (se detallará más adelante)
    elif mensaje_del_asistente.tool_calls:
        print("El asistente solicitó llamar a una función.")
        # Lógica para manejar tool_calls (ver sección 4.5)
        pass

except openai.APIError as e:
    # Manejar errores específicos de la API de OpenAI
    print(f"Error de la API de OpenAI: {e.status_code} - {e.message}")
except Exception as e:
    # Manejar otros errores inesperados
    print(f"Ocurrió un error inesperado: {e}")
```

Se han incluido comentarios en el código para explicar cada parte. Este es el esqueleto básico sobre el cual se construirán interacciones más complejas.

4.2.2. Gestión del Historial de Conversación (messages)

El parámetro messages es, en opinión de este estudio, uno de los más importantes para lograr una conversación coherente. Es una lista de diccionarios, donde cada diccionario representa un turno en la conversación y debe contener, como mínimo, las claves rolen y content. Los roles que se utilizarán principalmente son:

- system: Para el mensaje inicial que define el comportamiento y las instrucciones generales del agente.
- user: Para los mensajes enviados por el usuario final al agente.
- assistant: Para las respuestas generadas previamente por el modelo (el agente).
- tool: (Se verá en function calling) Para proporcionar los resultados de las llamadas a funciones que el modelo haya solicitado.

Es fundamental que la aplicación backend envíe la lista messages completa y actualizada en cada llamada a la API. Así es como el modelo "recuerda" el contexto de la conversación. Después de recibir una respuesta del asistente (o después de una llamada a función), la lógica de backend se encargará de añadir el último mensaje del usuario y la respuesta/resultado correspondiente a la lista messages antes de la siguiente interacción.

```
# Ejemplo de cómo se actualiza el historial 'mensajes'

# Cuando el usuario envía un nuevo mensaje:
mensaje_usuario_actual = "Perfecto, quiero reservar para 2 personas mañana a
las 8 PM."
mensajes.append({"role": "user", "content": mensaje_usuario_actual})
```

```
# ... luego se realiza la llamada a la API con la lista 'mensajes'
actualizada ...
# respuesta_api = client.chat.completions.create(...)

# Cuando se recibe una respuesta de texto del asistente:
# contenido_respuesta_asistente = respuesta_api.choices[0].message.content
# if contenido_respuesta_asistente:
# mensajes.append({"role": "assistant", "content":
contenido_respuesta_asistente})

# (El manejo de tool_calls y sus resultados en 'mensajes' se detallará en la
sección 4.5)

# La lista 'mensajes' ahora contiene el historial actualizado, lista para la
siguiente interacción.
```

4.2.3. Definición del Comportamiento del Agente (system role)

El primer mensaje en la lista messages, aquel con role: "system", es la oportunidad para darle al agente sus instrucciones fundamentales. Este prompt debe ser lo más claro y específico posible para guiar el comportamiento deseado. Se ha elaborado un ejemplo inicial para el agente de reservas, que se irá refinando:

```
system_prompt_reservas = (
    "Eres 'ReservaPro', un asistente virtual experto en la gestión de
reservas para el restaurante 'El Buen Sabor'."
    "Tu principal objetivo es ayudar a los usuarios a crear, modificar o
cancelar reservas de mesa de manera eficiente y amable. "
    "Siempre debes ser cortés, paciente y profesional en tus interacciones. "
    "Para una NUEVA RESERVA, debes obtener la siguiente información del
usuario: "
    "1. Fecha de la reserva. "
    "2. Hora de la reserva. "
    "3. Número de comensales. "
    "Una vez tengas estos datos básicos, debes verificar la disponibilidad
ANTES de continuar. "
    "Si hay disponibilidad, o si el usuario confirma una opción que le
ofreces, entonces debes solicitar: "
    "4. Nombre completo de la persona a cargo de la reserva. "
    "5. Un número de teléfono de contacto. "
    "6. Pregunta explícitamente si hay alguna alergia, restricción dietética
(ej. vegetariano, celíaco), o preferencia especial (ej. mesa junto a la
ventana, zona tranquila, celebración de cumpleaños, necesidad de trona para
bebé). "
```

```
    "Antes de finalizar CUALQUIER operación (nueva reserva, modificación),
resume TODOS los detalles y pide confirmación explícita al usuario. "
    "Si no hay disponibilidad para la solicitud inicial, informa al usuario
y, si es posible, sugiere horarios o días alternativos cercanos. "
    "El restaurante no acepta reservas para más de 8 personas a través tuyo;
para grupos más grandes, indica amablemente que deben llamar directamente al
restaurante al [Número de Teléfono del Restaurante]. "
    "Para MODIFICAR o CANCELAR una reserva, primero necesitarás identificar
la reserva existente. Pregunta por el nombre bajo el cual se hizo la reserva
y el número de teléfono, o un ID de reserva si lo tuvieran."
)
mensajes = [
   {"role": "system", "content": system_prompt_reservas}
    # ... aquí comenzaría la conversación con el primer mensaje del usuario
]
```

Este system prompt establece la personalidad, el objetivo, la información necesaria, el flujo básico y algunas reglas de negocio.

4.2.4. Procesamiento de Respuestas

La respuesta de la API de OpenAI, como se ha visto en las pruebas, viene en formato JSON. El contenido del mensaje del asistente (si es una respuesta de texto) se encuentra típicamente en `respuesta_api.choices[0].message.content`. Es importante que el código extraiga este texto para mostrarlo al usuario y, como se mencionó, lo añada al historial messages con el rol assistant.

También es crucial implementar un manejo de errores robusto. Las llamadas a la API pueden fallar por diversas razones (problemas de red, errores en la solicitud, cuotas de API excedidas, etc.). El código incluirá bloques try-except para capturar estas excepciones y manejarlas de la forma más adecuada, informando al usuario si es necesario o reintentando la operación si tiene sentido.

4.3. Ingeniería de Prompts para Tareas de Reserva

La ingeniería de prompts es, según la experiencia acumulada hasta ahora, más un arte que una ciencia exacta, pero fundamental para obtener los resultados deseados del LLM.47 Se trata de diseñar cuidadosamente las entradas (prompts) para guiar al modelo.

4.3.1. Principios Clave Que Aplicar

Basándose en la documentación y en pruebas, se han identificado varios principios clave para una buena ingeniería de prompts:

- **Claridad y Especificidad:** Se buscará ser lo más directo y detallado posible en las instrucciones al modelo. Se evitará la ambigüedad.

- **Contexto Suficiente:** Se proporcionará toda la información relevante para que el modelo entienda la tarea y la situación. El historial en messages es la forma principal de dar contexto conversacional.

- **Instrucciones Claras y Concisas:** Se utilizarán verbos imperativos (ej. "Extrae...", "Resume...", "Pregunta...") y, si es necesario, se separarán claramente las instrucciones del texto que el modelo debe procesar, usando delimitadores como ### o comillas triples """ si el texto de entrada es complejo.

- **Especificar el Formato de Salida:** Cuando se necesite que el modelo devuelva información estructurada (especialmente para function calling o para la lógica interna), se le indicará explícitamente el formato deseado (pj. "Responde únicamente en formato JSON con las claves 'fecha', 'hora'...", "Devuelve solo el número entero").

- **Iteración y Pruebas:** La ingeniería de prompts es un proceso iterativo. Se comenzará con prompts simples (lo que se llama "zero-shot", es decir, sin ejemplos previos) y, si los resultados no son los esperados, se añadirá complejidad gradualmente: se pueden proporcionar ejemplos de cómo se desea la salida ("few-shot prompting"), o desglosar la tarea en instrucciones paso a paso. Se probarán exhaustivamente los prompts con diferentes entradas de usuario.

4.3.2. Técnicas para Extracción de Datos Específicos

Una de las tareas cruciales del agente será extraer información específica de las respuestas del usuario. Para esto, se diseñarán prompts que el backend utilizará internamente (no serán prompts que el usuario vea directamente) para procesar la entrada del usuario y obtener datos estructurados. Aquí algunos ejemplos conceptuales a desarrollar:

- **Extraer Fecha y Hora:**

    - Prompt de ejemplo:

```
 `Del siguiente texto del usuario, extrae la fecha y la hora
solicitadas para la reserva. Intenta normalizar la fecha al formato
YYYY-MM-DD y la hora al formato HH:MM (24h). Si el usuario dice
"mañana", considera que la fecha de hoy es {fecha_actual}. Si falta
alguna parte o es ambiguo, responde con "INCOMPLETO". Texto del
usuario: """{texto_del_usuario_previamente_validado_o_aclarado}"""
Fecha extraída: Hora extraída:`
```

- **Extraer Número de Comensales:**
    - Prompt de ejemplo:

```
`Analiza el siguiente texto e identifica el número de personas para la reserva. Responde solo con el número entero. Si no se especifica claramente, responde con "NO_ESPECIFICADO". Texto: """{texto_del_usuario}""" Número de comensales:`
```

- **Extraer Alergias, Restricciones y Preferencias (en formato JSON):**
    - Prompt de ejemplo:

```
`Del siguiente fragmento de conversación, extrae cualquier mención de alergias, restricciones dietéticas (vegetariano, vegano, sin gluten, etc.) o preferencias especiales para una reserva de restaurante. Formatea tu salida estrictamente como un objeto JSON con las siguientes claves: "alergias" (debe ser una lista de strings), "restricciones_dieteticas" (lista de strings), y "preferencias_adicionales" (lista de strings). Si no se menciona nada para una clave, utiliza una lista vacía []. No incluyas ningún otro texto antes o después del JSON. Texto del usuario: """{texto_del_usuario_sobre_necesidades_especiales}""" JSON:`
```

Esta técnica de pedir JSON es muy potente para function calling también.

4.3.3. Manejo de Ambigüedad y Solicitud de Clarificaciones

Es vital que el agente pueda manejar situaciones donde la información proporcionada por el usuario sea ambigua o incompleta. Se incluirán instrucciones en el system prompt y también se podrán generar prompts específicos contextualmente para que el modelo solicite aclaraciones.

- Parte del system prompt: `Si la solicitud del usuario es incompleta o ambigua (por ejemplo, si dice "quiero cenar" sin especificar la hora, o "una mesa" sin decir para cuántas personas), debes hacer una pregunta clara y amable para obtener la información que falta antes de proceder con la verificación de disponibilidad o cualquier otra acción.

4.3.4. Uso de Ejemplos (Few-Shot Prompting)

Si se observa que el modelo tiene dificultades para extraer información o formatearla de la manera necesaria con instrucciones directas (zero-shot), se recurrirá al "few-shot prompting". Esto implica proporcionar al modelo uno o más ejemplos completos de la tarea que se quiere que realice (entrada y salida deseada). Esto suele mejorar significativamente la fiabilidad.

- Ejemplo para extracción JSON con few-shot:

```
System: Extrae alergias, restricciones y preferencias del texto del
usuario y formatea la salida como un objeto JSON con las claves
"alergias", "restricciones_dieteticas", y "preferencias_adicionales".

User: Hola, quiero una mesa para 2 mañana a las 7. Uno de nosotros es
alérgico a los frutos secos y preferiríamos una mesa tranquila, si es
posible.
Assistant: {"alergias": ["frutos secos"], "restricciones_dieteticas": [],
"preferencias_adicionales": ["mesa tranquila"]}

User: Necesito reservar para 4 personas el sábado por la noche, sobre las
9. Somos dos vegetarianos en el grupo.
Assistant: {"alergias": [], "restricciones_dieteticas": ["vegetariano"],
"preferencias_adicionales": []}

User: {texto_del_usuario_actual_a_procesar}
Assistant: (Aquí se espera que el modelo genere el JSON correspondiente)

Este tipo de ejemplos se incluirían en el historial (messages) enviado a
la API.
```

4.3.5. Descomposición de Tareas Complejas (Prompt Chaining-Conceptual)

Para un proceso tan complejo como una reserva completa, que implica múltiples intercambios de información, validaciones y confirmaciones, en lugar de intentar manejarlo todo con un único prompt gigantesco (lo cual sería ineficaz y propenso a errores), el backend gestionará una secuencia de interacciones con el LLM. Esto se conoce a veces como "prompt chaining" o, más bien, una gestión de diálogo por estados donde cada estado puede tener un prompt específico.

1. **Prompt Inicial (Usuario -> ChatGPT):** Recibir la solicitud inicial del usuario. La respuesta de ChatGPT, guiada por el system prompt, podría ser una bienvenida y la solicitud de los primeros datos (ej. fecha, hora, comensales).

2. **Prompt de Extracción (Backend -> ChatGPT, internamente):** Una vez que el usuario proporciona los detalles básicos, el backend podría usar un prompt específico (no visible para el usuario) para instruir a ChatGPT a extraer esa información en un formato estructurado.

3. **Llamada a Función (Backend -> API Restaurante):** Con los datos estructurados, el backend llamaría a la función check_availability.

4. **Prompt de Presentación de Resultados (Backend -> ChatGPT -> Usuario):** Basándose en el resultado de check_availability, el backend construiría un prompt para que ChatGPT presente las opciones disponibles (o la falta de ellas) al usuario de forma natural.

5. Y así sucesivamente, guiando la conversación paso a paso.

4.3.6. Técnicas de Prompt Engineering para Reservas a Aplicar

Se ha creado una tabla para resumir las técnicas de ingeniería de prompts que se consideran más relevantes para el proyecto de agente de reservas:

Tabla 2: Técnicas de Prompt Engineering Aplicadas a Reservas de Restaurante

Técnica	Descripción	Ejemplo de Prompt Snippet (Conceptual)	Caso de Uso
Instrucción Clara	Ser directo y específico sobre la tarea requerida.	Extrae la fecha de reserva del siguiente texto: "{texto}". Responde solo en formato YYYY-MM-DD.	Extracción precisa de datos individuales.
Especificar Rol	Definir la personalidad y función del AI (usualmente en system prompt).	Eres un asistente de reservas amable y eficiente para 'La Trattoria'.	Establecer el tono y comportamiento general del agente.
Especificar Formato	Indicar explícitamente cómo debe ser la salida (JSON, lista, número).	... Formatea la salida como un objeto JSON con claves "alergias" y "preferencias".	Obtener datos estructurados para procesamiento backend.
Usar Delimitadores	Separar instrucciones del contexto o texto a procesar.	Analiza el texto del cliente entre ###. Texto:	Mejorar la claridad para el modelo, especialmente con entradas complejas.

		###{texto_cliente}###. Extrae la hora:	
Few-Shot Prompting	Proporcionar uno o más ejemplos de entrada/salida deseada.	Texto: "Mesa para 3, alergia al gluten." JSON: {"alergias": ["gluten"]} ### Texto: "{texto_actual}" JSON:	Guiar al modelo hacia un formato o estilo específico, mejorar precisión.
Solicitar Clarificación	Instruir al AI para que pida más información si la entrada es ambigua.	Si el usuario no especifica el número de personas, pregúntale "¿Para cuántas personas sería la reserva?"	Manejar entradas incompletas o vagas del usuario.
Chain-of-Thought (CoT)	Pedir al modelo que razone paso a paso (más útil para problemas complejos que extracción directa).	Primero, identifica la fecha. Luego, identifica la hora. Finalmente, identifica el número de comensales. Responde con: Fecha={fecha}, Hora={hora}, Comensales={num}	Descomponer tareas de extracción complejas o flujos de decisión.
Prompt Chaining	Usar múltiples llamadas a la API, donde la salida de una es entrada para la siguiente (gestionado por backend).	No es un prompt único, sino una secuencia de llamadas API gestionada por el backend.	Manejar procesos de reserva complejos con múltiples pasos de validación.

4.4. Estrategias de Gestión del Diálogo

Una gestión eficaz del diálogo es, a entender de este estudio, crucial para guiar al usuario a través del proceso de reserva de una manera lógica, fluida y satisfactoria. No se trata solo de que el LLM entienda y responda, sino de orquestar toda la interacción.

4.4.1. Seguimiento del Estado de la Conversación (Lógica del Backend)

El componente central de la estrategia de gestión del diálogo será el seguimiento del estado de la conversación. Esto no lo hará ChatGPT directamente; será una responsabilidad de la lógica de la aplicación backend. Para cada usuario interactuando con el agente, el backend mantendrá un registro del estado actual de su proceso de reserva. Se ha definido una serie de estados posibles que ayudarán a estructurar el flujo:

- ESPERANDO_INTENCION: Estado inicial, esperando que el usuario exprese su deseo de reservar, modificar, cancelar o preguntar algo.

- RECOPILANDO_INFO_BASICA: El agente está solicitando o esperando la fecha, hora y número de comensales para una nueva reserva.

- VERIFICANDO_DISPONIBILIDAD: El agente ha recibido la información básica y está en proceso de llamar a la función check_availability (esperando el resultado de la API del restaurante).

- PRESENTANDO_OPCIONES: Se ha verificado la disponibilidad (o no) y el agente está presentando opciones al usuario o informando sobre la falta de disponibilidad.

- RECOPILANDO_DETALLES_CLIENTE: Se ha acordado un horario y el agente está solicitando nombre, contacto y necesidades especiales (alergias, preferencias).

- PENDIENTE_CONFIRMACION_FINAL: Se han recopilado todos los datos y el agente ha presentado un resumen, esperando la confirmación explícita del usuario para proceder.

- CONFIRMANDO_RESERVA_EN_SISTEMA: El usuario ha confirmado, y el agente está en proceso de llamar a la función create_reservation.

- RESERVA_COMPLETADA: El proceso de reserva ha finalizado con éxito y se ha informado al usuario.

- IDENTIFICANDO_RESERVA_EXISTENTE: El usuario quiere modificar o cancelar, y el agente está solicitando datos para encontrar la reserva.

- GESTIONANDO_MODIFICACION: Se ha identificado una reserva y se están procesando los cambios solicitados (implica verificar nueva disponibilidad, etc.).
- GESTIONANDO_CANCELACION: Se ha identificado una reserva y se está procesando la solicitud de cancelación.

- RESPONDIENDO_FAQ: El agente está proporcionando información general sobre el restaurante.

- MANEJANDO_ERROR: Ha ocurrido un error (técnico o de comprensión) que necesita ser gestionado.

El estado actual de la conversación determinará qué prompts específicos se enviarán a ChatGPT, qué funciones (tools) estarán disponibles para que el LLM las llame, y qué acciones tomará el backend (pj. si en estado RECOPILANDO_INFO_BASICA el LLM extrae los datos, el backend cambiará el estado a VERIFICANDO_DISPONIBILIDAD y preparará la llamada a la función check_availability).

4.4.2. Gestión de Turnos y Flujo Conversacional

El backend será el director de orquesta de los turnos de la conversación. Utilizará el estado actual y la respuesta de ChatGPT (ya sea un mensaje de texto o una solicitud de tool_call) o el resultado de una función ejecutada para decidir el siguiente paso lógico:

- **Ejemplo de flujo:**

1. Usuario (estado ESPERANDO_INTENCION): "Quiero reservar para cenar."
2. Backend actualiza messages, llama a ChatGPT.
3. ChatGPT (respuesta de texto): "¡Claro! ¿Para qué fecha, hora y cuántas personas sería?" (Backend cambia estado a RECOPILANDO_INFO_BASICA).
4. Usuario: "Para mañana a las 9, para 2."
5. Backend actualiza messages. Llama a ChatGPT con la tool check_availability definida.
6. ChatGPT (respuesta tool_calls): Solicita llamar a check_availability con los argumentos extraídos `(fecha="YYYY-MM-DD de mañana", hora="21:00", party_size=2). (Backend cambia estado a VERIFICANDO_DISPONIBILIDAD)`.
7. Backend ejecuta mi_funcion_check_availability() que llama a la API del restaurante. Supongamos que devuelve: {"disponible": true, "horarios_confirmados": ["21:00"]}.
8. Backend actualiza messages con el resultado de la tool. Llama a ChatGPT de nuevo para que formule una respuesta basada en este resultado.
9. ChatGPT (respuesta de texto): "¡Perfecto! Mañana a las 21:00 para 2 personas está disponible. ¿Le gustaría que proceda a solicitar su nombre y teléfono para la reserva?" `(Backend cambia estado a RECOPILANDO_DETALLES_CLIENTE)`.

Este ciclo de [Usuario -> Backend -> ChatGPT -> (Función si es necesario) -> Backend -> ChatGPT -> Usuario] se repetirá, con el backend gestionando los estados y la lógica de transición.

4.4.3. Manejo de Modificaciones y Cancelaciones

Estos flujos son un poco más complejos porque primero requieren identificar una reserva existente. El proceso diseñado es:

1. **Reconocer Intención:** El NLU de ChatGPT, guiado por el system prompt, deberá reconocer si el usuario quiere modificar o cancelar.

2. **Solicitar Identificación:** El agente pedirá al usuario los detalles necesarios para encontrar su reserva (p. ej., nombre y teléfono, o un ID de reserva).

3. **Recuperar Detalles de Reserva:** El backend usará una función específica (p. ej., get_booking_details) que llamará a la API del restaurante para obtener los detalles de la reserva encontrada.

4. **Confirmar Reserva Correcta:** El agente (ChatGPT) presentará los detalles de la reserva encontrada al usuario y le pedirá que confirme que es la correcta antes de proceder.

5. **Para Modificación:**

    - Se recopilarán los nuevos detalles deseados (nueva fecha, hora, número de comensales).

    - Se verificará la disponibilidad para la nueva solicitud (usando check_availability).

    - Si hay disponibilidad y el usuario confirma, se llamará a una función modify_reservation.

6. **Para Cancelación:**

    - Tras la confirmación, se llamará a una función cancel_reservation.

7. **Confirmar Resultado:** El agente informará al usuario (usando ChatGPT) si la modificación o cancelación se realizó con éxito o si hubo algún problema.

4.4.4. Presentación Clara de Opciones Disponibles

Si la función check_availability devuelve múltiples horarios o quizás diferentes tipos de mesa disponibles (pj. "interior" o "terraza"), el backend deberá formatear esta información de manera estructurada. Luego, enviará esta información a ChatGPT con un prompt específico que le instruya a presentar estas opciones al usuario de forma clara y comprensible, y a solicitarle que elija una.

- Prompt de ejemplo para el LLM:

> La verificación de disponibilidad para el {fecha} a las {hora} para {comensales} personas ha devuelto las siguientes opciones: {lista_de_opciones_disponibles_formateada}. Por favor, presenta estas opciones al usuario de manera amigable y clara. Pregúntale cuál de estas opciones prefiere o si desea probar con otra fecha u hora.

Esta gestión del diálogo basada en estados y la orquestación cuidadosa de las llamadas a la API de ChatGPT y a las funciones externas son, en opinión de este estudio, la clave para construir un agente que no solo sea inteligente en sus respuestas, sino también efectivo en la consecución de la tarea de reserva.

4.5. Integración con Sistemas Externos (Function Calling)

Esta sección es, para el desarrollo de este TFM, una de las más críticas y emocionantes, ya que el function calling es el mecanismo que permitirá al agente de IA ir más allá de una simple conversación y realizar acciones reales o consultar información en tiempo real. Es lo que lo hará verdaderamente funcional.

4.5.1. La Necesidad de Conexión a la API del Restaurante (POS/RMS/DB)

ChatGPT, por sí mismo, no tiene conocimiento del mundo en tiempo real ni acceso a bases de datos privadas. (Zeng et al., 2023) No sabe si hay mesas disponibles en "El Buen Sabor" para el próximo sábado a las 9 PM, ni puede anotar una reserva en su sistema. Por lo tanto, es absolutamente indispensable conectar el agente de IA con el sistema de gestión que utilice el restaurante. Este sistema podría ser un software de Punto de Venta (POS) con módulo de reservas, un Sistema de Gestión de Restaurantes (RMS), o incluso una base de datos personalizada que el restaurante haya desarrollado.

La viabilidad y la complejidad de esta integración dependerán crucialmente de un factor externo: si el sistema del restaurante ofrece una API (Interfaz de Programación de Aplicaciones) accesible y, muy importante, de la calidad y claridad de su documentación. Sin una API a la que el backend pueda llamar, la automatización completa de la reserva sería extremadamente difícil, si no imposible.

4.5.2. Explicación Detallada del Mecanismo de Function Calling

El function calling (o tool_calls en las versiones más recientes de la API de OpenAI) permite que el modelo LLM indique a la aplicación cuándo considera que necesita ejecutar una pieza de código externa para poder continuar la conversación o responder adecuadamente al usuario. El proceso, tal como se ha entendido y se planea implementar, sigue estos pasos:

1. **Definición de Funciones (Herramientas):** En el código de backend, se definirán las funciones que se desea que el LLM pueda "solicitar". Para cada función, se especificará su nombre, una descripción clara de lo que hace (esta descripción es muy importante, ya que el LLM la usará para decidir cuándo llamarla), y los parámetros que espera recibir, utilizando un esquema JSON Schema. Estas definiciones se envían a la API de ChatGPT junto con la conversación (messages) en el parámetro tools.

2. **Detección por parte del LLM:** El modelo analiza la conversación y el system prompt. Si, basándose en la solicitud del usuario y las descripciones de las funciones proporcionadas, determina que necesita ejecutar una de esas funciones para obtener información o realizar una acción, en lugar de devolver un mensaje de texto, la API de ChatGPT devolverá un objeto tool_calls. Este objeto contendrá el nombre de la función que el modelo quiere que se llame y un string JSON con los argumentos que el modelo ha extraído de la conversación y considera necesarios para esa función.

3. **Ejecución en el Backend:** El código de backend recibirá esta respuesta. Verificará si contiene tool_calls. Si es así, parseará los argumentos JSON, identificará la función solicitada (p. ej., check_availability) y ejecutará la lógica correspondiente en el servidor. Esta lógica es la que *realmente* interactuará con la API del restaurante o la base de datos.

4. **Obtención del Resultado de la Función:** La función local (p. ej., mi_funcion_check_availability) se ejecutará y devolverá un resultado. Este resultado podría ser, por ejemplo, una lista de horarios disponibles, un ID de confirmación de reserva, un mensaje de error si algo falló, etc. Este resultado debe ser serializable (normalmente un string o JSON).

5. **Segunda Llamada a ChatGPT con el Resultado:** El backend tomará este resultado, lo formateará adecuadamente (como un mensaje con role: "tool", el tool_call_id correspondiente y el content con el resultado de la función) y lo añadirá al historial de messages. Luego, realizará una *segunda llamada* a la API de ChatGPT, enviando el historial actualizado (que ahora incluye la solicitud de función original y el resultado obtenido por el código).

6. **Respuesta Final del LLM al Usuario:** Con esta nueva información (el resultado de la función), el LLM ahora tendrá el contexto necesario para formular una respuesta final, coherente y en lenguaje natural para el usuario (pj. "Sí, tenemos mesas disponibles a las 20:00 y 20:30. ¿Cuál prefiere?").

Este ciclo de dos pasos (LLM solicita -> Código ejecuta -> Código devuelve resultado al LLM -> LLM responde al usuario) es la esencia del function calling.

4.5.3. Definición de Funciones (Herramientas) Relevantes para Reservas

Para el agente de reservas, se han identificado varias funciones clave que necesitarán ser definidas. A continuación, se presentan los esquemas JSON conceptuales que se usarán, basándose en la estructura que espera la API de OpenAI.

```
tools_para_reservas = [
    {
        "type": "function",
        "function": {
            "name": "check_availability",
            "description": "Verifica la disponibilidad de mesas en el
restaurante para una fecha, hora y número de comensales específicos. Devuelve
los horarios disponibles o informa si no hay disponibilidad.",
            "parameters": {
                "type": "object",
                "properties": {
                    "date": {
                        "type": "string",
                        "description": "La fecha deseada para la reserva, en
formato YYYY-MM-DD."
                    },
                    "time": {
                        "type": "string",
                        "description": "La hora deseada para la reserva, en
formato HH:MM (24 horas)."
                    },
                    "party_size": {
                        "type": "integer",
                        "description": "El número de comensales para la
reserva."
                    }
                },
                "required": ["date", "time", "party_size"]
            }
        }
    },
    {
        "type": "function",
        "function": {
            "name": "create_reservation",
            "description": "Crea una nueva reserva en el sistema del
restaurante con los detalles proporcionados. Devuelve un ID de confirmación
si la reserva es exitosa.",
            "parameters": {
                "type": "object",
```

```
                "properties": {
                    "date": {"type": "string", "description": "Fecha de la
reserva (YYYY-MM-DD)."},
                    "time": {"type": "string", "description": "Hora de la
reserva (HH:MM)."},
                    "party_size": {"type": "integer", "description": "Número
de comensales."},
                    "guest_name": {"type": "string", "description": "Nombre
completo del cliente principal para la reserva."},
                    "contact_phone": {"type": "string", "description":
"Número de teléfono de contacto del cliente."},
                    "notes": {
                        "type": "string",
                        "description": "Notas adicionales para la reserva,
como alergias (ej. 'alergia a los cacahuetes'), restricciones dietéticas (ej.
'vegetariano'), preferencias de asiento (ej. 'cerca de la ventana'), u
ocasión especial (ej. 'cumpleaños'). Opcional."
                    }
                },
                "required": ["date", "time", "party_size", "guest_name",
"contact_phone"]
            }
        }
    },
    {
        "type": "function",
        "function": {
            "name": "get_booking_details",
            "description": "Busca y recupera los detalles de una reserva
existente utilizando el nombre del cliente y su número de teléfono, o un ID
de reserva.",
            "parameters": {
                "type": "object",
                "properties": {
                    "guest_name": {"type": "string", "description": "Nombre
bajo el cual se hizo la reserva. Opcional si se proporciona booking_id."},
                    "contact_phone": {"type": "string", "description":
"Teléfono de contacto asociado a la reserva. Opcional si se proporciona
booking_id."},
                    "booking_id": {"type": "string", "description": "ID de
confirmación de la reserva. Opcional si se proporcionan nombre y teléfono."}
                },
                "required": [] // La lógica de validación para asegurar al
menos un identificador se hará en el backend
            }
        }
```

```
    }
    // Se podrían añadir funciones para modify_reservation y
cancel_reservation de manera similar.
]
```

Es importante que las description de las funciones y sus parámetros sean muy claras, ya que el LLM se basará en ellas para decidir cuándo y cómo usarlas.

4.5.4. Pasos de Implementación (Ejemplo Conceptual en Python)

Aquí se presenta un esqueleto de cómo se manejaría la respuesta de la API y la ejecución de funciones en Python. Este código se integraría en el bucle principal de la aplicación de chat.

```
import json
# Suponiendo:
# client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
# MODELO_CHATGPT = "gpt-4o"
# mensajes = [{"role": "system", "content": system_prompt_reservas}, {"role":
"user", "content": "Quiero reservar para 2 mañana a las 8pm"}]
# tools_para_reservas = [...] # Definidas como en el ejemplo anterior

# --- Primera llamada a la API ---
# print("Enviando a ChatGPT:", mensajes)
# respuesta_api = client.chat.completions.create(
#     model=MODELO_CHATGPT,
#     messages=mensajes,
#     tools=tools_para_reservas,
#     tool_choice="auto" # Permite al modelo decidir si llama a una función
# )

# mensaje_respuesta_llm = respuesta_api.choices[0].message
# mensajes.append(mensaje_respuesta_llm) # Añadir la respuesta del LLM (sea
texto o tool_call) al historial

# if mensaje_respuesta_llm.tool_calls:
#     print("ChatGPT solicitó llamar a las siguientes funciones:",
mensaje_respuesta_llm.tool_calls)
#     available_functions = {
#         "check_availability": mi_funcion_real_check_availability,
#         "create_reservation": mi_funcion_real_create_reservation,
#         "get_booking_details": mi_funcion_real_get_booking_details,
#     }
```

```
#     for tool_call in mensaje_respuesta_llm.tool_calls:
#         function_name = tool_call.function.name
#         function_to_call = available_functions.get(function_name)

#         if function_to_call:
#             function_args = json.loads(tool_call.function.arguments)
#             print(f"Ejecutando función '{function_name}' con argumentos:
{function_args}")

#             # --- Ejecutar la función real ---
#             try:
#                 function_response = function_to_call(**function_args)
#                 print(f"Resultado de '{function_name}':
{function_response}")
#                 # Añadir el resultado de la función al historial para la
segunda llamada
#                 mensajes.append(
#                     {
#                         "tool_call_id": tool_call.id,
#                         "role": "tool",
#                         "name": function_name,
#                         "content": json.dumps(function_response) # El
resultado debe ser un string JSON
#                     }
#                 )
#             except Exception as e:
#                 print(f"Error al ejecutar la función {function_name}: {e}")
#                 mensajes.append(
#                     {
#                         "tool_call_id": tool_call.id,
#                         "role": "tool",
#                         "name": function_name,
#                         "content": json.dumps({"error": str(e)})
#                     }
#                 )
#         else:
#             print(f"Error: Función '{function_name}' no reconocida.")
#             # Se podría añadir un mensaje de error genérico al historial
aquí

#     # --- Segunda llamada a la API con el resultado de la(s) función(es) --
-
#     print("Enviando a ChatGPT con resultados de funciones:", mensajes)
#     segunda_respuesta_api = client.chat.completions.create(
#         model=MODELO_CHATGPT,
#         messages=mensajes
```

```
#         # No se necesita pasar 'tools' aquí usualmente, a menos que se
espere otra llamada a función inmediatamente
#     )
#     mensaje_final_llm = segunda_respuesta_api.choices[0].message
#     print("Respuesta final del Asistente (post-función):",
mensaje_final_llm.content)
#     mensajes.append(mensaje_final_llm) # Añadir al historial

# else:
#     # No hubo llamada a función, la respuesta es texto directo del LLM
#     if mensaje_respuesta_llm.content:
#         print("Respuesta directa del Asistente:",
mensaje_respuesta_llm.content)
#     else:
#         print("Respuesta inesperada del LLM (sin contenido ni
tool_calls).")

# # --- Aquí se definirían las funciones reales (simuladas por ahora) ---
# def mi_funcion_real_check_availability(date: str, time: str, party_size:
int):
#     print(f"[SIMULACIÓN API RESTAURANTE] Verificando disponibilidad para
{date} a las {time} para {party_size} personas.")
#     if "2025-06-01" in date and party_size <= 4: # Simulación simple
#         return {"status": "disponible", "available_times": [time, "20:30",
"21:00"], "message": f"Hay disponibilidad para {party_size} el {date}."}
#     else:
#         return {"status": "no_disponible", "message": f"Lo siento, no hay
disponibilidad para {party_size} el {date} a las {time}."}

# def mi_funcion_real_create_reservation(date: str, time: str, party_size:
int, guest_name: str, contact_phone: str, notes: str = ""):
#     print(f"[SIMULACIÓN API RESTAURANTE] Creando reserva para {guest_name}
({contact_phone}) el {date} a las {time} para {party_size} personas. Notas:
{notes}")
#     booking_id = f"BKNG-{hash((date, time, guest_name)) % 10000}"
#     return {"status": "reserva_confirmada", "booking_id": booking_id,
"details": {"guest_name": guest_name, "date": date, "time": time,
"party_size": party_size, "notes": notes}}

# def mi_funcion_real_get_booking_details(guest_name: str = None,
contact_phone: str = None, booking_id: str = None):
#     print(f"[SIMULACIÓN API RESTAURANTE] Buscando reserva:
Nombre='{guest_name}', Tel='{contact_phone}', ID='{booking_id}'")
#     if booking_id == "BKNG-1234" or (guest_name == "Juan Perez" and
contact_phone == "555-1234"):
#         return {"status": "reserva_encontrada", "booking_id": "BKNG-1234",
```

```
"guest_name": "Juan Perez", "date": "2025-05-30", "time": "20:00",
"party_size": 2, "notes": "Aniversario"}
#     else:
#         return {"status": "reserva_no_encontrada", "message": "No se
encontró una reserva con los datos proporcionados."}
```

Este código es conceptual y necesitará ser integrado en una aplicación más grande que maneje la sesión del usuario, la interfaz, etc. Las funciones mi_funcion_real_... son donde residirá la lógica de negocio real y las llamadas a las APIs del restaurante.

4.5.5. Consideraciones sobre APIs Específicas de Restaurantes y Código Abierto

La implementación real de las funciones como mi_funcion_real_check_availability dependerá al 100% de la API específica que proporcione el sistema POS o RMS del restaurante. Esto implicará, probablemente, manejar autenticación adicional (tokens, claves API del restaurante), diferentes formatos de datos, y una gestión de errores específica para esa API.

Se ha explorado si existen soluciones de código abierto que pudieran facilitar esta tarea. En plataformas como GitHub, se pueden encontrar algunos envoltorios generales para la API de ChatGPT, o incluso proyectos específicos de bots de reserva o APIs de reserva genéricas. Sin embargo, es poco probable, en opinión de este estudio, encontrar una solución de código abierto que se integre "out-of-the-box" con cualquier sistema POS/RMS sin una adaptación y desarrollo significativos. La clave, se insiste, será obtener y entender la documentación de la API del sistema de destino del restaurante.

Por lo tanto, se ve la integración como un puente a construir: por un lado, la capacidad conversacional estandarizada de ChatGPT, y por el otro, la diversidad de sistemas específicos de cada restaurante. Mientras que la interacción con ChatGPT puede seguir las pautas desarrolladas en esta guía, la conexión final requerirá un desarrollo a medida, basado en la API particular del restaurante con el que se quiera integrar.

5. Desafíos y Consideraciones Éticas

La implementación de un agente de reservas basado en ChatGPT, aunque prometedora, conlleva una serie de desafíos técnicos, operativos y éticos que deben ser cuidadosamente considerados y abordados para garantizar un despliegue exitoso y responsable.

5.1. Desafíos Técnicos Inherentes a los Modelos de Lenguaje

La viabilidad y eficacia de un agente de reservas basado en LLM dependen de la superación de varias limitaciones técnicas intrínsecas a esta tecnología.

- **Precisión y Mitigación de Alucinaciones:** Los Modelos de Lenguaje de Gran Escala (LLM), como ChatGPT, son susceptibles de generar información imprecisa o incurrir en "alucinaciones", es decir, producir datos ficticios con apariencia de veracidad. En el contexto específico de las reservas, este fenómeno podría traducirse en la comunicación de horarios erróneos, la confirmación de disponibilidad inexistente (especialmente si la integración con sistemas subyacentes presenta fallos) o la incorrecta interpretación de detalles críticos proporcionados por el usuario, como pueden ser las alergias alimentarias.

    - **Estrategias de Mitigación:** Para contrarrestar estos riesgos, es crucial depender de mecanismos como el *function calling* para la obtención de datos factuales (ej. disponibilidad horaria, mesas libres) directamente desde el sistema de gestión del. Se requiere una ingeniería de *prompts* robusta y meticulosamente diseñada para la extracción precisa de información sensible (alergias, preferencias especiales), complementada con pasos de verificación explícita con el usuario. En lo referente a información estática, como respuestas a preguntas frecuentes (FAQs), la implementación de arquitecturas Retrieval-Augmented Generation o el uso de bases de conocimiento internas pueden asegurar una mayor fidelidad de la información.

- **Limitaciones de la Ventana de Contexto:** Los LLM operan con una "ventana de contexto" finita, limitando la cantidad de información conversacional previa que pueden procesar simultáneamente en una única interacción. En diálogos extensos o particularmente complejos, el modelo podría omitir u "olvidar" detalles relevantes mencionados con anterioridad por el usuario.

    - **Estrategias de Mitigación:** Resulta fundamental una gestión explícita del estado de la conversación en la lógica del *backend*. Para interacciones prolongadas, aunque menos comunes en un flujo de reserva estándar, se podrían aplicar técnicas de resumen progresivo del historial conversacional antes de su envío a la API.

- **Gestión de Errores y Fiabilidad del Sistema:** El sistema debe exhibir una alta robustez frente a posibles fallos en la API de ChatGPT, errores durante la ejecución de funciones o problemas de comunicación con la API del sistema de reservas del restaurante. Una respuesta anómala o un fallo no gestionado adecuadamente pueden interrumpir el proceso de reserva, generando frustración en el usuario.

    - **Estrategias de Mitigación:** Se debe implementar un manejo exhaustivo de excepciones (ej. bloques try-except) en el código del *backend* para todas las interacciones con APIs externas. Es necesario diseñar flujos de recuperación de errores que contemplen reintentos automáticos, la comunicación clara al usuario sobre la incidencia y la oferta de alternativas viables, como la derivación a un agente humano o la sugerencia de contacto telefónico directo con el establecimiento.

- **Consistencia y Previsibilidad del Comportamiento:** Asegurar que el agente de IA se comporte de manera consistente y predecible ante interacciones similares es un factor crítico para fomentar la confianza del usuario. La naturaleza inherentemente probabilística de los LLM puede introducir variabilidad en las respuestas.

    - **Estrategias de Mitigación:** Se deben conducir pruebas exhaustivas que abarquen una amplia diversidad de escenarios de interacción. El ajuste fino de la temperatura de la API (optando por valores inferiores para una mayor previsibilidad) y el refinamiento continuo de los *prompts* de sistema y las instrucciones son clave para minimizar la ambigüedad y la variabilidad no deseada.

5.2. Desafíos de Integración Operacional

La introducción de esta tecnología requiere una cuidadosa planificación para su correcta inserción en las operaciones diarias del restaurante.

- **Impacto en el Personal y Necesidades de Formación:** La incorporación de un agente de IA puede suscitar aprensión en el personal respecto a la estabilidad laboral. Adicionalmente, el equipo humano necesitará comprender el funcionamiento del sistema, cómo interactuar con él para tareas específicas (p.ej., gestión de anulaciones manuales o resolución de excepciones) y cómo gestionar eficientemente las derivaciones o escalamientos provenientes del bot.

    - **Estrategias de Mitigación:** Es fundamental comunicar de forma transparente que la IA se implementa como una herramienta para potenciar la eficiencia operativa y liberar al personal para que pueda dedicarse a tareas de mayor valor añadido, como la interacción personalizada con los clientes presentes en el local, y no como un sustituto de sus funciones. Se

debe proveer formación exhaustiva sobre el manejo del sistema y los nuevos protocolos de trabajo.

- **Adaptación de Flujos de Trabajo Existentes:** El agente de IA debe integrarse de manera fluida y coherente en los procesos de reserva preexistentes. Es imperativo definir con claridad quién supervisará las reservas gestionadas por el bot, cómo se abordarán las solicitudes complejas o inusuales que excedan la capacidad del agente, y cómo se garantizará la sincronización de la información con el personal de sala.

    - **Estrategias de Mitigación:** Se debe diseñar meticulosamente el flujo de trabajo, estableciendo puntos de escalada claros hacia el personal humano. Una integración técnica sólida con los sistemas existentes (POS/RMS) es crucial para centralizar la información y asegurar su consistencia.

- **Análisis de Costes de Implementación:** La adopción de esta tecnología conlleva costes asociados al desarrollo inicial; por la la contratación de servicios especializados, las tarifas por uso de la API de OpenAI (dependientes del volumen de interacciones y el modelo de LLM seleccionado), y los posibles costes derivados de la integración con APIs de terceros.

    - **Estrategias de Mitigación:** Se debe evaluar con rigor el retorno de la inversión (ROI) esperado, considerando ahorros en tiempo del personal, reducción de errores de gestión y el potencial incremento en el volumen de reservas (Bushra, 2024). Considerar el inicio con modelos de LLM más económicos (ej, GPT-3.5 Turbo) si la funcionalidad lo permite, y optimizar el uso de la API mediante *prompts* eficientes.

- **Mantenimiento Continuo y Evolutivo del Sistema:** La IA no constituye una solución estática ("configurar y olvidar"), sino que demanda un compromiso de mantenimiento proactivo y continuo. Esto incluye la monitorización constante del rendimiento, el ajuste iterativo de los *prompts* basado en el análisis de interacciones reales, la adaptación a cambios en las APIs (tanto de OpenAI como del propio restaurante) y, potencialmente, el reentrenamiento si se emplean modelos personalizados.

    - **Estrategias de Mitigación:** Es necesario asignar recursos específicos para el mantenimiento continuo. La recopilación y el análisis sistemático de los registros de interacción son fundamentales para identificar áreas de mejora y optimizar el comportamiento del agente.

5.3. Consideraciones Éticas y Legales Fundamentales

Las implicaciones éticas y el cumplimiento del marco legal vigente constituyen aspectos cruciales que deben ser intrínsecos al diseño y desarrollo del sistema desde sus fases iniciales.

- **Privacidad de Datos y Cumplimiento Normativo (GDPR):** El agente gestionará datos personales de carácter sensible, como nombre, número de teléfono, dirección de correo electrónico y, potencialmente, información sobre alergias y preferencias alimentarias. Es imperativo garantizar el cumplimiento estricto de las regulaciones de protección de datos aplicables, destacando el Reglamento General de Protección de Datos (RGPD) en el contexto europeo (Avneesh Sood, n.d.).

    - **Estrategias de Mitigación:** Se deben implementar medidas técnicas y organizativas robustas, incluyendo el almacenamiento seguro de los datos, el cifrado de la información tanto en tránsito como en reposo, la redacción de políticas de privacidad claras, concisas y accesibles para los usuarios, la obtención de consentimiento explícito cuando sea legalmente requerido, y la verificación de que OpenAI (así como cualquier otro proveedor involucrado) cumple con los requisitos establecidos para el procesamiento de datos. Se debe ejercer extrema cautela para prevenir la fuga accidental de datos confidenciales a través de los *prompts* enviados a la API.

- **Seguridad de la Información y Gestión de Credenciales:** La salvaguarda de credenciales de acceso (claves API) y la seguridad de las comunicaciones entre los distintos componentes del sistema son vitales para prevenir accesos no autorizados y brechas de seguridad.

    - **Estrategias de Mitigación:** Adherirse a las mejores prácticas para la gestión de claves API, como evitar su incrustación directa en el código fuente, utilizando en su lugar variables de entorno o sistemas de gestión de secretos. Emplear conexiones seguras para todas las comunicaciones que involucren APIs.

- **Transparencia en la Interacción Humano-IA:** Los usuarios deben ser informados de manera explícita y comprensible si están interactuando con un sistema de inteligencia artificial o con un ser humano. Si el usuario no tiene claro con quien interactúa puede perder confinza en el sistema y en el establecimiento.

    - **Estrategias de Mitigación:** Incluir una declaración clara e inequívoca al inicio de cada interacción, indicando que el usuario está comunicándose con un chatbot.

- **Mitigación de Sesgos Algorítmicos y Equidad:** Los LLM pueden perpetuar o incluso amplificar sesgos inherentes a sus datos de entrenamiento. Esto podría manifestarse en un trato desigual hacia los clientes, por ejemplo, mediante la mala interpretación de acentos o variantes dialectales no estándar, la oferta sistemática de peores opciones a determinados grupos demográficos, o la formulación de recomendaciones sesgadas.

    - **Estrategias de Mitigación:** Diseñar *prompts* con el objetivo de promover la neutralidad y la objetividad. Implementar auditorías periódicas de equidad para detectar y corregir posibles sesgos en el comportamiento del agente. Hay que asegurar que las reglas de negocio implementadas mediante *function calling* sean justas y no discriminatorias.

- **Responsabilidad (Accountability) y Rendición de Cuentas:** Resulta imperativo establecer un marco claro de responsabilidad ante errores críticos cometidos por el agente, como una reserva incorrectamente registrada o perdida, o una alergia no registrada adecuadamente que derive en un problema de salud para el cliente.

    - **Estrategias de Mitigación:** Establecer políticas de responsabilidad interna bien definidas. Implementar un sistema de registro detallado de todas las interacciones y decisiones tomadas por el agente para facilitar la trazabilidad e investigación de errores. Asegurar la existencia de mecanismos de supervisión humana y vías de recurso efectivas para los clientes afectados.

- **Equilibrio entre Automatización y Contacto Humano Personalizado:** El sector de la hostelería se caracteriza por el alto valor que los clientes otorgan al contacto humano, la empatía y la atención personalizada. Una automatización excesiva o percibida como impersonal puede deteriorar significativamente la experiencia del cliente.

    - **Estrategias de Mitigación:** Diseñar el agente de IA para gestionar eficientemente tareas rutinarias y repetitivas, pero incorporando siempre una opción clara, accesible y sencilla para que el usuario pueda solicitar la intervención de un agente humano si así lo desea, o si la situación excede las capacidades del bot. Utilizar la IA como un medio para optimizar el tiempo del personal, permitiéndoles así enfocarse en ofrecer interacciones humanas de mayor calidad y valor.

En conclusión, estos desafíos técnicos, operativos y ético-legales, que no representa únicamente una obligación de la normativa. Constituye, los cimientos para desarrollar un sistema de reservas automatizado que sea no solo tecnológicamente avanzado y eficiente, sino también fiable, equitativo y capaz de generar y mantener la confianza de los clientes y del personal del establecimiento. La negligencia en el mantenimiento de la privacidad, la transparencia o la equidad algorítmica puede derivar en problemas legales, así como en un deterioro de la reputación y la viabilidad del negocio.

6. Conclusiones y Trabajo Futuro

En este capítulo final, se presentan los hallazgos y resultados consolidados de este Trabajo de Fin de Grado. Expondré las conclusiones principales que he extraído del desarrollo de la guía práctica para implementar un agente de inteligencia artificial, basado en ChatGPT, para la gestión de reservas en restaurantes. Asimismo, identificaré las limitaciones inherentes a este estudio y propondré líneas de investigación y desarrollo futuras que considero podrían expandir y profundizar en la aplicación de esta tecnología en el sector de la hostelería.

6.1. Conclusiones Principales

A lo largo de la realización de este Trabajo de Fin de Grado, he alcanzado una serie de conclusiones significativas en relación con los objetivos que me propuse y los resultados obtenidos:

1. **Consecución del Objetivo Principal**: Considero que se ha desarrollado con éxito una guía práctica y exhaustiva, tal como se planteó en el objetivo principal. Esta guía está diseñada para la creación e implementación de un agente de IA basado en ChatGPT para la gestión de reservas en restaurantes y abarca desde la conceptualización y el diseño funcional (Capítulo 3) hasta una hoja de ruta detallada para la implementación técnica (Capítulo 4). En ella, he incluido aspectos clave como la interacción con la API de OpenAI, la ingeniería de *prompts*, la utilización del *function calling* y las estrategias para la gestión del diálogo.

2. **Viabilidad de ChatGPT para la Gestión de Reservas**: A través de la investigación realizada, concluyo que ChatGPT, mediante su API y, de forma crucial, la funcionalidad de *function calling*, representa una herramienta potente y viable para el desarrollo de agentes conversacionales sofisticados. Estos agentes están destinados a la gestión de reservas y su capacidad para la comprensión del lenguaje natural (NLU) permite manejar una amplia gama de interacciones con los usuarios de manera intuitiva.

3. **Criticidad de la Ingeniería de *Prompts* y el *Function Calling***: La investigación y el desarrollo práctico llevados a cabo en este TFG han subrayado que una ingeniería de *prompts* meticulosa y bien diseñada es crucial. Esta permite dirigir el comportamiento del LLM, asegurar la extracción precisa de información clave (fechas, horas, número de comensales, alergias, etc.) y mantener un flujo conversacional coherente y efectivo, como detallé en la sección 4.3. Igualmente, el mecanismo de *function calling* (sección 4.5) se ha revelado como indispensable para dotar al agente de la capacidad de interactuar con sistemas externos (como la API del restaurante), transformándolo de un mero conversador a un sistema funcional capaz de verificar disponibilidad y formalizar reservas.

4. **Importancia de la Orquestación del Backend**: Si bien ChatGPT proporciona la inteligencia conversacional, he constatado que una lógica de *backend* robusta es imprescindible. Esta se encarga de la gestión del estado de la conversación, la orquestación de las llamadas a la API de ChatGPT, la ejecución de las funciones externas y el manejo del flujo general de la interacción, como se discute en las secciones 3.2, 3.3 y 4.4. El LLM es, por tanto, un componente central, pero no constituye la solución completa por sí mismo.

5. **Relevancia de la Guía Práctica**: Estimo que este TFG aporta un recurso de valor para desarrolladores, restauradores y estudiantes interesados en la aplicación práctica de los LLMs en el sector hostelero. La guía que he elaborado busca acortar la brecha entre el potencial teórico de tecnologías como ChatGPT y su implementación efectiva en un caso de uso concreto y relevante, como es la gestión de reservas, cumpliendo así con el impacto esperado del proyecto (Capítulo 1).

6. **Conciencia de los Desafíos**: Durante el estudio, he identificado los desafíos técnicos (precisión, "alucinaciones", limitaciones de la ventana de contexto), operativos y ético-legales (Capítulo 5) inherentes a la implementación de estos sistemas, y la guía propuesta los aborda. Concluyo que una consideración proactiva y la implementación de estrategias de mitigación son fundamentales para un despliegue responsable y exitoso.

6.2. Limitaciones del Estudio

A pesar de mi esfuerzo por realizar un trabajo exhaustivo, este TFG presenta ciertas limitaciones que es importante reconocer:

1. **Alcance de la Implementación Práctica**: La guía que he desarrollado en el Capítulo 4 proporciona una hoja de ruta detallada y ejemplos de código conceptuales. Sin embargo, la implementación de un sistema completo, desplegado en un entorno de producción real e integrado con un sistema de gestión de restaurante (POS/RMS) específico, excede el alcance temporal y los recursos de los que disponía para este TFG. Por lo que el código de phyton que se muestra mi tfg es un modelo que necesitaría de ajustes para hacerlo funcionar.

2. **Evolución Tecnológica Acelerada**: Los modelos de lenguaje grande y las APIs asociadas, como las de OpenAI, están en constante y rápida evolución. Aunque he procurado utilizar información y modelos recientes algunas de las herramientas, técnicas específicas o incluso la estructura de la API podrían cambiar en el futuro, lo que requeriría adaptaciones a la guía.

3. **Generalización vs. Especificidad de Restaurantes**: Si bien la guía busca ofrecer principios y prácticas aplicables de forma general, soy consciente de que cada restaurante posee necesidades operativas, una infraestructura tecnológica y una clientela particular. Por tanto, cualquier implementación real requerirá un grado significativo de personalización y adaptación de la solución propuesta.

4. **Evaluación Empírica Reducida**: El enfoque principal de este TFG ha sido la creación de una guía práctica. Una evaluación empírica exhaustiva del rendimiento del agente propuesto (tiempos de respuesta, tasa de éxito en reservas, satisfacción del usuario) en un entorno real con un volumen significativo de interacciones no se ha llevado a cabo, ya que requeriría un despliegue completo que escapaba a las posibilidades de este trabajo.

5. **Profundidad en Aspectos Operativos y Legales Específicos**: Aunque en el Capítulo 5 abordo los desafíos operativos y ético-legales, un análisis exhaustivo del impacto socio-técnico en una organización específica, o una auditoría legal completa para el cumplimiento normativo (ej. RGPD) en un contexto de despliegue particular, requeriría investigaciones adicionales y especializadas que no formaban parte del alcance de este TFG.

6.3. Líneas de Trabajo Futuro

Los resultados obtenidos y las limitaciones identificadas en este TFG abren, desde mi punto de vista, diversas vías para futuras investigaciones y desarrollos que podrían expandir y mejorar la solución propuesta:

1. **Implementación y Validación en Entorno Real**:
    - Considero fundamental desarrollar e integrar plenamente el agente de reservas en uno o varios restaurantes reales, conectándolo con sus sistemas POS/RMS existentes.

    - Sería muy valioso realizar pruebas de usuario exhaustivas (UAT) tanto con clientes finales como con el personal del restaurante para recoger *feedback* y evaluar la usabilidad y eficacia del sistema.

    - Se deberían medir métricas de rendimiento clave: tasa de finalización de reservas, reducción de errores, tiempo ahorrado al personal, satisfacción del cliente.

2. **Personalización Avanzada y Aprendizaje Continuo**:
    - Sugiero explorar la capacidad del agente para aprender de interacciones pasadas y personalizar la experiencia del usuario (ej., recordar preferencias de mesa, alergias frecuentes, o sugerir horarios basados en reservas anteriores).

- Sería interesante investigar la integración con sistemas de CRM para una visión 360º del cliente.

3. **Interacción Multimodal**:

   - Propongo ampliar las capacidades del agente para incluir interacciones por voz, mediante la integración de tecnologías de reconocimiento de voz (Speech-to-Text) y síntesis de voz (Text-to-Speech), ofreciendo una experiencia más natural y accesible.

   - También se podría considerar la integración con elementos visuales, como la muestra de planos del restaurante o fotografías de platos si el sistema lo permite.

4. **Capacidades Proactivas del Agente**:
   - Una línea de desarrollo interesante sería desarrollar funcionalidades que permitan al agente tomar la iniciativa, como el envío de recordatorios de reserva, la confirmación proactiva de asistencia, o la notificación a usuarios en lista de espera sobre cancelaciones.

5. **Ampliación de Funcionalidades del Agente**:
   - Se podría extender el alcance del agente más allá de las reservas para incluir otras interacciones como la toma de pedidos para llevar (*take-away*), la respuesta a preguntas detalladas sobre el menú (posiblemente mediante técnicas de *Retrieval-Augmented Generation* - RAG con la carta del restaurante), o la gestión de programas de fidelización.

6. **Análisis Comparativo de Modelos y Plataformas**:
   - Recomiendo realizar estudios comparativos entre diferentes LLMs (incluyendo alternativas a OpenAI) y versiones más recientes, evaluando su rendimiento, coste y facilidad de integración para la tarea específica de gestión de reservas.

7. **Optimización de la Arquitectura RAG para FAQs**:
   - Sería beneficioso profundizar en la implementación y optimización de arquitecturas RAG para la gestión de preguntas frecuentes, buscando mejorar la precisión y la capacidad de respuesta ante consultas complejas o específicas del restaurante.

8. **Exploración del *Fine-tuning***:
   - Para restaurantes con un volumen suficiente de datos de interacción, valdría la pena investigar la viabilidad y los beneficios de realizar un *fine-tuning* de un modelo de lenguaje base para adaptarlo mejor al argot específico del restaurante, sus promociones habituales o los patrones de solicitud de sus clientes.

9. **Auditoría Ética Continua y Mitigación de Sesgos**:
    - Considero crucial desarrollar e implementar un marco para la auditoría ética periódica del agente una vez desplegado, con el fin de detectar, analizar y mitigar posibles sesgos algorítmicos, asegurando un trato justo y equitativo a todos los usuarios.

10. **Integración con Plataformas *Low-Code/No-Code***:
    - Finalmente, sugiero investigar cómo los principios y componentes de la guía desarrollada podrían adaptarse o servir de base para la creación de soluciones de reserva mediante IA en plataformas de desarrollo *low-code* o *no-code*, con el fin de democratizar el acceso a estas tecnologías para restaurantes con menores recursos técnicos.

La culminación de este Trabajo de Fin de Grado no solo evidencia, desde mi perspectiva, el potencial transformador de la inteligencia artificial en el sector de la restauración, sino que también sienta las bases para futuras innovaciones que, espero, continúen mejorando la eficiencia operativa y la experiencia del cliente.

7. Bibliografía / Referencias

Ackerson, T. F., Dan. (2023, August 3). Function Calling: Integrate Your GPT Chatbot With Anything. *Semaphore*. https://semaphore.io/blog/function-calling

Alawami, A., Alawami, M., Obaid, A., Alrushaydan, M., Boudjedra, M. L., Boudjedra, B. E., Alharbi, M., Alzoori, H., Abosaif, S., Bshir, G., Alawami, H., & Abufawr, M. (2025). A Systematic Review of AI-Driven Innovations in the Hospitality Sector: Implications on Restaurant Management. *American Journal of Industrial and Business Management*, *15*(01), 30–40. https://doi.org/10.4236/ajibm.2025.151003

Alipour, H., Pendar, N., & Roy, K. (2024). *ChatGPT Alternative Solutions: Large Language Models Survey*. https://doi.org/10.5121/csit.2024.140514

Aslam, W., Ham, M., Mirza, F., Ting, D. H., & Hussain, A. (2025). Revolutionizing food ordering: Predicting the dynamics of chatbot adoption in a tech-driven era. *Journal of Foodservice Business Research*, 1–25. https://doi.org/10.1080/15378020.2025.2468035

Avneesh Sood. (n.d.). *How AI is redefining hospitality – and why we need policies to guide it—ET HospitalityWorld*. ETHospitalityWorld.Com. https://hospitality.economictimes.indiatimes.com/news/speaking-heads/how-ai-is-redefining-hospitality-and-why-we-need-policies-to-guide-it/118925567

Bushra, S. (2024, December 17). *How Conversational AI in Restaurants Transforms Customer Engagement*. https://convin.ai/blog/conversational-ai-restaurants

Celi-Parraga, R. J., Varela-Tapia, E. A., Acosta-Guzmán, I. L., & Montaño-Pulzara, N. R. (2021). Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional textual. *AlfaPublicaciones*, *3*(4.1), 40–52. https://doi.org/10.33262/ap.v3i4.1.123

Choi, Y., & Kim, D. (2024). Artificial Intelligence in The Tourism Industry: Current Trends and Future Outlook. *International Journal on Advanced Science, Engineering and Information Technology*, *14*(6), 1889–1895. https://doi.org/10.18517/ijaseit.14.6.20452

Davis, G. (2024, May 7). *How AI is Simplifying the Booking Process for Restaurants*. https://www.intelligentliving.co/ai-booking-process-for-restaurants/

Joshi, H., Liu, S., Chen, J., Weigle, R., & Lam, M. S. (2024). *Coding Reliable LLM-based Integrated Task and Knowledge Agents with GenieWorksheets* (No. arXiv:2407.05674). arXiv. https://doi.org/10.48550/arXiv.2407.05674

Lacalle, E. (2023, April 4). *Impact of ChatGPT in the hospitality industry | Mews Systems*. https://www.mews.com/en/blog/chat-gpt-in-hospitality

Pavlo. (2024, July 2). *ChatGPT API: Tutorial Guide*. https://www.codica.com/blog/chat-gpt-guide/

To, W. M., & Yu, B. T. W. (2025). Artificial Intelligence Research in Tourism and Hospitality Journals: Trends, Emerging Themes, and the Rise of Generative AI. *Tourism and Hospitality*, *6*(2), 63. https://doi.org/10.3390/tourhosp6020063

Wüst, K., & Bremser, K. (2025). Artificial Intelligence in Tourism Through Chatbot Support in the Booking Process—An Experimental Investigation. *Tourism and Hospitality*, *6*(1), 36. https://doi.org/10.3390/tourhosp6010036

Zeng, Y., Rajasekharan, A., Padalkar, P., Basu, K., Arias, J., & Gupta, G. (2023). *Automated Interactive Domain-Specific Conversational Agents that Understand Human Dialogs* (No. arXiv:2303.08941). arXiv. https://doi.org/10.48550/arXiv.2303.08941