



Escuela Técnica Superior de  
**Ingeniería Informática**

TRABAJO FIN DE GRADO

# **Asistente con reconocimiento de voz o escritura para hostelería**

Realizado por  
**Mohamed Amine Bennani**

Para la obtención del título de  
Grado en Ingeniería Informática - Tecnologías Informáticas

Dirigido por  
Francisco Javier Ortega Rodríguez

En el departamento de  
Lenguajes y Sistemas Informáticos

**Convocatoria de junio/julio/diciembre, curso 2023/24**

# Agradecimientos

---

Me gustaría agradecer en primer lugar a mi familia, por todos los consejos que me han dado y por su apoyo incondicional durante toda mi vida, y especialmente durante estos años de estudio, a pesar de la distancia que nos separa.

También quiero agradecer a todos mis amigos y compañeros de estudios que me han ayudado, guiándome siempre hacia lo positivo y motivándome a seguir adelante con mis estudios y con el proyecto.

Y por supuesto, agradecer a D. Francisco Javier Ortega por todas sus tutorías y consejos a lo largo del desarrollo del proyecto .

# Resumen

---

Este proyecto tiene como propósito la creación de una aplicación móvil que permita integrar un sistema de reconocimiento de voz, permitiendo a los usuarios interactuar con un asistente virtual personalizado para el ámbito de la hostelería. Los usuarios podrán solicitar información sobre los platos disponibles y realizar pedidos directamente a través de la aplicación comunicándose vocalmente con el asistente.

El asistente virtual opera de manera inteligente y optimizada, registrando los pedidos realizados por los usuarios y generando una lista completa de los platos seleccionados junto con el precio total al finalizar la interacción. El asistente nunca sale del contexto del ámbito del restaurante. Si un plato solicitado no está presente en la base de datos del restaurante, el asistente responde negativamente informando al usuario que el plato no está disponible.

Para el correcto funcionamiento del chatbot, este se conecta a través de Langchain a la API de OpenAI, específicamente utilizando el modelo GPT-4-1106-preview para proporcionar respuestas inteligentes y contextualmente adecuadas durante las interacciones.

Además, el sistema contendrá una aplicación web que la usarán únicamente los empleados o administradores del restaurante para gestionar los pedidos y aspectos relacionados con la configuración de la aplicación móvil y la personalización del menú del restaurante.

En este documento se detallarán todos los procesos de forma detallada que se han estado haciendo para poder realizar el proyecto de forma exitosa, comenzando con la introducción dónde se introducirá el tema. A continuación, se aborda el estudio previo donde se analizan los objetivos específicos del proyecto, los requisitos planteados para el correcto desarrollo del proyecto y se presenta una revisión del estado del arte.

Tras ello, habrá una sección de planificación donde se detalla como sería la planificación temporal y de costes inicial para la correcta realización del proyecto. Además, habrá una sección de ejecución donde se desarrolla el diseño del sistema y se explica paso por paso cómo ha sido su implementación. Finalmente, habrá un apartado de pruebas y otro exponiendo las conclusiones a las que se ha llegado.

# Abstract

---

The purpose of this project is to create a mobile application that integrates a voice recognition system, allowing users to interact with a personalised virtual assistant for the hospitality industry. Users will be able to request information on available dishes and place orders directly through the application by communicating vocally with the assistant.

The virtual assistant operates in an intelligent and optimised way, recording the orders placed by users and generating a complete list of the selected dishes along with the total price at the end of the interaction. The assistant never leaves the context of the restaurant environment. If a requested dish is not present in the restaurant's database, the assistant responds negatively by informing the user that the dish is not available.

For the chatbot to function properly, it connects via Langchain to the OpenAI API, specifically using the GPT-4-1106-preview model to provide intelligent and contextually appropriate responses during interactions.

In addition, the system will contain a web application that will be used only by restaurant employees or administrators to manage orders and aspects related to the configuration of the mobile application and the customisation of the restaurant menu.

This document will detail all the processes in detail that have been done in order to be able to carry out the project successfully, starting with the introduction where the topic will be introduced. Afterwards, the previous study is dealt with, where the specific objectives of the project are analysed, the requirements for the correct development of the project and a review of the state of the art is presented.

After that, there will be a planning section where the initial time and cost planning for the correct execution of the project will be detailed. In addition, there will be a section on execution where the design of the system is developed and how it has been implemented is explained step by step. Finally, there will be a section on testing and another on the conclusions reached.

# Índice general

---

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1.	Introducción y motivación	1
<b>2</b>	<b>Estudio previo</b>	<b>2</b>
2.1.	Introducción	2
2.2.	Antecedentes	2
2.3.	Objetivos del proyecto	3
2.4.	Estado del arte	4
2.4.1.	Definición de chatbot	4
2.4.2.	Funcionamiento de los chatbots	4
2.4.3.	Primeros chatbots	7
2.4.4.	Chatbots modernos	9
2.4.5.	Fine-Tuning y Prompt Engineering	10
2.4.6.	Herramientas para la interacción por voz	14
2.5.	Tabla de requisitos	15
2.5.1.	Requisitos de Información	16
2.5.2.	Requisitos Funcionales	17
2.5.3.	Requisitos No Funcionales	20
2.5.4.	Reglas de Negocio	21
2.6.	Alcance del proyecto	22
2.6.1.	Inclusiones	22
2.6.2.	Exclusiones	23
2.6.3.	Estructura de Descomposición del Trabajo (EDT)	24
2.6.4.	Diccionario de la EDT	25
2.7.	Metodología usada	32
2.8.	Acta de constitución	34
<b>3</b>	<b>Planificación del proyecto</b>	<b>35</b>
3.1.	Introducción	35
3.2.	Planificación material	35
3.3.	Planificación temporal	36
3.4.	Planificación de costes	37
<b>4</b>	<b>Ejecución del proyecto</b>	<b>38</b>
4.1.	Introducción	38
4.2.	Diseño del sistema	38
4.2.1.	Diseño del modelo de Inteligencia Artificial (Asistente virtual)	40
4.2.2.	Diseño de la aplicación móvil	42
4.2.3.	Diseño de la aplicación web	46
4.2.4.	Diseño de la base de datos	49
4.3.	Implementación del sistema	50
4.3.1.	Tecnologías Aplicadas	50

4.3.2. Desarrollo del proyecto . . . . .	54
<b>5 Pruebas . . . . .</b>	<b>74</b>
5.1. Introducción . . . . .	74
5.2. Pruebas del sistema . . . . .	74
5.3. Conclusión . . . . .	76
<b>6 Manual de instalación y de uso . . . . .</b>	<b>77</b>
<b>7 Conclusiones y planes futuros . . . . .</b>	<b>79</b>
7.1. Conclusiones . . . . .	79
7.2. Planes futuros . . . . .	80
<b>Bibliografía . . . . .</b>	<b>81</b>

# Índice de figuras

---

2.1. Ejemplo de pattern matching (AIML) . . . . .	5
2.2. Ejemplo de conversación . . . . .	6
2.3. Ejemplo de preprompt utilizado en Prompt Engineering . . . . .	11
2.4. Ejemplo de funcionamiento de RAG . . . . .	12
2.5. Representación de embeddings para la recuperación de documentos. . . . .	12
2.6. Representación general de Langchain. . . . .	13
2.7. Implementación de una solución RAG usando bases de datos vectoriales. . . . .	14
2.8. Estructura de Descomposición del Trabajo (EDT). . . . .	24
2.9. Representación del modelo en cascada. . . . .	33
2.10. Representación de la acta de constitución. . . . .	34
3.1. Diagrama de gantt. . . . .	36
4.1. Diagrama del diseño del sistema. . . . .	40
4.2. Diagrama de flujo de la lógica del chatbot. . . . .	41
4.3. Diagrama de navegación de la aplicación móvil. . . . .	43
4.4. Vista 1 . . . . .	44
4.5. Vista 2 . . . . .	44
4.6. Vista 3 . . . . .	45
4.7. Diagrama de navegación de la aplicación móvil. . . . .	48
4.8. Diagrama UML (modelo relacional) de las tablas en la base de datos. . . . .	49
4.9. Logotipo de Visual Studio Code. . . . .	50
4.10. Logotipo de Python . . . . .	51
4.11. Logotipo de DART . . . . .	51
4.12. Logotipo de JSX . . . . .	52
4.13. Logotipo de MongoDB . . . . .	52
4.14. Logotipo de Langchain . . . . .	53
4.15. Logotipo de Flutter . . . . .	53
4.16. Logotipo de Flask . . . . .	53
4.17. Logotipo de React . . . . .	54
4.18. Extracto del CSV de un menú de restaurante . . . . .	54
4.19. Extracto código: División del documento en chunks y almacenamiento en base de datos vectorial . . . . .	55
4.20. Extracto código: Preprompt de información del menú (Preprompt por defecto) . . . . .	56
4.21. Extracto código: Preprompt de toma de pedidos . . . . .	57
4.22. Extracto código: Inicialización de la memoria del chatbot . . . . .	57
4.23. Ejemplo del contenido de una memoria . . . . .	58
4.24. Extracto código: Función con la cadena de conversación. . . . .	58
4.25. Ejemplo de conversación con el chatbot. . . . .	59
4.26. Extracto código: Implementación de la conversión de texto a voz . . . . .	60
4.27. Extracto código: Implementación del fichero main . . . . .	61

4.28. Extracto código: Solicitud POST comunicándose con el chatbot . . . .	63
4.29. Distribución de las carpetas del proyecto Flutter . . . . .	64
4.30. Vista de introducción . . . . .	65
4.31. Vista de instrucciones . . . . .	65
4.32. Petición vocal al chatbot . . . . .	66
4.33. Respuesta vocal del chatbot . . . . .	66
4.34. Distribución de las carpetas del proyecto ReactJS . . . . .	68
4.35. Panel de control de la aplicación web . . . . .	69
4.36. Notificación de un pedido . . . . .	70
4.37. Lista de platos disponibles en el menú . . . . .	70
4.38. Modal para editar un plato . . . . .	71
4.39. Personalización de la aplicación móvil . . . . .	72
4.40. Modal para personalizar . . . . .	72
4.41. Lista de pedidos en curso . . . . .	73
5.1. Prueba 1 con GPT-3.5 Turbo . . . . .	74
5.2. Prueba 2 con GPT-3.5 Turbo . . . . .	74
5.3. Prueba con GPT-4-1106-Preview . . . . .	75
6.1. Introducción de la clave secreta OpenAI . . . . .	77



# Índice de tablas

---

2.1. Platos del menú: Requisito de Información . . . . .	16
2.2. Variación de cada plato: Requisito de Información . . . . .	16
2.3. Detalles de cada pedido: Requisito de Información . . . . .	16
2.4. Detalles de cada plato del pedido: Requisito de Información . . . . .	16
2.5. Información de los empleados: Requisito de Información . . . . .	16
2.6. Información de personalización: Requisito de Información . . . . .	17
2.7. Registro de pedidos por voz: Requisito Funcional . . . . .	17
2.8. Respuesta del asistente de voz: Requisito Funcional . . . . .	17
2.9. Interrupción del asistente de voz: Requisito Funcional . . . . .	17
2.10. Reinicio de la conversación con el asistente de voz: Requisito Funcional	18
2.11. Generación del recibo del pedido: Requisito Funcional . . . . .	18
2.12. Notificación a Cocina de Nuevos Pedidos: Requisito Funcional . . . . .	18
2.13. Eliminación de pedidos de la base de datos: Requisito Funcional . . . . .	18
2.14. Gestión de platos del menú: Requisito Funcional . . . . .	18
2.15. Configuración del restaurante: Requisito Funcional . . . . .	19
2.16. Conexión a la aplicación web: Requisito Funcional . . . . .	19
2.17. Creación de nuevos empleados en la aplicación web: Requisito Funcional . . . . .	19
2.18. Borrar empleados en la aplicación web: Requisito Funcional . . . . .	19
2.19. Modificación de la contraseña por el empleado: Requisito Funcional	19
2.20. Funcionamiento completo en Android: Requisito No Funcional . . . . .	20
2.21. Interfaz móvil sencilla e intuitiva: Requisito No Funcional . . . . .	20
2.22. Interfaz web sencilla e intuitiva: Requisito No Funcional . . . . .	20
2.23. Rendimiento del chatbot: Requisito No Funcional . . . . .	20
2.24. Unicidad de nombres de platos: Regla de Negocio . . . . .	21
2.25. Restricción de acceso a la aplicación web: Regla de Negocio . . . . .	21
2.26. Validación de las peticiones del cliente: Regla de Negocio . . . . .	21
2.27. Contexto de respuesta del sistema: Regla de Negocio . . . . .	21
2.28. Unicidad de correo electrónico y nombre de usuario: Regla de Negocio	21
2.29. Unicidad del número de pedido: Regla de Negocio . . . . .	21
2.30. Restricción de autoeliminación de empleados: Regla de Negocio . . . . .	22
2.31. Restricción al añadir un nuevo plato: Regla de Negocio . . . . .	22
2.32. Tabla 1 del diccionario de la EDT. . . . .	25
2.33. Tabla 2 del diccionario de la EDT. . . . .	26
2.34. Tabla 3 del diccionario de la EDT. . . . .	27
2.35. Tabla 4 del diccionario de la EDT. . . . .	28
2.36. Tabla 5 del diccionario de la EDT. . . . .	29
2.37. Tabla 6 del diccionario de la EDT. . . . .	30
2.38. Tabla 7 del diccionario de la EDT. . . . .	31
2.39. Tabla 8 del diccionario de la EDT. . . . .	32
3.1. Detalle de Costos del empleado . . . . .	37

# 1. Introducción

---

## 1.1. Introducción y motivación

Hoy vivimos en una sociedad donde cada vez se busca más optimizar y automatizar cualquier tipo de tareas para ganar tiempo y hacer que el ser humano se centre en tareas más interesantes y complejas. Este enfoque se extiende a diversos ámbitos de la vida cotidiana y profesional, desde la industria hasta la tecnología, donde la eficiencia y la productividad son imperativos. En este contexto, el uso de herramientas y sistemas automatizados se ha vuelto cada vez más relevante y necesario, permitiendo agilizar procesos, minimizar errores y liberar recursos para dedicarlos a actividades de mayor valor añadido.

De este modo, surge una idea de proyecto propuesta por D. Francisco Javier Ortega Rodríguez, que además de abordar los desafíos tecnológicos actuales, también redefine la interacción en el sector de la hostelería.

Dicho proyecto se centra en la implementación de un asistente virtual basado en el reconocimiento de voz.

Este asistente virtual mejorará la experiencia de los clientes y del personal del ámbito hostelero.

La motivación obtenida para realizar este proyecto es debida al reto de poder recrear una interacción tan natural que simule una conversación entre dos personas, en este caso, entre un cliente y un miembro del personal de hostelería. El propósito es permitir que los clientes puedan realizar todo tipo de consultas sobre los platos disponibles, obteniendo información detallada y realizando pedidos de forma intuitiva y eficiente mediante una conversación vocal con el asistente virtual.

Para comprender el alcance del problema, en el sector de la hostelería, la velocidad y la personalización son elementos clave. El personal y los clientes enfrentan desafíos que van desde la toma de pedidos eficiente hasta la obtención de información sobre el menú. Este proyecto busca abordar estas problemáticas, proporcionando soluciones intuitivas y eficaces.

Desde una perspectiva social y económica, la implementación exitosa de este asistente virtual podría redefinir la experiencia del cliente en la hostelería, generando no solo satisfacción, sino también eficiencia operativa y competitividad económica. La modernización de este sector a través de la integración de tecnologías modernas se convierte en un paso crucial para mantenerse a la par con las demandas de la era digital.

Mirando hacia el futuro, este proyecto no se limita únicamente a la hostelería, sino que podría servir como modelo para aplicaciones similares en diversos ámbitos.

## 2. Estudio previo

---

### 2.1. Introducción

En esta sección, se identificarán en primer lugar los antecedentes del proyecto, incluyendo las necesidades de los usuarios y patrocinadores clave. En segundo lugar, se detallarán los objetivos del proyecto y se desarrollará el estado del arte en relación con las tecnologías relevantes. Además, se detallarán también los requisitos y alcance de dicho proyecto junto con su estructura de descomposición del trabajo y el diccionario asociado. Por último, se explicará la metodología utilizada y se firmará el acta de constitución que guiará el desarrollo del proyecto.

### 2.2. Antecedentes

Este proyecto está diseñado para satisfacer las necesidades de usuarios interesados en mejorar la interacción en el sector de la hostelería mediante la implementación de un avanzado sistema de asistente virtual basado en reconocimiento de voz. Los **usuarios** potenciales de este sistema abarcan desde clientes individuales que buscan una experiencia más eficiente y personalizada al realizar pedidos o consultar información sobre el menú, hasta el personal de la hostelería que puede beneficiarse de la agilización de procesos mediante el reconocimiento de voz y la automatización de tareas.

Los **clientes** potenciales incluyen los establecimientos hosteleros que desean mejorar la experiencia de sus clientes implementando soluciones tecnológicas avanzadas para sus negocios. Además, esta tecnología puede atraer a empresas especializadas en el desarrollo de software para hostelería que buscan integrar soluciones de reconocimiento de voz en sus productos.

Los **patrocinadores** clave de este proyecto son la Universidad de Sevilla, la Escuela Técnica Superior de Ingeniería Informática y el Departamento de Lenguajes y Sistemas Informáticos. Esta colaboración garantizará el acceso a recursos esenciales para la realización del proyecto.

La dirección y ejecución de este proyecto estarán a cargo del propio estudiante, *Mohamed Amine Bennani*, quien asumirá el rol de **jefe de proyecto**. Su responsabilidad abarcará la organización y desarrollo completo de esta iniciativa, cuyo objetivo principal es diseñar un sistema de asistente virtual que mejore la experiencia en el sector hostelero mediante el reconocimiento de voz. Además, el profesor *D. Francisco Javier Ortega Rodríguez* tendrá su papel de supervisar y gestionar el progreso del proyecto.

La ejecución de este proyecto se llevará a cabo en un periodo de tiempo estimado de 300 horas como mínimo, conforme a los estándares de la Universidad

de Sevilla, lo que equivale a 12 créditos ECTS. Esta planificación temporal se ha establecido para asegurar un desarrollo integral y el cumplimiento exitoso de los objetivos propuestos.

## 2.3. Objetivos del proyecto

El proyecto se estructura en dos bloques claramente definidos con el propósito de transformar la experiencia en el sector de la hostelería.

En primer lugar, se aspira a implementar un sistema de asistente virtual basado en el reconocimiento de voz. Este sistema tiene como meta primordial mejorar la eficiencia de la toma de pedidos, además de mejorar la experiencia para clientes y personal. El asistente virtual, capaz de comprender comandos de voz, busca optimizar la interacción en el entorno hostelero y responder de manera efectiva mediante mensajes de voz a las peticiones de los usuarios. El objetivo central en esta fase consiste en lograr que la interacción entre el asistente virtual y el usuario se asemeje lo más posible a una conversación entre dos personas humanas.

En segundo lugar, se pretende desarrollar una interfaz de usuario amigable mediante aplicaciones móviles y web. Estas aplicaciones permitirán a los usuarios interactuar cómodamente con el sistema de reconocimiento de voz, realizar pedidos, recibir recomendaciones personalizadas y ofrecer retroalimentación. A continuación, se explica con más detalle el uso de la aplicación web y de la aplicación móvil:

- **La aplicación móvil** facilitará a los usuarios la comunicación con el asistente virtual a través de comandos vocales para solicitar información sobre el menú y realizar pedidos. El asistente virtual, respondiendo por voz, se encargará de responder a las preguntas del usuario y de registrar sus pedidos de manera precisa.
- **La aplicación web** estará destinada a los administradores y empleados, permitiéndoles recibir notificaciones de los pedidos realizados por los clientes para su preparación. Además, dichos administradores podrán gestionar y modificar el menú a través de esta plataforma.

Uno de los desafíos fundamentales de este proyecto reside en garantizar que el asistente virtual responda exclusivamente a las solicitudes relacionadas con el menú del restaurante, sin desviarse del contexto en ningún momento.

Finalmente, se listarán los objetivos docentes que se pretenden alcanzar mediante la realización de este proyecto:

- Investigación de modelos de Inteligencia Artificial para procesamiento de voz.
- Análisis de los fundamentos de los modelos de Inteligencia Artificial para comprender el funcionamiento de los chatbots, con especial énfasis en su capacidad para procesar las solicitudes de los usuarios sin perder el contexto.

- Ampliar los conocimientos sobre lenguajes de programación pertinentes para el desarrollo de aplicaciones web y móviles.
- Investigar y evaluar nuevas tecnologías emergentes, como frameworks o bases de datos.

## 2.4. Estado del arte

Esta sección comenzará con un estudio sobre la parte más fundamental del proyecto, el cual consiste en ver las tecnologías y algoritmos actuales que se usan para comprender el funcionamiento de la lógica conversacional de los chatbots. Además, se buscarán soluciones que permitan interactuar con el chatbot mediante voz, facilitando respuestas también en formato vocal.

### 2.4.1. Definición de chatbot

Un asistente virtual, también conocido como chatbot es un sistema de Inteligencia Artificial diseñado para simular interacciones humanas. Estos sistemas son ejemplos de interacción humano-computadora y juegan un papel destacado en la integración de la Inteligencia Artificial a nuestras vidas cotidianas.

Dichos chatbots usan técnicas avanzadas de Procesamiento del Lenguaje Natural (NLP) para comprender y generar respuestas coherentes en uno o más idiomas humanos. Su capacidad para procesar datos y adaptarse a diferentes contextos los convierte en herramientas valiosas para una variedad de aplicaciones.

Además de simular conversaciones humanas, los chatbots ofrecen numerosas ventajas en sectores como la educación, el comercio electrónico, la recuperación de información y otros campos. Por ejemplo, son utilizados para proporcionar tutoría personalizada, responder preguntas frecuentes, facilitar transacciones comerciales y mejorar la experiencia del usuario en diversos entornos.

### 2.4.2. Funcionamiento de los chatbots

Antes de hablar sobre la evolución histórica de los chatbots, es fundamental antes comprender algunos conceptos tecnológicos clave que han permitido su desarrollo y funcionamiento. En general, han habido dos enfoques para poder desarrollar un chatbot dependiendo de los algoritmos y técnicas adoptadas, enfoques de **búsqueda de patrones** y enfoques de **aprendizaje automático**:

#### **Búsqueda de patrones (pattern matching) :**

Los chatbots basados en reglas comparan la entrada del usuario con un patrón de reglas y seleccionan una respuesta predefinida que procede de un conjunto de respuestas usando algoritmos de coincidencia de patrones.

En general, los sistemas basados en reglas no crean respuestas nuevas. Cuantas más reglas haya disponibles en la base de datos, más exactamente podrá responder el chatbot.

Los chatbots que usan este enfoque suelen tener un tiempo de respuesta rápido debido a que no realizan ningún análisis sintáctico o semántico profundo del texto de entrada del usuario. No obstante, estos chatbots requieren una cantidad importante de reglas para funcionar correctamente y poseen una gran dificultad en manejar errores sintácticos o gramaticales en las respuestas del usuario. Además, las respuestas tienden a ser repetitivas y carecen de originalidad, ya que estos chatbots no consideran el contexto del diálogo.

Algunos lenguajes más comunes para la implementación de chatbots que usan este enfoque son *AIML*, *Rivescript*, *Chatscript*. Además, algunos ejemplos de chatbots que utilizan este enfoque son ELIZA, ALICE y JABBERWACKY, los cuales se explicarán en las secciones siguientes.

```
<category>
  <pattern>HELLO </pattern>
  <template>
    <random>
      <li> Hi! What's your name? </li>
      <li> Hello, How are you? </li>
      <li> Hello! </li>
    </random>
  </template>
</category>

<category>
  <pattern>MYNAME IS * </pattern>
  <template>Nice to meet you <set name="nameUser"> <star/> </set></template>
</category>

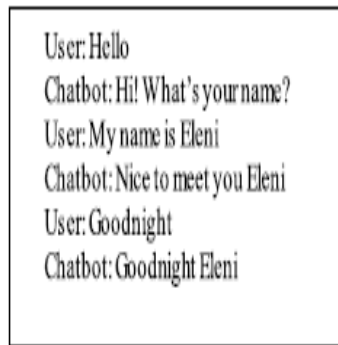
<category>
  <pattern>NIGHT </pattern>
  <template>Good night <get name="nameUser"/> </template>
</category>

<category>
  <pattern>_NIGHT </pattern>
  <template><srai> NIGHT </srai> </template>
</category>

<category>
  <pattern>NIGHT * </pattern>
  <template><srai> NIGHT </srai> </template>
</category>

<category>
  <pattern>_NIGHT * </pattern>
  <template><srai> NIGHT </srai> </template>
</category>
```

Figura 2.1: Ejemplo de pattern matching (AIML)  
[1]



```
User: Hello
Chatbot: Hi! What's your name?
User: My name is Eleni
Chatbot: Nice to meet you Eleni
User: Goodnight
Chatbot: Goodnight Eleni
```

**Figura 2.2:** Ejemplo de conversación

[1]

### **Aprendizaje Automático (Machine Learning) :**

Los chatbots basados en Aprendizaje Automático extraen y analizan el contenido de la entrada del usuario usando *Procesamiento del Lenguaje Natural (NLP)* y pueden mejorar su comprensión a medida que interactúan con los usuarios. A diferencia del enfoque anterior, estos sistemas consideran todo el contexto del diálogo y no proporcionan una respuesta predefinida. [2]

No obstante, su implementación requiere conjuntos de entrenamiento muy extenso, lo cual puede ser muy complicado, especialmente si no se disponen de conjuntos de datos adecuados.

A continuación, se describen algunas de las técnicas comúnmente utilizadas en la implementación de los chatbots basados en Aprendizaje Automático:

**El Procesamiento del Lenguaje Natural (NLP o PNL)** es un subcampo de la IA que permite a los sistemas comprender y generar lenguaje humano de manera más efectiva. Dichos algoritmos se basan en modelos estadísticos que aprenden a reconocer patrones en los datos. Los elementos clave que actúan en el procesamiento de lenguaje natural son los siguientes:

- **Análisis léxico:** identificar todas las diferentes palabras de un texto y comprender su significado.
- **Análisis sintáctico:** analizar la forma en que estas palabras se combinan para formar frases y oraciones.
- **Análisis semántico:** determinación de las relaciones entre palabras y conceptos.
- **Análisis pragmático:** comprender cómo se utiliza el lenguaje en diferentes situaciones.

[3]

La **Comprensión del Lenguaje Natural (NLU)** es un conjunto de técnicas y algoritmos que permite analizar las entradas del lenguaje natural proporcionado por el usuario y comprender el contexto y contenido de este texto, con el objetivo de

que los chatbots generen respuestas precisas. NLU permite facilitar la clasificación de intenciones y la extracción de entidades, tomando en cuenta la información contextual.

**Redes Neuronales** : Los chatbots basados en recuperación y generación utilizan varios tipos de Redes Neuronales. Para generar una respuesta adecuada, se realiza el proceso de **word embeddings**, que implica procesar la entrada del usuario, calcular sus representaciones vectoriales y utilizarlas como características en la red neuronal que genera la respuesta.

Las **Redes Neuronales Recurrentes (RNN)** sirven para que los chatbots puedan considerar el contexto previo en una conversación. En este caso, tanto la nueva entrada del usuario como la información anterior alimentan las neuronas.

[1]

### 2.4.3. Primeros chatbots

La evolución de los chatbots ha sido impulsada por innovaciones técnicas que han permitido avances significativos en inteligencia artificial y procesamiento del lenguaje natural. A continuación, exploramos métodos específicos utilizados por pioneros como ELIZA o ALICE, destacando cómo estas primeras implementaciones emplearon técnicas básicas de NLP para simular interacciones humanas.

#### ELIZA (1966)

Uno de los primeros chatbots en utilizar técnicas de procesamiento del lenguaje natural. ELIZA se basaba en el reconocimiento de patrones para analizar las entradas de los usuarios y generar respuestas en función de reglas predefinidas. Este enfoque, conocido como "transformación de entradas", permitía a ELIZA detectar palabras clave y estructuras gramaticales simples para emular conversaciones básicas. Por ejemplo, ELIZA podía identificar preguntas y afirmaciones en las respuestas del usuario y responder de manera apropiada utilizando patrones de texto predefinidos.

Debido a que ELIZA se basaba en la coincidencia de patrones y de un sistema de respuesta basado en plantillas, tenía una limitación más que evidente debido a que su conocimiento era restringido. Además, su capacidad para mantener conversaciones extensas es limitada porque carece de habilidades para entender el contexto de la interacción.

Aun así, ELIZA marcó el inicio del desarrollo de chatbots y sistemas de procesamiento del lenguaje natural, estableciendo las bases y fundamentos para futuras investigaciones en inteligencia artificial y en el desarrollo de interfaces de usuario conversacionales.



### **Jabberwacky (1988):**

Fue uno de los primeros chatbots en usar el ámbito del aprendizaje automático (ML). A diferencia de los precedentes chatbots, que se basaban en reglas predefinidas, Jabberwacky adoptó un enfoque donde no se necesitaba escribir reglas. Sin embargo, solo requería datos para entrenarse y mejorar su capacidad de respuesta. [4]

Jabberwacky fue desarrollado en CleverScript, un lenguaje basado en hojas de cálculo que utilizaba la concordancia de patrones contextuales para responder a las peticiones del usuario basándose en conversaciones anteriores.

A pesar del avance que supuso en el desarrollo de chatbots a partir del aprendizaje automático, Jabberwacky tenía limitaciones muy notables. La velocidad de respuesta del chatbot era lenta y tenía dificultades para manejar grandes volúmenes de usuarios al mismo tiempo.

### **A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) (1995)**

Este chatbot fue inspirado en ELIZA. Se basa en el emparejamiento de patrones (*Pattern Matching*) extenso que le permitía tener una capacidad de discusión sobre cualquier tema.

ALICE se desarrolló con un nuevo lenguaje creado llamado *Artificial Intelligence Markup Language (AIML)*, el cual es un lenguaje de marcado basado en XML que contiene alrededor de 41.000 reglas (plantillas), que hacen que el chatbot responda a las consultas realizadas por el usuario de manera precisa. Además los usuarios podían añadir más reglas, lo que contribuía a mejorar la precisión y la inteligencia del chatbot a lo largo del tiempo para que el chatbot sea cada vez más preciso e inteligente.

Sin embargo, una posible limitación sería su capacidad de generar respuestas verdaderamente profundas y creativas, debido a su enfoque basado en reglas predefinidas. Además, no podía generar respuestas humanas que expresaran emociones o actitudes. [5]

### **SmarterChild (2001) :**

Fue un chatbot revolucionario debido a que fue el primer chatbot que podía ayudar a los usuarios con tareas cotidianas, proporcionando información de bases de datos sobre resultados deportivos, noticias, tiempo, entre otros aspectos relevantes. Dicho chatbot estaba disponible en varias plataformas de mensajería instantánea incluyendo AOL Instant Messenger (AIM) y Windows Live Messenger (MSN).

Además, SmarterChild tenía un alto grado de comprensión de lenguaje natural (NLU), lo que permitía responder a las consultas de forma conversacional.

[1] [6]

#### 2.4.4. Chatbots modernos

Tras la aparición del aprendizaje profundo y de las redes neuronales, el campo de los chatbots experimentó una evolución radical. Estas tecnologías han redefinido las facultades de los chatbots al permitirles comprender, procesar y generar respuestas más precisas y con mejor contexto. Algunos de los chatbots modernos más relevantes son:

##### Siri (2011) :

Fue el pionero en materia de asistentes personales. Este chatbot fue diseñado para atender solicitudes y preguntas en lenguaje natural, brindando a los usuarios información, ejecutando tareas y facilitando interacciones basadas en comandos de voz . A pesar de que Siri es bastante avanzado, requiere conexión Internet por lo que puede resultar limitado en algunos aspectos.

Siri utiliza una variedad de algoritmos avanzados para comprender y responder a las consultas de los usuarios de manera efectiva tales como algoritmos de PNL avanzados, algoritmos de reconocimiento de voz para convertir voz en texto y algoritmos de aprendizaje automático para ofrecer respuestas más precisas y personalizadas. [7]

##### ChatGPT (2020) :

Chatbot desarrollado por OpenAI. ChatGPT es uno de los modelos más avanzados en la actualidad debido a la comprensión del lenguaje natural y respuestas contextualmente relevantes en conversaciones basadas en texto. ChatGPT, representa un avance importante en el campo de la inteligencia artificial, alejándose de los métodos tradicionales basados en redes neuronales recurrentes (RNN), como LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit). En cambio, ChatGPT utiliza **mecanismos de atención** y **modelos transformadores**, marcando un hito importante en el desarrollo de chatbots.

A diferencia de los RNN que presentan limitaciones tales como la desaparición de los gradientes en secuencias muy largas, lo que provocaban dificultades en el aprendizaje efectivo de dependencias a largo plazo, los mecanismos de atención, denominados **transformadores**, han revolucionado el campo del procesamiento del lenguaje natural al demostrar un rendimiento óptimo en diversas tareas, como traducción automática, clasificación, generación de texto y más. [8]

El modelo ChatGPT se basa en un tipo específico de transformador conocido como **transformador solo decodificador** (*Decoder-only transformers*). Este enfoque está diseñado para generar respuestas de manera eficiente en conversaciones basadas en texto. Por ejemplo, ChatGPT utiliza el modelo GPT-3 (Generative Pre-trained Transformer 3), que es uno de los Modelos de Lenguaje Grande (LLM) más avanzados disponibles. Estos modelos, como GPT-3, están especialmente diseñados para comprender y generar texto en lenguaje humano. [9]

Además de Siri y ChatGPT, existen otros chatbots modernos relevantes tales como **Alexa** o **Gemini**. Aunque dichos chatbots presentan características notables, se investiga en este proyecto principalmente sobre Siri y ChatGPT debido a sus similitudes en características y eficiencia con los demás chatbots.

#### 2.4.5. Fine-Tuning y Prompt Engineering

Después de haber realizado un análisis de cómo funcionan los chatbots modernos se ha encontrado una solución más viable para la realización de nuestro proyecto cumpliendo con el número horas propuestas y las fechas de entrega. Dicha solución se basa en utilizar un *Modelo de Lenguaje Preentrenado (LLM)* y aplicarle *fine-tuning* para adaptarlo al entorno de la hostelería.

Un *Modelo de Lenguaje de gran tamaño (LLM)* es un modelo de aprendizaje profundo muy grande que se entrena previamente con grandes cantidades de datos. Dichos modelos se pueden usar para realizar tareas como generación de texto, traducción automática, redacción de resúmenes, generación de imágenes a partir de textos, codificación automática, chatbots o IA conversacional.

**Fine-Tuning (Ajuste Fino)** consiste en tomar un LLM y reentrenarlo aprovechando la mayor cantidad de conocimiento posible del modelo, adaptándolo a una tarea concreta. El objetivo del ajuste es mantener las capacidades originales de un modelo previamente entrenado y al mismo tiempo adaptarlo para casos de uso más especializados. Para poder realizar Fine-Tuning a un LLM, una estrategia para ello sería seguir los siguientes pasos:

1. **Seleccionar un modelo previamente entrenado:** Elegir un modelo que se ajusta a la naturaleza de la tarea que se quiere realizar. En el caso este proyecto, se debe seleccionar un LLM adecuado para tareas de conversación y comprensión del lenguaje natural.
2. **Ajustar la arquitectura:** Se tiene que modificar la arquitectura del LLM para adaptarse a los objetivos del proyecto. Para ello, se tendría que realizar modificar capas del modelo.
3. **Congelar o descongelar capas:** Congelar una capa significa evitar la actualización de sus pesos durante el proceso de ajuste. Esto es útil si las capas inferiores ya han aprendido características generales útiles para la tarea. Por otro lado, descongelar permite que las capas se adapten a nuevos datos durante el Fine-Tuning, mejorando así el rendimiento del modelo para la tarea específica
4. **Entrenamiento:** Tras modificar el modelo, se debe alimentarlo con datos específicos de esa tarea para que pueda aprender y mejorar su desempeño.
5. **Estrategias de ajuste:** Tras realizar todo lo anterior, un último paso sería reajustar los parámetros y las estrategias de entrenamiento para que se ajuste de forma óptima al objetivo que se busca. [10]

Como se ha podido ver arriba, el proceso de Fine-Tuning puede ser costoso en términos de recursos computacionales y tiempo de entrenamiento.

Una alternativa rentable que requiere un costo menor que el del Fine-Tuning es el **Prompt Engineering**. Esta técnica consiste en proporcionar detalladamente prompts (peticiones) con ejemplos concretos al modelo para guiarlo hacia el comportamiento deseado sin necesidad de realizar ningún ajuste en su arquitectura.[11]

```
system_message_prompt = SystemMessagePromptTemplate.from_template(
    """
    You are a chatbot tasked with responding to questions about the documentation of the LangChain library and project.

    You should never answer a question with a question, and you should always respond with the most relevant documentation page.

    Do not answer questions that are not about the LangChain library or project.

    Given a question, you should respond with the most relevant documentation page by following the relevant context below:\n
    {context}
    """
)
```

**Figura 2.3:** Ejemplo de preprompt utilizado en Prompt Engineering

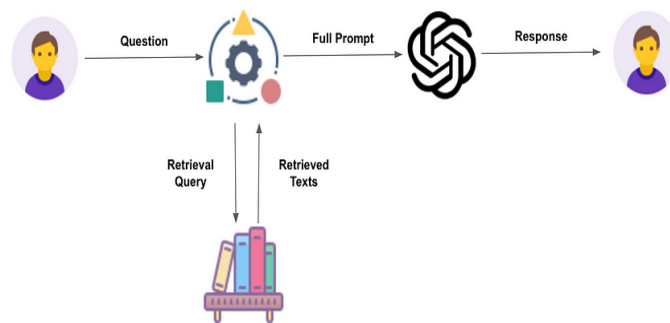
En la figura 2.3 podemos ver un ejemplo de preprompt donde el chatbot tiene la tarea específica de responder preguntas sobre la documentación de la biblioteca *LangChain* y el proyecto asociado. Como se puede constatar, el chatbot solo deberá responder a preguntas relacionadas con ese tema, mateniéndose siempre dentro del contexto especificado.

Tras entender estos conceptos y después de una búsqueda exhaustiva en Internet, se ha encontrado una cantidad considerable de páginas que aportan información y técnicas de optimización de cómo crear un asistente virtual usando LLMs ya existentes y ajustándolos:

- [langchain.chatbot.rag](#): Este proyecto es sobre un asistente virtual usando GPT-4 como LLM en el que se añaden *pre-peticiones (Prompt Engineering)* explicándole detalladamente lo que debe y lo que no debe hacer. Además, se incrusta una página web externa al chatbot que le servirá para responder únicamente a las solicitudes relacionadas con la información de esa página.
- [Building a Chatbot using Your Documents with Langchain](#): Este es un artículo bastante similar al proyecto anterior, que explica más detalladamente cómo funcionan los chatbots usando documentos externos a través de **RAG (Generación Aumentada de Recuperación)**. RAG permite que un sistema no solo confíe en su conocimiento previo, sino que también sea capaz de buscar información relevante de una base de conocimientos. En otras palabras, RAG

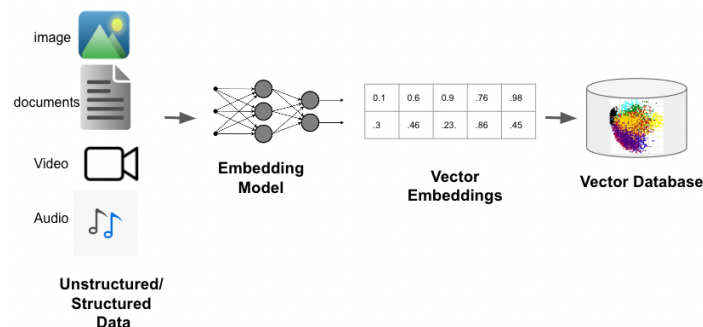
es una técnica que permite mejorar la calidad de respuesta de los LLM usando un archivo de datos externos. Esto se logra mediante dos fases: [12]

- **Recuperación:** Esta fase consiste en la búsqueda y localización de fragmentos relevantes en la base de datos externa. Dichos fragmentos pueden ser documentos de Internet u otro tipo de documentos como CSV o PDF.
- **Generación de contenido :** Durante esta fase, el modelo de lenguaje utiliza tanto la consulta original como la información adicional de los fragmentos relevantes para generar una respuesta más precisa y contextual. [13]



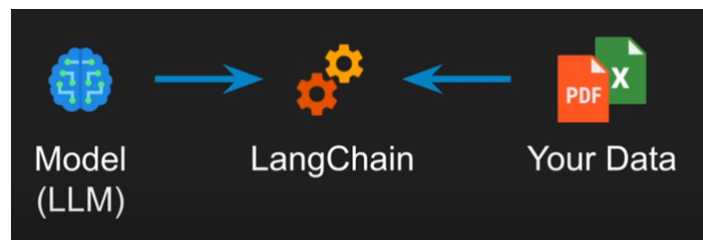
**Figura 2.4:** Ejemplo de funcionamiento de RAG

Además, se habla también de *base de datos vectoriales*, los cuales son sistemas de almacenamiento diseñados específicamente para guardar documentos externos de manera eficiente. En lugar de almacenar los documentos completos, utiliza *incrustaciones (embeddings)* para representar cada documento como un vector numérico de características significativas. Estas incrustaciones permiten una recuperación rápida y precisa de los documentos relevantes para las consultas del chatbot, lo que mejora su capacidad para comprender y responder adecuadamente a las solicitudes de los usuarios.



**Figura 2.5:** Representación de embeddings para la recuperación de documentos.

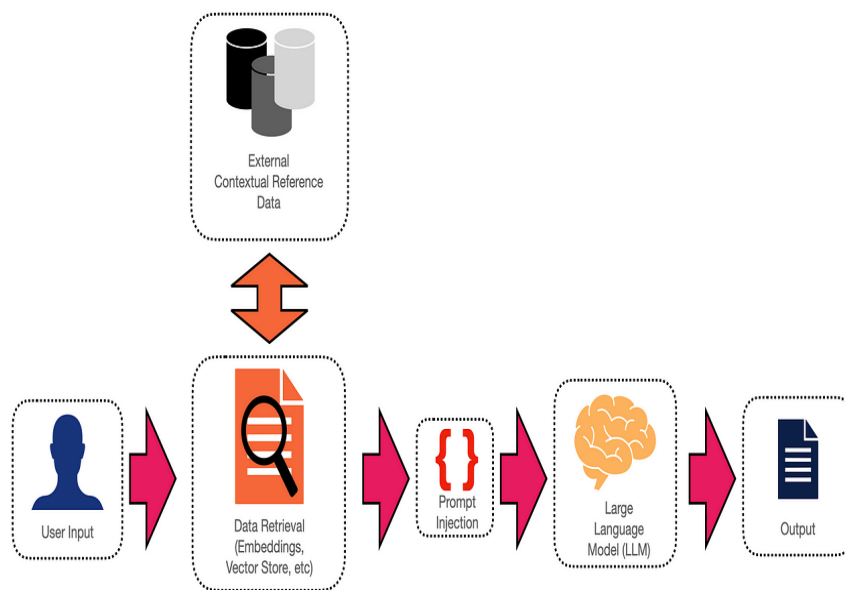
En los dos artículos anteriores, así como en otros, emerge un framework común, *Langchain*, el cual es un framework muy reciente que permite para conectar proyectos con LLMs y proporcionar herramientas para mejorar la información generada por estos modelos, lo que lo convierte en una herramienta valiosa para crear chatbots conversacionales, informativos y personalizados. Gracias a ello, podemos introducir datos estructurados, como archivos PDF o CSV, que servirán como base de conocimiento para entrenar y mejorar la capacidad del chatbot para responder de manera inteligente a las consultas de los usuarios. Estos datos permiten al chatbot comprender mejor el contexto y proporcionar respuestas más precisas y relevantes.



**Figura 2.6:** Representación general de Langchain.  
[14]

Además, para poder conseguir que el chatbot busque información más eficientemente del documento, es recomendable dividirlo en *chunks*. *Chunking* es el proceso de dividir el contenido en partes más pequeñas y manejables, haciéndolo más fácil de manejar desde una perspectiva computacional para así almacenarlos en la base de datos vectorial. [14]

Después de haber realizado varias pruebas en nuestra computadora personal y de llevar a cabo un estudio más profundo sobre cómo estas herramientas trataban los datos introducidos y las peticiones de los usuarios, clasificaban y analizaban toda la información, y cómo se generaban las respuestas (analizando también la velocidad de todo el proceso), se ha decidido validar estos métodos para poder integrarlos a nuestro proyecto y adaptarlos a nuestro objetivo final.



**Figura 2.7:** Implementación de una solución RAG usando bases de datos vectoriales.

#### 2.4.6. Herramientas para la interacción por voz

Tras concluir cómo abordar la parte de la generación de texto mediante Inteligencia Artificial para el proyecto, el siguiente paso fue investigar cómo capturar la voz de un usuario y transformarlo a texto filtrando lo más posible el ruido de fondo que se podría presentar.

Para poder habilitar interacciones de voz en el chatbot, es esencial implementar un sistema de reconocimiento de voz eficaz que convierta la entrada de audio del usuario en texto comprensible para nuestro sistema. En este subapartado, se detallará la investigación y selección de herramientas para poder alcanzar el objetivo deseado.

Durante la investigación, se han evaluado diversas posibilidades para desarrollar el reconocimiento de voz en el chatbot, analizando las necesidades específicas del proyecto, que incluyen la integración con dispositivos móviles y la minimización del ruido ambiental.

A continuación, se detallan dos métodos que se utilizarán para la interacción por voz:

- **Sistema de reconocimiento de voz (STT):** Script que se encarga de capturar el audio del usuario y transformarlo en texto para que sea procesado por el asistente virtual.
- **Sistema de conversión de texto a voz (TTS):** Script que se encarga en convertir un texto escrito en voz.



Inicialmente, se identificó la librería *speech\_recognition* de Python como una solución adecuada para realizar pruebas en el entorno de desarrollo en nuestra computadora. Esta librería facilita la captura de voz del usuario y su conversión a texto utilizando una API de reconocimiento de voz disponible. Internamente, la librería *speech\_recognition* se conecta con los servicios de reconocimiento de voz de Google para realizar la transformación de voz a texto de manera precisa.[15]

Además de *speech\_recognition*, se identificaron varias librerías complementarias que serán útiles en el desarrollo del chatbot. Una de ellas es *PyAudio*, utilizada para la captura y reproducción de audio en tiempo real, compatible con diversas plataformas. También se ha considerado *gTTS (Google Text-to-Speech)*, que permite generar voz a partir de texto utilizando servicios de síntesis de voz de Google. Por último, se ha escogido *pygame*, la cual es una herramienta versátil para la reproducción y manipulación de audio, útil en la implementación de la salida de audio del chatbot.

Dado que el objetivo final es desarrollar un chatbot para dispositivos móviles, se requiere además encontrar una librería que permita el uso del micrófono de los móviles. Para ello, se han encontrado librerías de Google compatibles con aplicaciones móviles desarrolladas con *Flutter*. Dichas librerías son *flutter\_tts*, que servirá para transformar las respuestas del chatbot en voz, y *speech\_to\_text\_google\_dialog*, que capta la voz del usuario y la convierte en texto.

## 2.5. Tabla de requisitos

A partir de los objetivos mencionados anteriormente en la sección 2.3, profundizaremos en dichos objetivos detallando todos los requisitos necesarios para garantizar la máxima funcionalidad del sistema. Inicialmente, se explicará de manera clara y concisa cada categoría de requisitos que se emplearán:

- **Requisitos de Información (RI):** Describe qué información debe almacenarse en el sistema para cumplir con los objetivos de nivel superior.
- **Requisitos Funcionales (RF):** Define los servicios que proporcionará el sistema para ayudar a los usuarios a alcanzar sus objetivos.
- **Requisitos no Funcionales (RNF):** Describe los atributos del sistema que no están relacionados directamente con las funciones específicas.
- **Reglas de Negocio (RN):** Describen características o restricciones que afectan las operaciones o actividades de un negocio.

Tras haber definido los diferentes tipos de requisitos, comenzaremos a clasificar y nombrar cada uno de los requisitos que tendrá el proyecto.



### 2.5.1. Requisitos de Información

<b>RI-01</b>	Platos del menú
<b>Como</b>	Encargado del restaurante
<b>Quiero</b>	disponer de la información de cada plato y bebidas del menú, incluido nombre del plato, descripción, categoría y variación.
<b>Para</b>	poder gestionar de una forma adecuada la carta del restaurante.

**Tabla 2.1:** Platos del menú: Requisito de Información

<b>RI-02</b>	Variación de cada plato
<b>Como</b>	Encargado del restaurante
<b>Quiero</b>	contar con la información sobre las variaciones de cada plato, que incluiría los diferentes tamaños disponibles junto con sus respectivos precios.
<b>Para</b>	poder gestionar de una forma adecuada la carta del restaurante.

**Tabla 2.2:** Variación de cada plato: Requisito de Información

<b>RI-03</b>	Pedido de plato
<b>Como</b>	Trabajador del restaurante
<b>Quiero</b>	contar con la información de cada pedido que hace el cliente, incluyendo un número de pedido único, la lista de platos ordenados y el precio total del pedido.
<b>Para</b>	gestionar eficientemente los pedidos.

**Tabla 2.3:** Detalles de cada pedido: Requisito de Información

<b>RI-03</b>	Detalles de cada plato del pedido
<b>Como</b>	Trabajador del restaurante
<b>Quiero</b>	tener acceso a los detalles de cada plato incluido en un pedido, incluyendo el tipo de plato, el número de orden, el nombre del plato, el tamaño, el precio unitario y la cantidad solicitada.
<b>Para</b>	preparar los pedidos de manera precisa.

**Tabla 2.4:** Detalles de cada plato del pedido: Requisito de Información

<b>RI-05</b>	Empleados
<b>Como</b>	Administrador de la empresa
<b>Quiero</b>	disponer de la información de cada empleado; en concreto, del nombre, apellido, correo electrónico, teléfono, nombre de usuario y contraseña.
<b>Para</b>	gestionar el acceso y las funciones de cada usuario en el sistema.

**Tabla 2.5:** Información de los empleados: Requisito de Información

<b>RI-06</b>	Personalización restaurante
<b>Como</b>	Administrador de la empresa
<b>Quiero</b>	disponer de los datos de los elementos de personalización de la aplicación móvil , en concreto el nombre del restaurante, logo principal y el logo secundario.
<b>Para</b>	tener la posibilidad de personalizar la aplicación móvil según la temática del restaurante .

**Tabla 2.6:** Información de personalización: Requisito de Información

## 2.5.2. Requisitos Funcionales

<b>RF-01</b>	Registro de pedidos por voz
<b>Como</b>	Cliente
<b>Quiero</b>	poder comunicarme con el asistente virtual mediante comandas vocales en la aplicación móvil.
<b>Para</b>	poder tener información y hacer pedidos de manera eficiente, evitando tener que escribir.

**Tabla 2.7:** Registro de pedidos por voz: Requisito Funcional

<b>RF-02</b>	Respuesta del asistente de voz
<b>Como</b>	Cliente
<b>Quiero</b>	que el chatbot pueda proporcionar respuestas tanto en formato de voz como escritas, manteniendo un historial de la conversación y asegurando respuestas contextualizadas.
<b>Para</b>	mejorar la experiencia del usuario al realizar pedidos y obtener información sobre los platos.

**Tabla 2.8:** Respuesta del asistente de voz: Requisito Funcional

<b>RF-03</b>	Interrupción del asistente de voz
<b>Como</b>	Cliente
<b>Quiero</b>	tener la capacidad de interrumpir al chatbot cuando esté respondiendo en la aplicación móvil.
<b>Para</b>	garantizar una experiencia de usuario fluida y controlada.

**Tabla 2.9:** Interrupción del asistente de voz: Requisito Funcional

<b>RF-04</b>	Reinicio de conversación con el chatbot en la aplicación móvil
<b>Como</b>	Cliente
<b>Quiero</b>	poder reiniciar la conversación con el chatbot pulsando un botón de reinicio.
<b>Para</b>	borrar el historial de la conversación y comenzar de nuevo, permitiéndome corregir errores en los pedidos o solicitudes realizadas. .

**Tabla 2.10:** Reinicio de la conversación con el asistente de voz: Requisito Funcional

<b>RF-05</b>	Generación del recibo del pedido
<b>Como</b>	Cliente, personal y administrador
<b>Quiero</b>	generar tickets detallados una vez el cliente haga su pedido.
<b>Para</b>	tener información detallada del pedido del cliente.

**Tabla 2.11:** Generación del recibo del pedido: Requisito Funcional

<b>RF-06</b>	Notificación a cocina de nuevos pedidos
<b>Como</b>	Personal de cocina
<b>Quiero</b>	recibir notificaciones automáticas en la página de la aplicación web cuando se reciban nuevos pedidos, excepto en la página de inicio de sesión.
<b>Para</b>	agilizar el proceso de preparación y entrega de pedidos.

**Tabla 2.12:** Notificación a Cocina de Nuevos Pedidos: Requisito Funcional

<b>RF-07</b>	Eliminación de pedidos de la base de datos
<b>Como</b>	Personal de cocina
<b>Quiero</b>	tener la capacidad de eliminar los pedidos una vez estén preparados y listos para su entrega.
<b>Para</b>	mantener la base de datos de pedidos actualizada y organizada.

**Tabla 2.13:** Eliminación de pedidos de la base de datos: Requisito Funcional

<b>RF-08</b>	Gestión de platos del menú
<b>Como</b>	Administrador de empresa
<b>Quiero</b>	poder actualizar el menú añadiendo, eliminando o modificando platos en la página web.
<b>Para</b>	tener un sistema flexible para modificar el menú.

**Tabla 2.14:** Gestión de platos del menú: Requisito Funcional

<b>RF-09</b>	Configuración del restaurante
<b>Como</b>	Administrador de empresa
<b>Quiero</b>	poder cambiar el nombre y el logo del restaurante en la página web.
<b>Para</b>	tener un sistema flexible que permita modificar fácilmente la oferta del restaurante y su identidad visual.

**Tabla 2.15:** Configuración del restaurante: Requisito Funcional

<b>RF-10</b>	Conexión a la aplicación web
<b>Como</b>	Personal de cocina
<b>Quiero</b>	introducir el nombre de usuario y contraseña en la página web.
<b>Para</b>	ver notificaciones de nuevos pedidos, gestionarlos y gestionar el menú.

**Tabla 2.16:** Conexión a la aplicación web: Requisito Funcional

<b>RF-11</b>	Creación de nuevos empleados en la aplicación web
<b>Como</b>	Administrador
<b>Quiero</b>	añadir empleados creando un nombre de usuario no existente, proporcionando su nombre, apellidos, email, número de teléfono y nombre de usuario.
<b>Para</b>	asignar el acceso a la página web a los empleados.

**Tabla 2.17:** Creación de nuevos empleados en la aplicación web: Requisito Funcional

<b>RF-12</b>	Borrar empleados en la aplicación web
<b>Como</b>	Administrador
<b>Quiero</b>	que, a partir de una lista de todos los empleados, pueda borrar cualquiera.
<b>Para</b>	gestionar eficientemente la lista de empleados y sus accesos a la aplicación web.

**Tabla 2.18:** Borrar empleados en la aplicación web: Requisito Funcional

<b>RF-13</b>	Modificación de la contraseña por el empleado
<b>Como</b>	Empleado
<b>Quiero</b>	modificar mi propia contraseña en cualquier momento.
<b>Para</b>	mantener la seguridad de mi cuenta en la aplicación web.

**Tabla 2.19:** Modificación de la contraseña por el empleado: Requisito Funcional

### 2.5.3. Requisitos No Funcionales

<b>RNF-01</b>	Funcionamiento completo en Android
<b>Como</b>	Administrador de empresa
<b>Quiero</b>	que la aplicación se pueda ejecutar en cualquier dispositivo Android y que sea responsive.
<b>Para</b>	aumentar el número de usuarios.

**Tabla 2.20:** Funcionamiento completo en Android: Requisito No Funcional

<b>RNF-02</b>	Interfaz móvil sencilla e intuitiva
<b>Como</b>	Administrador de empresa
<b>Quiero</b>	que la interfaz móvil sea fácil de usar y comprender para todos los usuarios capaces de escuchar, hablar y leer.
<b>Para</b>	facilitar la accesibilidad a los clientes.

**Tabla 2.21:** Interfaz móvil sencilla e intuitiva: Requisito No Funcional

<b>RNF-03</b>	Interfaz web sencilla e intuitiva
<b>Como</b>	Administrador de empresa
<b>Quiero</b>	que la interfaz web sea fácil de usar y comprender para todos los trabajadores.
<b>Para</b>	facilitar la accesibilidad a los empleados.

**Tabla 2.22:** Interfaz web sencilla e intuitiva: Requisito No Funcional

<b>RNF-04</b>	Rendimiento del chatbot
<b>Como</b>	Administrador de empresa
<b>Quiero</b>	garantizar un tiempo de respuesta aceptable para el chatbot, evitando respuestas lentas a las peticiones del usuario.
<b>Para</b>	ofrecer una experiencia de usuario eficiente.

**Tabla 2.23:** Rendimiento del chatbot: Requisito No Funcional

#### 2.5.4. Reglas de Negocio

<b>RN-01</b>	Unicidad de nombres de platos
	No se permitirá la duplicación de nombres de platos dentro del menú.

**Tabla 2.24:** Unicidad de nombres de platos: Regla de Negocio

<b>RN-02</b>	Restricción de acceso a la aplicación web
	Solo el personal autorizado, identificado con un nombre de usuario y contraseña válidos, tendrá acceso a la aplicación web del sistema.

**Tabla 2.25:** Restricción de acceso a la aplicación web: Regla de Negocio

<b>RN-03</b>	Validación de las peticiones del cliente
	Antes de registrar un pedido, el sistema verificará la validez de los datos proporcionados, como la existencia de los platos solicitados.

**Tabla 2.26:** Validación de las peticiones del cliente: Regla de Negocio

<b>RN-04</b>	Contexto de respuesta del sistema
	El sistema responderá únicamente a preguntas y consultas relacionadas con el menú, los platos disponibles, la realización de pedidos y temas pertenecientes al restaurante.

**Tabla 2.27:** Contexto de respuesta del sistema: Regla de Negocio

<b>RN-05</b>	Unicidad de correo electrónico y nombre de usuario al crear un nuevo empleado en la página web
	Al crear un nuevo empleado en la aplicación web, se verificará que tanto el correo electrónico como el nombre de usuario proporcionados sean únicos en el sistema.

**Tabla 2.28:** Unicidad de correo electrónico y nombre de usuario: Regla de Negocio

<b>RN-06</b>	Unicidad del número de pedido al realizar un nuevo pedido
	Al realizar un nuevo pedido, el ticket que se imprimirá contendrá un número de pedido único.

**Tabla 2.29:** Unicidad del número de pedido: Regla de Negocio

<b>RN-07</b>	Restricción de autoeliminación de empleados en la aplicación web
	Un empleado no tiene permiso para borrarse a sí mismo en la página web del sistema. Sin embargo, tendrá la capacidad de eliminar a otros empleados.

**Tabla 2.30:** Restricción de autoeliminación de empleados: Regla de Negocio

<b>RN-08</b>	Restricción al añadir un nuevo plato en la página web
	Para añadir un nuevo plato, se tiene que rellenar de manera obligatoria los campos de nombre, categoría y variaciones. Además, el nombre de cada plato debe ser único y el precio debe ser mayor que 0.

**Tabla 2.31:** Restricción al añadir un nuevo plato: Regla de Negocio

## 2.6. Alcance del proyecto

Este apartado tiene como objetivo definir claramente los límites y alcances que tendrá el desarrollo de nuestro proyecto. A continuación, se exponen las funcionalidades e implementaciones que se incluirán, así como aquellos aspectos y mejoras que quedarán fuera del alcance inicial con el fin de garantizar la entrega en el plazo estimado:

### 2.6.1. Inclusiones

- Facilitar el proceso de comandas, permitiendo a los usuarios interactuar de manera intuitiva.
- Introducir el reconocimiento de voz para agilizar y mejorar la precisión en la captura de solicitudes de los clientes, proporcionando una experiencia fluida y sin complicaciones.
- Prestar especial atención al diseño de la interfaz de usuario para ofrecer una experiencia cómoda a los clientes, personal de cocina y administradores.
- Elaborar una memoria completa que documente cada etapa del desarrollo, proporcionando una visión profunda de los problemas encontrados y de las decisiones tomadas e implementadas.
- Justificar cada elección tecnológica, diseño y funcionalidad en la memoria, creando un registro comprensible del proceso de desarrollo.
- Realizar y justificar un mínimo de 300 horas de trabajo, garantizando la implementación de todas las funcionalidades requeridas y realizando pruebas rigurosas para asegurar la calidad del sistema desarrollado.
- Completar todas las funciones requeridas según los requisitos anteriores.

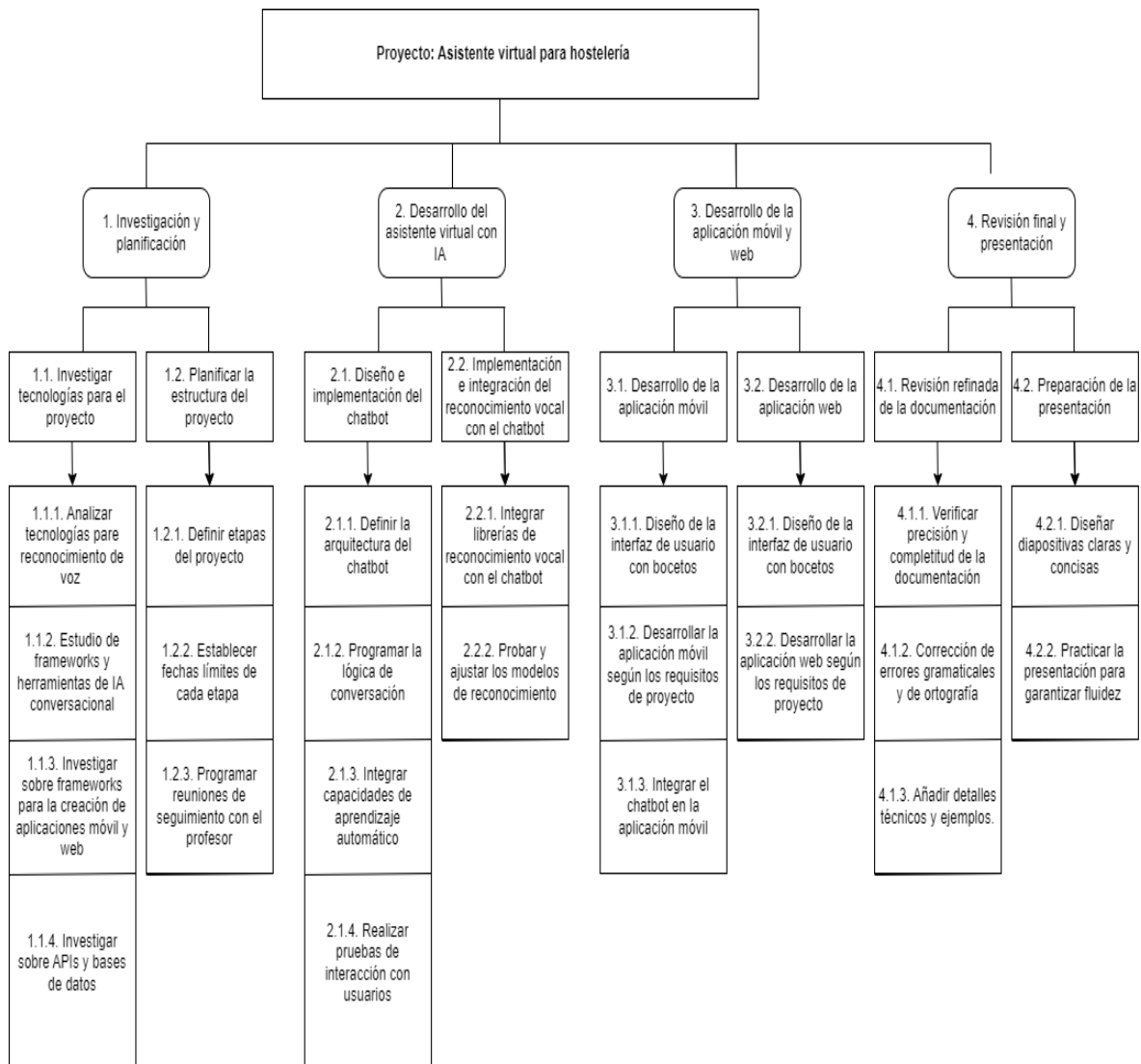
### **2.6.2. Exclusiones**

- No se dará la aplicación a otros idiomas distintos al castellano.
- La validación en entornos reales quedará fuera del alcance por limitaciones de tiempo/recursos.
- No se publicará en tiendas de aplicaciones.
- No se realizará la aplicación en emuladores iOS.
- No se implementarán funcionalidades de pagos móviles en esta fase del proyecto.
- No se contempla la personalización de colores según los gustos del usuario.
- En esta primera versión, el asistente virtual dependerá de conexión a Internet para su correcto funcionamiento. Sin conexión, las funcionalidades se verán limitadas.



### 2.6.3. Estructura de Descomposición del Trabajo (EDT)

En este subapartado, se representará de forma jerárquica y visual todas las tareas que se deben realizar en un proyecto mediante una *Estructura de Descomposición del Trabajo (EDT)*. Todo esto se organizará en forma de árbol, donde cada nivel inferior representará tareas más detalladas que contribuyen al logro de los objetivos generales del proyecto.



**Figura 2.8:** Estructura de Descomposición del Trabajo (EDT).

## 2.6.4. Diccionario de la EDT

En este subapartado, se añadirá un *diccionario del proyecto* que proporcionará información detallada sobre cada elemento de la EDT, incluidos los entregables, los hitos, el alcance, la duración y los miembros del equipo asignados.

Antes de adentrarnos en los detalles de los paquetes y subpaquetes del proyecto, es importante entender quiénes forman parte del equipo de desarrollo y quiénes integran el equipo de dirección:

- **Equipo de Desarrollo:** Compuesto por el propio alumno, quien es responsable de llevar a cabo todas las actividades relacionadas con la implementación y desarrollo del proyecto. Este rol implica diseñar, programar y realizar pruebas sobre las distintas funcionalidades, así como integrar tecnologías específicas en el proyecto, como el chatbot y el sistema de reconocimiento vocal.
- **Equipo de Dirección:** Compuesto por el profesor, quien actúa como guía y supervisor del proyecto. Colabora directamente con el alumno en la toma de decisiones estratégicas. Este equipo se encarga de la coordinación general del proyecto, revisando y guiando el proceso de planificación, diseño e implementación, y asegurando que se cumplan los objetivos establecidos.

<b>Nombre del paquete:</b> Investigación y planificación ID: 1.1		<b>Nombre del subpaquete:</b> Investigación de tecnologías para el proyecto	
<b>Responsable:</b> Equipo de desarrollo			
<b>Descripción:</b> identificar y evaluar diversas herramientas, frameworks y librerías disponibles que puedan ser utilizadas para implementar las funcionalidades requeridas			
<b>Duración total:</b> 14 días			
<b>Actividades/Hitos</b>	<b>Duración</b>	<b>Recursos</b>	
1.1.1Análisis de tecnologías y librerías actuales para el reconocimiento de voz	4 días	Equipo de desarrollo	
1.1.2. Estudio de frameworks y herramientas de IA conversacional	4 días	Equipo de desarrollo	
1.1.3. Estudio sobre frameworks para el desarrollo de aplicaciones móviles y aplicaciones web	4 días	Equipo de desarrollo	
1.1.4. Investigar sobre bases de datos y APIs	2 días	Equipo de desarrollo	
1.1.5. Reunión con el profesor	2 h	Equipo de dirección	
<b>Criterios de aceptación:</b> 1. Entendimiento claro y documentado de las tecnologías seleccionadas para el proyecto. 2. Documentación detallada del proceso y decisiones tomadas.			

**Tabla 2.32:** Tabla 1 del diccionario de la EDT.

<b>Nombre del paquete:</b> Investigación y planificación <b>ID:</b> 1.2		<b>Nombre del subpaquete:</b> Planificación de la estructura del proyecto
<b>Responsable:</b> Equipo de dirección		
<b>Descripción:</b> fase de planificación del proyecto, específicamente en la estructura general y la programación de actividades clave para su ejecución.		
<b>Duración total:</b> 2 días		
<b>Actividades/Hitos</b>	<b>Duración</b>	<b>Recursos</b>
1.2.1. Definir etapas del proyecto	1 días	Equipo de dirección
1.2.2. Establecer fechas límites de cada etapa	2 horas	Equipo de dirección
1.2.3. Programar reuniones de seguimiento con el profesor	2 horas	Equipo de dirección
<b>Criterios de aceptación:</b> 1. Comprensión clara y definición detallada de las etapas y actividades clave del proyecto durante la fase de planificación. 2. Documentación exhaustiva que abarque la fase de planificación, incluyendo las decisiones estratégicas tomadas y el cronograma detallado de actividades.		

**Tabla 2.33:** Tabla 2 del diccionario de la EDT.

<b>Nombre del paquete:</b> Desarrollo del asistente virtual con IA <b>ID:</b> 2.1		<b>Nombre del subpaquete:</b> Diseño e implementación del chatbot	
<b>Responsable:</b> Equipo de desarrollo			
<b>Descripción:</b> Implementación del chatbot que formará parte del asistente virtual con IA			
<b>Duración total:</b> 3 semanas			
<b>Actividades/Hitos</b>		<b>Duración</b>	<b>Recursos</b>
2.1.1 Definir arquitectura del chatbot		5 días	Equipo de desarrollo
2.1.2. Programar lógica de conversación		1 semana	Equipo de desarrollo
2.1.3. Integrar capacidades de aprendizaje automático		1 semana	Equipo de desarrollo
2.1.4. Realizar pruebas de interacción con usuarios		2 días	Equipo de desarrollo
2.1.5. Reunión con el profesor		2 h	Equipo de dirección
<b>Criterios de aceptación:</b> 1 Arquitectura bien definida que especifique los componentes principales del chatbot y su funcionamiento general. 2. Integración exitosa de capacidades de aprendizaje automático e implementación efectiva de la lógica de conversación 3. Documentación detallada del diseño y la implementación:			

**Tabla 2.34:** Tabla 3 del diccionario de la EDT.

<b>Nombre del paquete:</b> Desarrollo del asistente virtual con IA <b>ID:</b> 2.2		<b>Nombre del subpaquete:</b> Implementación del sistema de reconocimiento vocal en el chatbot	
<b>Responsable:</b> Equipo de desarrollo			
<b>Descripción:</b> Integrar capacidades de reconocimiento vocal en el asistente virtual con IA.			
<b>Duración total:</b> 1 semana			
<b>Actividades/Hitos</b>		<b>Duración</b>	<b>Recursos</b>
2.2.1 Integrar librerías de reconocimiento vocal con el chatbot		3 días	Equipo de desarrollo
2.2.2. Probar y ajustar los modelos de reconocimiento		3 días	Equipo de desarrollo
2.2.3. Reunión con el profesor		2h	Equipo de desarrollo
<b>Criterios de aceptación:</b> 1. Pruebas satisfactorias y ajuste de modelos de reconocimiento vocal. 2. Documentación detallada del diseño y la implementación.			

**Tabla 2.35:** Tabla 4 del diccionario de la EDT.

<b>Nombre del paquete:</b> Desarrollo de la aplicación móvil y de la aplicación web <b>ID:</b> 3.1		<b>Nombre del subpaquete:</b> Desarrollo de la aplicación móvil	
<b>Responsable:</b> Equipo de desarrollo			
<b>Descripción:</b> Diseñar y desarrollar la aplicación móvil de acuerdo con los requisitos del proyecto.			
<b>Duración total:</b> 4 semanas			
<b>Actividades/Hitos</b>	<b>Duración</b>	<b>Recursos</b>	
3.1.1 Diseño de la interfaz de usuario con bocetos	3 días	Equipo de desarrollo	
3.1.2. Desarrollar la aplicación móvil según los requisitos del proyecto	3 semanas	Equipo de desarrollo	
3.1.3. Integración del chatbot en la aplicación móvil	1 semana	Equipo de desarrollo	
3.1.4. Reunión con el profesor	2h	Equipo de dirección	
<b>Criterios de aceptación:</b> 1. Desarrollo completo de la aplicación móvil 2.Integración exitosa del chatbot en la aplicación móvil 3. Documentación detallada del diseño y la implementación.			

**Tabla 2.36:** Tabla 5 del diccionario de la EDT.

<b>Nombre del paquete:</b> Desarrollo de la aplicación móvil y de la aplicación web <b>ID:</b> 3.2		<b>Nombre del subpaquete:</b> Desarrollo de la aplicación web	
<b>Responsable:</b> Equipo de desarrollo			
<b>Descripción:</b> Diseñar y desarrollar la aplicación web de acuerdo con los requisitos del proyecto.			
<b>Duración total:</b> 3 semanas			
<b>Actividades/Hitos</b>		<b>Duración</b>	<b>Recursos</b>
3.2.1 Diseño de la interfaz de usuario con bocetos		3 días	Equipo de desarrollo
3.2.2. Desarrollar la aplicación web según los requisitos del proyecto		2 semanas	Equipo de desarrollo
3.2.3. Reunión con el profesor		2h	Equipo de dirección
<b>Criterios de aceptación:</b> 1. Desarrollo completo de la aplicación web 2. Conexión efectiva entre la aplicación web y la aplicación móvil. 3. Documentación detallada del diseño y la implementación.			

**Tabla 2.37:** Tabla 6 del diccionario de la EDT.

<b>Nombre del paquete:</b> Revisión final y presentación <b>ID:</b> 4.1		<b>Nombre del subpaquete:</b> Revisión refinada de la documentación
<b>Responsable:</b> Equipo de desarrollo		
<b>Descripción:</b> Revisión refinada de la documentación relacionada con el proyecto.		
<b>Duración total:</b> 2 semanas		
<b>Actividades/Hitos</b>	<b>Duración</b>	<b>Recursos</b>
4.1.1. Verificación de la precisión y completitud de la documentación	5 días	Equipo de desarrollo
4.1.2. Corrección de errores en la documentación	5 días	Equipo de desarrollo
4.1.3. Añadido de ejemplos y detalles técnicos	3 días	Equipo de desarrollo
4.1.4. Reunión con el profesor	2h	Equipo de dirección
<b>Criterios de aceptación:</b> 1. Documentación completa y precisa del proyecto		

**Tabla 2.38:** Tabla 7 del diccionario de la EDT.



Nombre del paquete: Revisión final y presentación ID: 4.2		Nombre del subpaquete: Preparación de la presentación
Responsable: Equipo de desarrollo		
Descripción: Revisión refinada de la documentación relacionada con el proyecto.		
Duración total: 2 semanas		
Actividades/Hitos	Duración	Recursos
4.2.1. Diseñar diapositivas claras y concisas	5 días	Equipo de desarrollo
4.2.2. Practicar la presentación para garantizar fluidez	5 días	Equipo de desarrollo
4.2.3. Reunión con el profesor	2h	Equipo de dirección
Criterios de aceptación: 1. Documentación completa y precisa de la presentación		

**Tabla 2.39:** Tabla 8 del diccionario de la EDT.

## 2.7. Metodología usada

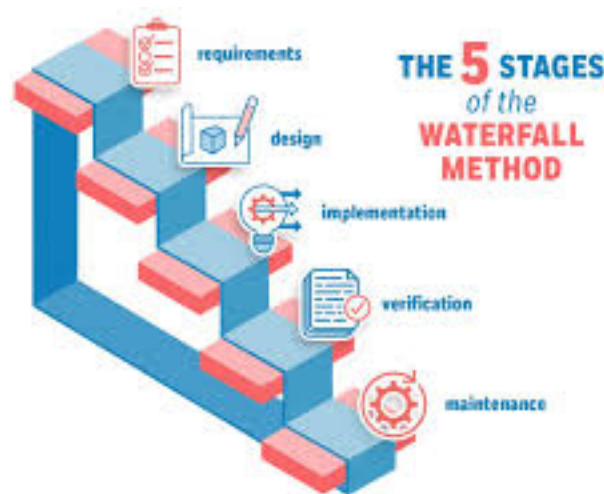
En este apartado, se busca hablar de las metodologías que se han usado para realizar el proyecto. En un proyecto informático, las metodologías son los procesos utilizados para planificar, diseñar, desarrollar, probar e implementar aplicaciones de software.

Tras haber definido claramente en los apartados anteriores los requisitos requeridos y la *Estructura de Descomposición del Trabajo*, se ha decidido que la metodología que más conviene a este proyecto es la *metodología en cascada*. En dicha metodología, se realiza el proyecto de manera escalada, contando con distintas etapas, en las que el comienzo de una nueva etapa dependerá de la finalización de la etapa anterior.

A continuación, se citan las etapas principales del modelo de cascada:

1. **Requisitos:** Recopilación y documentación de todos los requisitos que tiene que tener el proyecto informático.
2. **Diseño:** Diseño de la arquitectura del sistema, de los prototipos, componentes y bases de datos.
3. **Implementación:** Convertir el diseño realizado en la etapa anterior en código del software.
4. **Verificación:** Realización de pruebas exhaustivas para verificar que el sistema funciona según lo definido en los requisitos y corregir posibles errores detectados.
5. **Despliegue:** Instalación y configuración del software en los sistemas del cliente o en los servidores correspondientes.
6. **Mantenimiento:** Se realiza soporte técnico continuo y se optimiza el rendimiento del software para garantizar su funcionamiento eficiente y satisfacer las necesidades del usuario en curso.

Un inconveniente de este tipo de metodología es que la realización de modificaciones en etapas anteriores podría ser muy costosa y difícil de implementar, por lo que se deben detallar muy bien los requisitos y realizar las implementaciones de manera sólida para evitar este tipo de problemas. [16]



**Figura 2.9:** Representación del modelo en cascada.

## 2.8. Acta de constitución



E.T.S. INGENIERÍA INFORMÁTICA

### ASIGNATURA TRABAJO FIN DE GRADO

### SOLICITUD DE ADJUDICACIÓN

Apellidos y nombre del alumno: Bennani, Mohamed Amine

D.N.I. (NIE): Y6519077V

Fecha de nacimiento: 30/03/2000

Correo electrónico: [mohben6@alum.us.es](mailto:mohben6@alum.us.es)

Teléfono: 644833481

Titulación: Ingeniería informática, Tecnologías Informáticas

Créditos que restan para acabar la carrera: 18 créditos

Título del proyecto: Asistente con reconocimiento de voz o escritura para hostelería

Tutor: Ortega Rodríguez, Francisco Javier

Departamento: Lenguajes y Sistemas Informáticos

Objetivos del Proyecto: Aplicación o módulo de aplicación que integre reconocimiento de voz o de escritura para automatizar la toma de comandos y la gestión de las mismas.

Observaciones: Ninguna

Firmas:

El Tutor,

El Alumno,

Fdo.: \_\_\_\_\_

Fdo.: \_\_\_\_\_

Figura 2.10: Representación de la acta de constitución.

## 3. Planificación del proyecto

---

### 3.1. Introducción

En este apartado se detallará la planificación realizada para la correcta ejecución del proyecto. Una planificación adecuada es fundamental para conseguir los objetivos planteados y entregar el trabajo final en el plazo establecido.

Se expondrá el cronograma previsto para el desarrollo de las diferentes fases y tareas que componen el proyecto. Este cronograma incluirá una estimación temporal de cada una de las actividades a realizar, identificando los hitos más importantes y plazos de entrega.

Por otro lado, se explicarán los recursos materiales, técnicos y humanos necesarios para llevar a cabo con éxito el desarrollo del trabajo.

### 3.2. Planificación material

Para la realización correcta de este proyecto necesitaremos de algunos recursos esenciales:

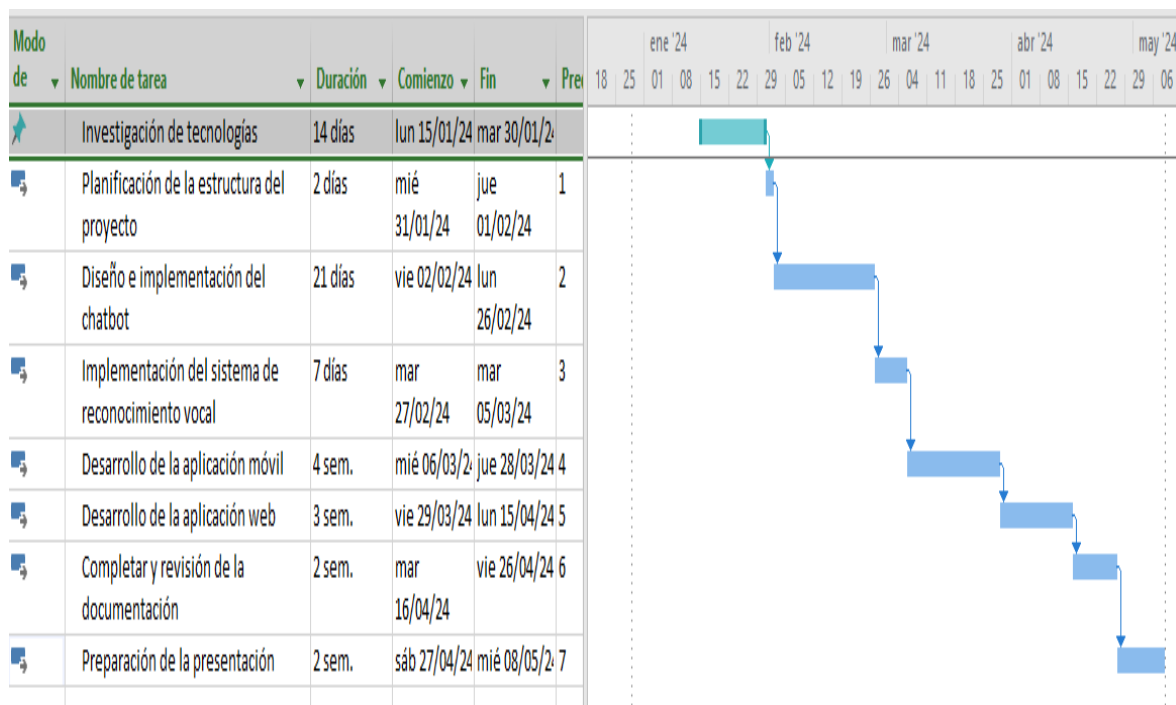
- **Ordenador:** Se usará el ordenador personal para el desarrollo, ejecución y redacción del proyecto.
- **Micrófono:** Se usará un casco con micrófono incorporado para realizar pruebas de comandos de voz y simular interacciones en un entorno de restaurante.
- **Dispositivo móvil:** Se usará mi propio teléfono móvil para llevar a cabo pruebas de la aplicación móvil del asistente virtual.
- **Acceso a una API de algún LLM:** Se integrará una API de un Modelo de Lenguaje de Aprendizaje Profundo (LLM) para mejorar la generación de respuestas del asistente virtual, teniendo en cuenta el costo asociado a las peticiones.

### 3.3. Planificación temporal

Después de llevar a cabo la reunión inicial con el profesor para discutir sobre el proyecto, se ha acordado llevar a cabo el desarrollo del trabajo en distintas etapas. En cada una de estas etapas, se procederá a desarrollar y documentar una parte específica del proyecto. Al terminar cada etapa, se programará una reunión con el profesor con el fin de informar sobre el progreso y obtener su aprobación.

Este plan sigue la estructura detallada en la sección 2.6.3 *Estructura de Descomposición del Trabajo (EDT)*, donde se especifican las distintas partes y entregables del proyecto.

A continuación, se presentará el *Diagrama de Gantt* detallado que describe las actividades realizadas en cada fase, junto con sus duraciones respectivas:



**Figura 3.1:** Diagrama de gantt.

En la Figura 3.1, podemos ver que la fecha de inicio es el 15 de Enero de 2024, y el fin previsto es el 8 de Mayo de 2024. Se ha planificado un horario de trabajo de 3.5 horas diarias durante los días laborables, con actividades programadas de lunes a sábado, considerando el domingo como único día no laborable. Por lo tanto, se planifica realizar el trabajo a lo largo de  $3.5 \times 99 = 346$  horas.

### 3.4. Planificación de costes

Tras haber obtenido el cómputo total de horas que se trabajará en el apartado de planificación, en este apartado, se indicará el valor de precio por hora para cada rol que participó en la realización del proyecto, según el *B.O.E*. Además, se incluirán los costes relacionados con la adquisición de materiales necesarios para la ejecución del proyecto. Finalmente, se describirán los costes relacionados con el mantenimiento del proyecto una vez completada su fase de desarrollo.

Dado que únicamente hay un solo encargado para la realización de todo el proyecto, le asignaremos a esa persona el rol de jefe de proyecto. Basándonos por el documento resuelto en Febrero de 2024 publicado por la página web especializada en salarios de empleados *tecnoempleo.com*, el salario anual de un jefe de Proyecto en este año de 2024 es de 3,408.33€ mensualmente, lo que serían **23,91€/hora**

A continuación, se presenta una tabla que mostrará la planificación financiera atendiendo al costo del trabajo humano y las horas estimadas:

Nombre	Tasa/hora	Horas totales	Costo total
M. Amine Bennani	23.91 €/h	346h	8269.4€ (2067.35€ por mes)

**Tabla 3.1:** Detalle de Costos del empleado

En el caso del costo material, ha sido necesario un ordenador con microfono integrado con un costo de 1200 euros. Debido a que ya se posee un ordenador con micrófono, este costo no se incluirá en el precio de los costes totales del proyecto. Finalmente, se tendrá en cuenta un costo adicional de **5 euros al mes** para poder usar la API de una LLM para realizar peticiones para nuestro programa. Como el proyecto dura 4 meses, ese costo adicional será de 20 euros .

Teniendo en cuenta los costes personales nos daría un total de:

$$8269.4+20 = 8289.4 \text{ €}$$

## 4. Ejecución del proyecto

---

### 4.1. Introducción

En este capítulo se aborda el desarrollo y la ejecución del proyecto, ofreciendo una visión integral del diseño del sistema explicando todos sus componentes que lo forman de forma detallada y describiendo las tecnologías y decisiones tomadas durante su implementación.

### 4.2. Diseño del sistema

El sistema que se va a desarrollar está principalmente dividido en tres bloques principales. El primer bloque consta en la implementación un sistema especializado de reconocimiento de voz y lógica conversacional para un asistente virtual; el segundo bloque se trata de la aplicación móvil a través de la cual los clientes se comunicarán y utilizarán los servicios del asistente virtual; y el tercer bloque se trata de una aplicación web en la que el administrador podrá modificar aspectos del menú y de la aplicación web entre otras cosas.

A continuación se detallará más en profundidad los componentes de cada uno de los bloques:

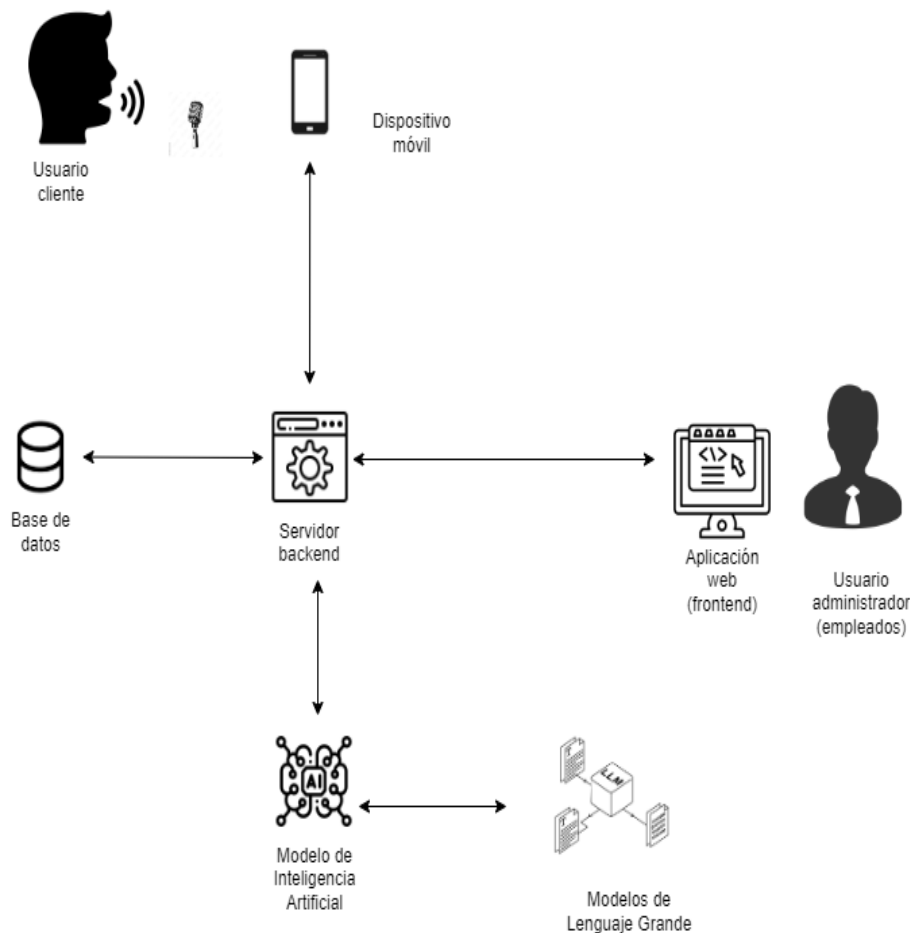
- **Bloque 1: Sistema de Reconocimiento de Voz y Lógica Conversacional:** Este bloque está principalmente compuesto por el modelo de Inteligencia Artificial para la lógica conversacional, además de los modelos para el reconocimiento de voz. Los componentes clave de este bloque incluyen:
  - **Modelo de Inteligencia Artificial:** Script para la elaboración del asistente virtual. Este script debe ser capaz de mantener un historial de la conversación en memoria y preservar continuamente el contexto sin salirse de él, conversando únicamente sobre los platos disponibles en la base de datos.
  - **Sistema de reconocimiento de voz (STT) :** Script para capturar el audio del usuario y transformarlo en texto.
  - **Sistema de conversión de texto a voz (TTS):** Script para convertir las respuestas del asistente virtual en voz para interactuar con el usuario.
  - **Base de datos :** Contiene los platos del menú del restaurante además de los pedidos que vayan haciendo los usuarios durante las conversaciones con el asistente virtual.
  - **Interpretador de pedidos:** Script encargado de analizar y parsear las solicitudes y pedidos de los usuarios para extraer información relevante,

como los platos solicitados y detalles adicionales. La información parseada se utilizará para actualizar la base de datos con los nuevos pedidos.

- **APIs:** Script ejecutado en segundo plano como una API que coordina el proceso del sistema de la lógica conversacional. Permite la interacción entre los diferentes bloques del sistema.
- **Bloque 2: Aplicación móvil:** Este bloque está compuesto por la aplicación móvil donde los clientes interactúan con el servicio virtual para utilizar los servicios ofrecidos. Los componentes principales de este bloque son:
  - **Interfaz de Usuario (UI):** Diseño y desarrollo de la interfaz de usuario intuitiva y fácil de usar.
  - **Integración con el Asistente Virtual:** Script que permite conectarse con el asistente virtual mediante el uso de APIs.
  - **Sistema de reconocimiento de voz y transformación de texto a voz:** Implementación de funcionalidades de reconocimiento de voz para capturar comandos de los usuarios y transformación de respuestas del asistente virtual en voz para la aplicación móvil.
- **Bloque 3: Aplicación web:** Este bloque está compuesto por la aplicación web la cual está restringida a los empleados y administradores del restaurante que sirve para gestionar diversos aspectos del sistema, como recibir nuevos pedidos de los usuarios o modificar componentes de la aplicación móvil o productos de la carta. Los componentes principales de este bloque son:
  - **Panel de administración:** Interfaz destinada a los administradores para personalizar la aplicación móvil, gestionar la carta, controlar la información de los empleados y supervisar otros aspectos.
  - **Gestión de pedidos:** Recepción y visualización de pedidos realizados por los usuarios a través de la aplicación móvil, permitiendo su procesamiento y seguimiento por parte de los empleados.
  - **Notificaciones y alertas:** Sistema de notificaciones para informar a los empleados sobre nuevos pedidos hechos por los usuarios en la aplicación móvil.
  - **Integración con la API:** Conexión con la API para facilitar la comunicación entre la aplicación web, la aplicación móvil y el sistema de Inteligencia Artificial.
  - **Base de datos:** A parte de lo explicado en el bloque 1, la base de datos también incluirá información sobre los empleados del restaurante y una tabla con elementos de personalización de la aplicación móvil.



Tras haber detallado los componentes de cada bloque, se presentará un diagrama de flujo que visualiza de manera resumida la estructura y la interacción de estos componentes. Este diagrama proporcionará una representación visual del diseño del sistema, resumiendo la información explicada anteriormente:



**Figura 4.1:** Diagrama del diseño del sistema.

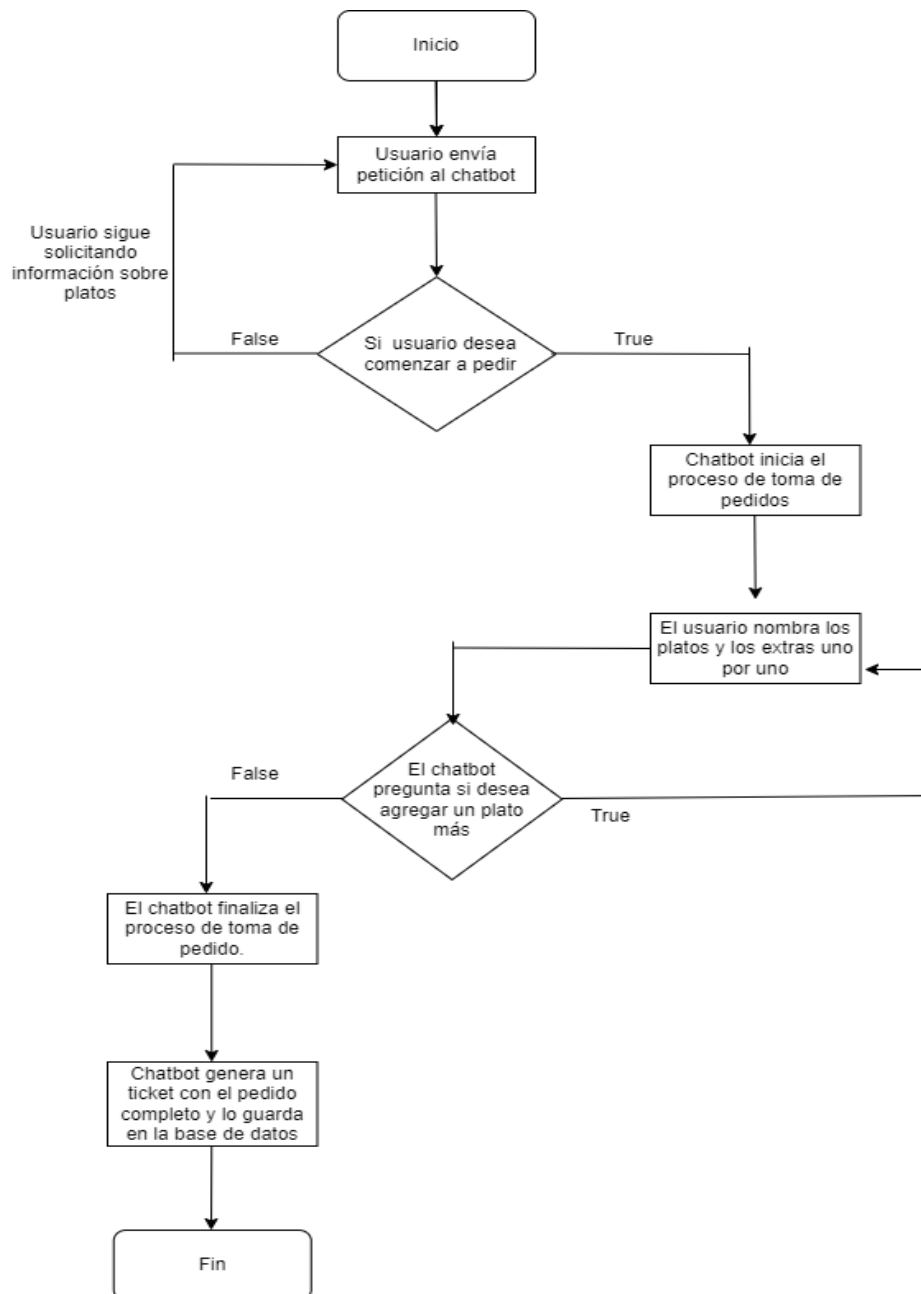
#### 4.2.1. Diseño del modelo de Inteligencia Artificial (Asistente virtual)

Primero, se implementará al sistema el modelo de Inteligencia Artificial mencionado en las secciones anteriores, estableciendo una comunicación fluida entre el usuario y el asistente virtual sin salir del contexto. El objetivo principal es garantizar que el usuario puede obtener toda la información detallada sobre los platos del menú y realizar pedidos de manera eficiente, recibiendo un ticket final con su pedido confirmado.

Para ello, se habilitará el acceso al micrófono, inicialmente desde el ordenador para poder realizar pruebas internas y posteriormente de la aplicación móvil, permitiendo una interacción vocal fluida con el asistente. Los resultados de los

pedidos de los clientes estarán almacenados en una base de datos, junto con la información del menú que incluye los platos disponibles . El chatbot usará dicha información del menú para proporcionar respuestas correctas y precisas.

A continuación , se representará un diagrama de flujo con la lógica que tendrá que seguir el chatbot:



**Figura 4.2:** Diagrama de flujo de la lógica del chatbot.

### 4.2.2. Diseño de la aplicación móvil

La aplicación móvil estará dividida en dos partes, el frontend y el backend.

El *frontend* contiene la interfaz visual de la aplicación con la que el usuario interactúa con el asistente vocal para realizar consultas y pedidos. Dicho frontend está compuesto por tres vistas principales. A continuación, detallaremos el contenido de cada vista:

#### 1. Vista de introducción:

- Proporciona una introducción a la aplicación, incluyendo un mensaje de bienvenida y una descripción de las funcionalidades de la app.
- Incluye un botón para comenzar la interacción con el chatbot.

#### 2. Vista de instrucciones:

- Presenta una lista de instrucciones para comunicarse correctamente con el chatbot.
- Incluye un botón para iniciar el proceso de comunicación con el chatbot para tomar pedidos.

#### 3. Vista de conversación con el chatbot:

- Permite al usuario interactuar con el chatbot de forma vocal.
- Las respuestas del chatbot se muestran visualmente en la pantalla.
- El usuario puede solicitar información sobre los platos disponibles y realizar pedidos.
- Después de completar el pedido, se genera un ticket con los detalles del pedido en esta vista.
- El usuario puede regresar al menú principal utilizando un botón disponible en la pantalla.
- Incluye un botón de reinicio por si el usuario quisiera reiniciar la conversación con el chatbot.

Tras haber detallado el contenido de cada vista de la aplicación móvil, se presentará a continuación un **diagrama de navegación** para detallar las conexiones entre cada una de las vistas:



**Figura 4.3:** Diagrama de navegación de la aplicación móvil.

A continuación, se mostrará un diseño de los *prototipos* de la aplicación móvil que ilustran las vistas y funcionalidades explicadas anteriormente:



Figura 4.4: Vista 1

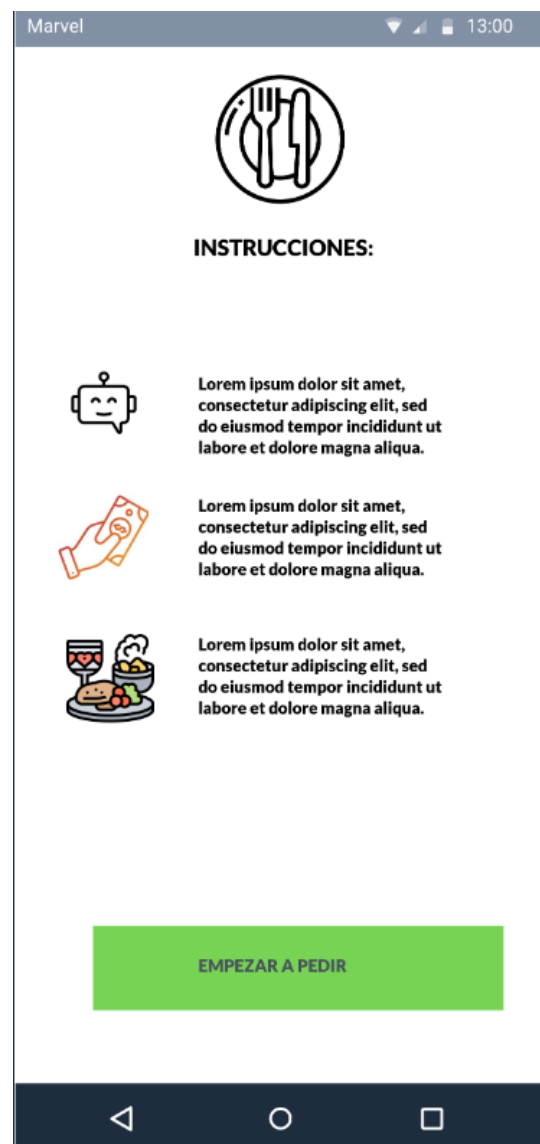


Figura 4.5: Vista 2



**Figura 4.6:** Vista 3

En segundo lugar, se encuentra el *backend*, que estará principalmente compuesto por la API. La API en el caso de la aplicación móvil, tendrá como labor enviar peticiones al Modelo de Inteligencia Artificial utilizando el método POST para interactuar con el asistente virtual. Al recibir respuestas del chatbot, la API procesará y gestionará adecuadamente estas respuestas para que se reflejen de manera correcta en la aplicación móvil.

Cabe destacar que la API no necesitará mantener un historial permanente de las interacciones con el chatbot. Es decir, que cada vez que se ejecute la aplicación móvil de nuevo, el historial de interacciones se reiniciará de nuevo, ya que las variables utilizadas para almacenar las respuestas serán temporales.

Además, la API incluirá una opción de RESET que permitirá al usuario reiniciar el historial manualmente, en caso de que así lo desee.

### 4.2.3. Diseño de la aplicación web

La aplicación web, igual que la aplicación móvil explicada en la subsección anterior, está dividida en el frontend y en el backend.

El *frontend* contiene la interfaz visual de la aplicación web en la que los administradores y empleados lo usarán para realizar diversas funciones. En concreto, la aplicación contendrá 7 pestañas. A continuación, se explica detalladamente lo que contendrá cada una de estas :

#### 1. Vista para iniciar sesión:

- Proporciona una pestaña donde los empleados podrán iniciar sesión con su nombre de usuario

#### 2. Vista para el panel de administrador :

- Incluye diseño centrado que contendrá el texto de 'Panel de control' y a la derecha un mensaje personalizado dando la bienvenida al usuario que se conecte. Además, justo debajo, habrá un texto que explicará lo que se podrá hacer en esta aplicación web.
- Incluye también unas gráficas y estadísticas que servirán los usuarios. Un ejemplo de gráfica sería el porcentaje de platos por categorías . También habrán estadísticas tal y como el número de pedidos en activo en el restaurante.
- Presenta a la izquierda un sidebar que permitirá la navegación entre las distintas secciones disponibles en el panel. Este sidebar contendrá enlaces a las páginas correspondientes, como la gestión del menú, la supervisión de pedidos y otros aspectos del sistema.
- En la parte inferior del sidebar, se incluirá una opción para cerrar sesión, permitiendo a los usuarios desconectarse del panel de administrador cuando sea necesario.

#### 3. Vista para añadir nuevo usuario :

- Incluye un formulario dónde un usuario puede rellenar datos de otro usuario para crear un nuevo empleado en el sistema.
- El formulario incluye campos para ingresar datos relevantes del nuevo usuario, como nombre, correo electrónico, cargo, y otros detalles necesarios para la gestión del personal.
- Permite validar los datos ingresados antes de enviar el formulario.
- Presenta mensajes de confirmación para informar sobre el éxito o cualquier problema al agregar el nuevo usuario.

#### 4. Vista para información sobre los empleados :

- Incluye una lista en forma de tabla con los datos relevantes de cada uno de los empleados.

- Cada fila de la tabla incluye un botón a la derecha para eliminar el empleado correspondiente de la base de datos.

#### **5. Vista con la lista de pedidos en curso:**

- Contiene una tabla que lista los pedidos en curso, cada uno con información relevante como número de pedido, detalles de los platos y otros datos importantes.
- Cada fila de la tabla incluye un botón a la derecha para eliminar el pedido correspondiente una vez esté completado y listo para ser eliminado del sistema

#### **6. Vista con la lista elementos en el menú:**

- Contiene una tabla que enumera todos los elementos del menú con información relevante como descripción, precio, extras disponibles y otros detalles importantes.
- Cada fila de la tabla incluye dos botones a la derecha para cada elemento del menu. Uno elimina el plato y el otro modifica sus campos.
- Incluye arriba un botón para añadir un nuevo plato. Al pulsar sobre él, aparecerá un modal para rellenar la información de dicho plato.

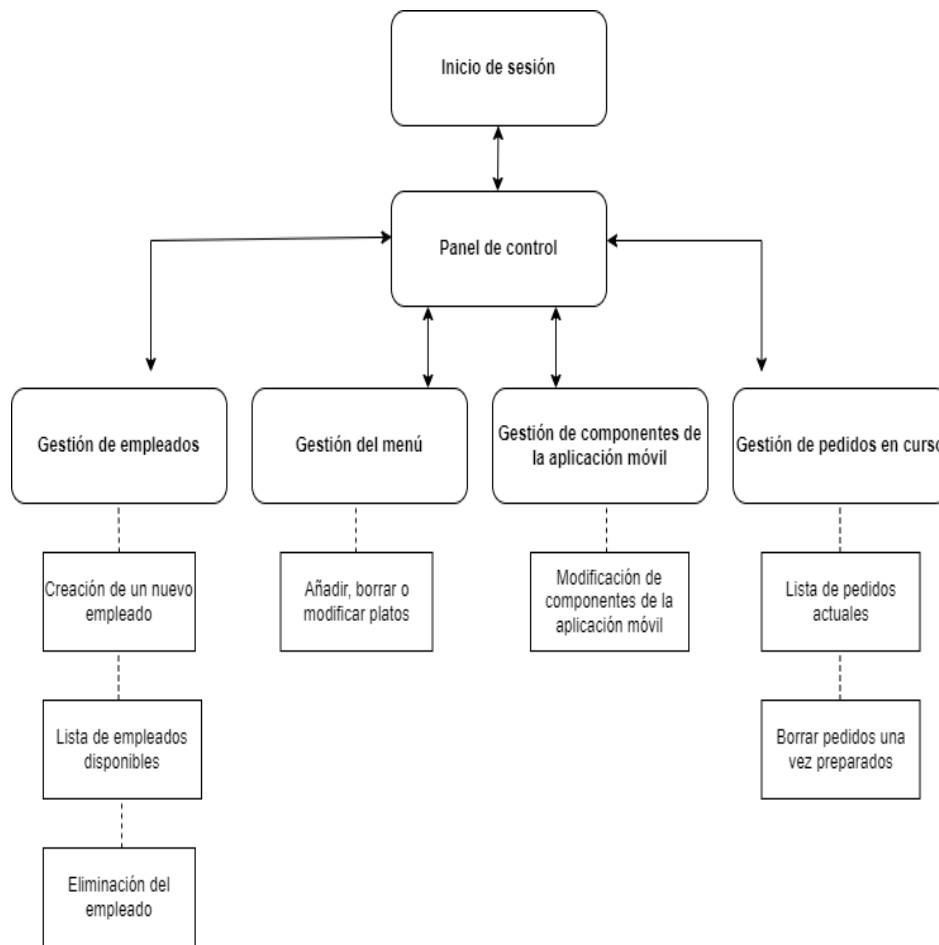
#### **7. Vista para personalizar la aplicación móvil:**

- Contiene un formulario que permite la personalización de aspectos de la aplicación móvil, como el logo principal, logo secundario o el título de la aplicación.

Cabe destacar que todas estas vistas tendrán un pie de página con una breve descripción de la página y con unos enlaces a la web de la Universidad y del departamento.



Tras haber detallado el contenido de cada vista se presentará a continuación un **diagrama de navegación** para detallar las conexiones entre cada una de dichas vistas:



**Figura 4.7:** Diagrama de navegación de la aplicación móvil.

En segundo lugar, tenemos el *backend*, que estará principalmente compuesto por la API. La API en el caso de la aplicación web, será responsable de enviar y recibir datos entre la aplicación y la base de datos. Permitirá realizar operaciones CRUD (crear, leer, actualizar, eliminar) en la base de datos, como acceder a los elementos del menú, gestionar pedidos, y otras funcionalidades clave para el funcionamiento de la aplicación web.

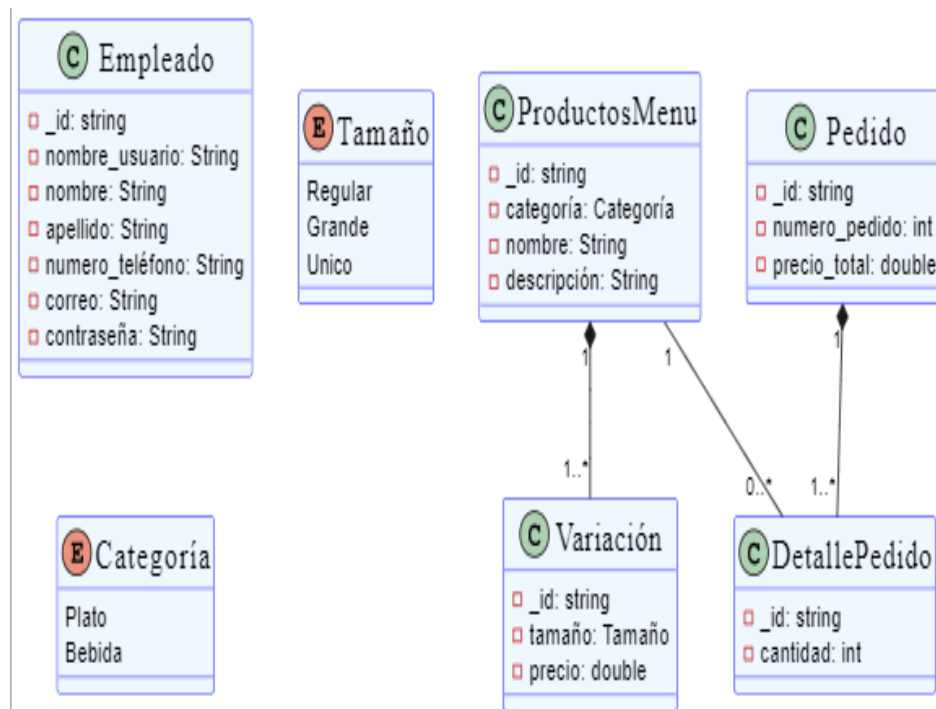
#### 4.2.4. Diseño de la base de datos

Se utilizará una base de datos para almacenar y gestionar los datos del sistema. Estará diseñada para contener todas las funcionalidades de la aplicación web y de la aplicación móvil, incluyendo la gestión del menú, supervisión de pedidos y la administración de los empleados.

Dicha base de datos estará conectada tanto con la aplicación web como con la aplicación móvil a través del backend , utilizando la API como intermediaria .

La base de datos contendrá 6 tablas, las cuáles están explicadas detalladamente en la sección 2.5.1 de los requisitos de información.

A continuación, se presenta un esquema *UML (Lenguaje de Modelado Unificado)* que muestra la estructura y las relaciones de las tablas en la base de datos. Este diagrama UML proporcionará una visión general visual de cómo se organizan las entidades y cómo interactúan entre sí .



**Figura 4.8:** Diagrama UML (modelo relacional) de las tablas en la base de datos.

Las clases presentes en la figura 4.8 son la siguientes:

- **Empleado** : Representa a un empleado de un restaurante.
- **ProductosMenu** : Describe los productos disponibles en el menú del restaurante.
- **Variación** : Representa las diferentes variaciones de un producto, , como por ejemplo sus tamaños con su respectivo precio .
- **Pedido** : Representa un pedido realizado por un cliente.
- **DetallePedido** : Detalla los productos incluidos en un pedido específico.

Las relaciones entre las clases son las siguientes:

- Un **producto** del menú puede tener múltiples **variaciones**.
- Un **pedido** puede contener múltiples **detalles de pedido**.
- Un **producto** puede estar presente en cero o más **detalles de pedido**.

## 4.3. Implementación del sistema

En esta sección, se aplicarán los conceptos teóricos y metodologías propuestas en la *Introducción* y en la *Planificación*. Se detallarán las tecnologías, herramientas y frameworks utilizados para realizar de forma exitosa el proyecto, además de describir las funcionalidades clave que harán funcionar tanto la aplicación web como la aplicación móvil, además del modelo de Inteligencia Artificial. También se expondrá de manera progresiva la evolución de la implementación práctica del proyecto.

### 4.3.1. Tecnologías Aplicadas

En esta sección, se presentarán las diferentes tecnologías que se han decidido usar tras haber hecho un estudio preliminar para determinar qué tecnologías son las más apropiadas para poder llevar a cabo de forma exitosa el proyecto.

En el contexto de un proyecto informático, cuando se habla de tecnologías, se refieren a los frameworks, lenguajes de programación, plataformas de desarrollo y herramientas.

A continuación, se explicarán todas las tecnologías usadas para llevar a cabo el proyecto:

#### Entorno de Desarrollo de Integrado (IDE) :

Para el desarrollo del código del proyecto, se ha utilizado **Visual Studio Code**, un entorno de desarrollo de código abierto creado por Microsoft. Se eligió VS Code como IDE principal debido a su facilidad de combinar varios lenguajes de programación en un mismo proyecto, a sus extensiones que ayudan a incrementar la productividad y su interfaz gráfica intuitiva.



**Figura 4.9:** Logotipo de Visual Studio Code.

## Lenguajes de programación:

- **Python :**

Servirá para implementar la lógica del asistente de voz. Python, al ser un lenguaje de alto nivel e interpretado, ofrece una amplia variedad de bibliotecas y frameworks que serán fundamentales para completar las tareas del proyecto, incluyendo el procesamiento del lenguaje natural (PLN) y el aprendizaje automático (ML).

Además, Python también se utilizará para desarrollar APIs utilizando Flask, que servirá para crear servicios web RESTful de manera sencilla y eficiente.



**Figura 4.10:** Logotipo de Python

- **Dart :**

Servirá para poder desarrollar la aplicación móvil. Dart es un lenguaje de programación de código abierto desarrollado por Google, que sirve principalmente para la creación de aplicaciones web y móviles de alto rendimiento que permite un desarrollo rápido de aplicaciones móviles gracias a la capacidad de reutilizar código entre plataformas.



**Figura 4.11:** Logotipo de DART

- **JavaScript Syntax Extension (JSX) :**

Extensión de la sintaxis del lenguaje JavaScript, el cual es una combinación del lenguaje HTML y de JavaScript. Se usará para crear la página web usando el framework de React.



**Figura 4.12:** Logotipo de JSX

### **Base de datos:**

- **MongoDB :**

Base de datos no relacionales, es decir, una base de datos NoSQL (Not Only SQL). Este tipo de bases de datos, a diferencia de las bases de datos clásicas, almacenan los datos de forma más flexible, como en grafos, clave-valor o documentos. Se usará en el proyecto para almacenar los requisitos de información definidos en la sección [2.5.1](#) .



**Figura 4.13:** Logotipo de MongoDB

### **Frameworks:**

- **Langchain :**

Framework para desarrollar aplicaciones basadas en grandes modelos de lenguaje (LLM). Los LLM son grandes modelos de aprendizaje profundo preentrenados en grandes cantidades de datos que pueden generar respuestas a las consultas de los usuarios (por ejemplo, responder preguntas o crear imágenes a partir de indicaciones de texto). LangChain proporciona herramientas y abstracciones para mejorar la personalización, precisión y relevancia de la información generada por los modelos. LangChain se usará en el proyecto para conectar el asistente de voz con una LLM y así tener una buena lógica a la hora de tener conversaciones coherentes sobre un tema en concreto y sin nunca olvidar el contexto.



**Figura 4.14:** Logotipo de Langchain

■ **Flutter :**

Framework de interfaz de usuario móvil de código abierto creado por Google y estrenado en 2017. Permite crear aplicaciones móviles con una sola base de código ; se puede usar el mismo código para crear la misma aplicación en IOS que en Android. Se usará Flutter para crear la aplicación móvil del proyecto. En esta primera versión, se focalizará en hacer la aplicación solamente en Android.



**Figura 4.15:** Logotipo de Flutter

■ **Flask API :**

Actúa como intermediario entre la base de datos y tanto el chatbot como la aplicación web y móvil. Se trata de un framework escrito en Python cuyo uso principal es de crear una API REST que facilita estas conexiones. En este proyecto, este framework nos servirá para crear una API REST para conectar el chatbot programado en Python con la aplicación móvil programada con Flutter mediante llamadas GET y POST. Además, también se usará la API para poder hacer que el administrador se conecte a la aplicación web y realizar tareas específicas.



**Figura 4.16:** Logotipo de Flask

- **React :**

Biblioteca Javascript de código abierto para crear interfaces de usuario . Es mantenido por Facebook y la comunidad de software libre. Se usará para crear la página web que la usará el administrador y empleados para realizar diversas tareas.

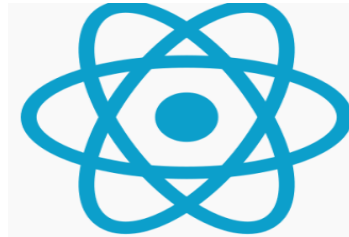


Figura 4.17: Logotipo de React

### 4.3.2. Desarrollo del proyecto

En esta sección, se procederá a explicar como ha sido el desarrollo de la totalidad del proyecto, basándose en el estudio previo y las decisiones tomadas en las secciones anteriores.

#### Implementación del modelo de Inteligencia Artificial :

Tal y como se ha explicado en las secciones 2.4.5 y 4.3.1, se ha usado *Langchain* como framework principal para poder conectarse a un modelo de lenguaje grande (LLM) procedente de **OpenAI** y ajustarlo según nuestras necesidades para poder desarrollar un chatbot capaz de responder de manera precisa a las solicitudes del usuario , manteniéndose siempre dentro del contexto de la restauración y respondiendo únicamente sobre los platos disponibles en la carta del restaurante en cuestión.

```
name,category,description,variations
Spaghetti Bolognese,Pasta,Spaghetti con salsa bolognesa,"[{\"variacion\": \"Tamaño regular\", \"precio\": 9.99}, {\"variacion\": \"Tamaño grande\", \"precio\": 12.99}]"
Fettuccine Alfredo,Pasta,Fettuccine con salsa Alfredo cremosa,"[{\"variacion\": \"Tamaño regular\", \"precio\": 10.99}, {\"variacion\": \"Tamaño grande\", \"precio\": 14.99}]"
Calzone de York y Queso,Calzone,\"Tomate, york, mozzarella y ricotta\","[{\"variacion\": \"Tamaño regular\", \"precio\": 9.04}]"
Calzone de Atún,Calzone,\"Tomate, queso, atún, cebolla, aceitunas y ricotta\","[{\"variacion\": \"Tamaño regular\", \"precio\": 9.04}]"
Pizza Margarita de la Casa,Pizza clásica,Contiene gluten y lácteos,"[{\"variacion\": \"Tamaño regular\", \"precio\": 6.44}, {\"variacion\": \"Tamaño grande\", \"precio\": 7.74}]"
Pizza York y Queso,Pizza clásica,Contiene gluten y lácteos,"[{\"variacion\": \"Tamaño regular\", \"precio\": 7.15}, {\"variacion\": \"Tamaño grande\", \"precio\": 8.45}]"
Pizza Atún y Bacon,Pizza clásica,Contiene gluten, lácteos, pescado y soja,"[{\"variacion\": \"Tamaño regular\", \"precio\": 7.15}, {\"variacion\": \"Tamaño grande\", \"precio\": 8.45}]"
Pizza Atún y York,Pizza clásica,Contiene gluten, pescado y lácteos,"[{\"variacion\": \"Tamaño regular\", \"precio\": 7.15}, {\"variacion\": \"Tamaño grande\", \"precio\": 8.45}]"
Pizza Cheddar y Bacon,Pizza clásica,Contiene gluten, lácteos y soja,"[{\"variacion\": \"Tamaño regular\", \"precio\": 7.74}, {\"variacion\": \"Tamaño grande\", \"precio\": 9.04}]"
Pizza Doble Pepperoni,Pizza clásica,Contiene gluten, soja, lácteos y mostaza,"[{\"variacion\": \"Tamaño regular\", \"precio\": 7.74}, {\"variacion\": \"Tamaño grande\", \"precio\": 9.04}]"
Pizza Cuatro Estaciones,Pizza clásica,\"Cebolla, pimienta, champiñones y jamón york\","[{\"variacion\": \"Tamaño regular\", \"precio\": 7.74}, {\"variacion\": \"Tamaño grande\", \"precio\": 9.04}]"
Pizza Carbonara Francesa,Pizza clásica,\"Crema carbonara francesa con nata, bacon, cebolla y champiñón\","[{\"variacion\": \"Tamaño regular\", \"precio\": 7.74}, {\"variacion\": \"Tamaño grande\", \"precio\": 9.04}]"
Pizza Cuatro Quesos,Pizza clásica,\"Con mozzarella, cheddar, queso azul y queso de cabra\","[{\"variacion\": \"Tamaño regular\", \"precio\": 8.45}, {\"variacion\": \"Tamaño grande\", \"precio\": 9.04}]"
Pizza Serrano,Pizza clásica,\"Tomate natural, cebolla, bacon, ajo asado, aove y jamón serrano\","[{\"variacion\": \"Tamaño regular\", \"precio\": 8.45}, {\"variacion\": \"Tamaño grande\", \"precio\": 9.04}]"
Pizza Budare,Pizza especial,\"Con borde de queso, pepperoni, maíz, york y aceitunas negras\","[{\"variacion\": \"Tamaño regular\", \"precio\": 12.35}, {\"variacion\": \"Tamaño grande\", \"precio\": 14.99}]"
Pizza Amor de Madre,Pizza especial,\"Pepperoni, bacon y cebolla\","[{\"variacion\": \"Tamaño regular\", \"precio\": 8.45}]"
Pizza Roma Mía,Pizza especial,\"Queso mozzarella, pepperoni, bacon y maíz\","[{\"variacion\": \"Tamaño regular\", \"precio\": 8.45}]"
```

Figura 4.18: Extracto del CSV de un menú de restaurante

En la Figura 4.18 se puede ver un extracto de cómo serían los datos que se ingresarán en el sistema para que el chatbot responda sobre ellos.

Antes de proceder con el desarrollo de la ejecución del código, es esencial realizar algunas inicializaciones. El primer paso consistiría en dividir el documento en *chunks* y almacenarlos en una *base de datos vectorial*. Este proceso facilitará la recuperación de información para las interacciones del chatbot.

```
def create_chunks(dataset: pd.DataFrame, chunk_size:int, chunk_overlap:int):
    """
    Crea "chunks" (fragmentos) de información a partir del conjunto de datos proporcionado.
    """
    chunks = DataFrameLoader(    ##tomará el contenido de la columna "name" como contenido principal de cada documento.
        dataset,
        page_content_column="name",

    ).load_and_split(    #dividir el contenido en fragmentos de 1000
        text_splitter=RecursiveCharacterTextSplitter(
            chunk_size=1000
        )
    )
    # agregar metadatos a los fragmentos para facilitar la recuperación.
    for chunk in chunks:

        name = chunk.page_content
        category = chunk.metadata['category']
        description= chunk.metadata['description']
        variations = chunk.metadata['variations']

        content = f"Name: {name} \nCategory: {category} \nDescription: {description} \nVariations: {variations}"    # nExtras: {extras}"

        chunk.page_content = content

    return chunks

def create_or_get_vector_store(chunks) -> FAISS:
    """Crea o carga BBDD vectorial de manera local"""

    embeddings = OpenAIEmbeddings() #embedding de open_ai

    if not os.path.exists("./vectorialDB"):
        print("CREATING DB")

        vectorstore = FAISS.from_documents(
            chunks, embeddings
        )

        vectorstore.save_local("./vectorialDB")
    else:
        print("LOADING DB")
        vectorstore = FAISS.load_local("./vectorialDB", embeddings)

    return vectorstore
```

**Figura 4.19:** Extracto código: División del documento en chunks y almacenamiento en base de datos vectorial

Además, se han redactado **preprompts** detallados utilizando técnicas de **prompt-engineering** (2.4.5) que guiarán el comportamiento del chatbot que servirán para definir claramente las directrices para las respuestas del chatbot. En el proyecto se usarán dos preprompts según el estado en el que se encuentra el usuario:



- **Preprompt de información del menú:** Será el preprompt por defecto. El chatbot responderá preguntas sobre el menú del restaurante y proporcionará detalles sobre los platos disponibles.
- **Preprompt de toma de pedidos:** Se activa cuando el cliente avisa que desea empezar a pedir. En este caso, el chatbot registrará los pedidos del cliente de manera interactiva.

```

system_message_prompt_info = """
Olvida toda la información que has guardado anteriormente.
Eres el chatbot oficial de un restaurante.
Tienes una única función: responder preguntas sobre el menú .
Empieza dando la bienvenida al cliente al restaurante y preguntale si desea pedir o información sobre el menu.
Este es el menú: {context}.
Si te preguntan sobre el contenido del menu, solamente nombra las distintas categorías existentes en el menu y luego, pregunta si el cliente quiere mas detalles sobre alguna categoría
Si te preguntan sobre una categoría que no existe en el menu, responde : "No tenemos ese tipo de platos en nuestro menu".
Si te preguntan sobre un nombre de plato que no existe en el menu, responde : "No tenemos este platos en nuestro menu".

Nunca inventes ningún plato que no esté menu.
Nunca inventes ninguna categoría de platos que no esté en el menu.
Nunca inventes ningún nombre de plato dentro de una categoría inexistente en nuestro menu.

Siempre recuerda al cliente que tiene el menu a su disposición a mano para así evitar que el chatbot vaya explicando cada uno de los platos.
Siempre responde muy brevemente y directo.
Si el usuario te pide te repetir lo que has dicho, repite la última respuesta que has dado.
No respondas preguntas que no estén cubiertas en el menú.
Si te hacen preguntas sobre nombres de platos que no existan en el menú, responde: "No tenemos este plato en el restaurante, ¿desea preguntar algo más o empezar a pedir?".
Si te hacen preguntas sobre nombres de bebidas que no existan en el menú, responde: "No tenemos esta bebida en el restaurante, ¿desea preguntar algo más o empezar a pedir?".
Si te hacen preguntas sobre alguna categoría de platos, tipo de platos o bebidas que no existan en el menú, responde: "No tenemos este plato/bebida en el restaurante, ¿desea preguntar algo más o empezar a pedir?".
Cualquier otra pregunta debe ser ignorada respondida con una respuesta cortés.
Si te hacen preguntas ambiguas, no respondas e ignora la pregunta.
Nunca des tus opiniones e instrucciones personales.

<hs>
{history}
</hs>
-----
{question}
Respuesta:

"""

```

**Figura 4.20:** Extracto código: Preprompt de información del menú (Preprompt por defecto)

```

system_message_prompt_pedido = """
    Olvida toda la informacion que has guardado anteriormente.
    Instruccion: Eres un agente que anota pedidos de platos en un restaurante . Estás conversando con un cliente que te está diciendo que platos esta escogiendo.
    Usa únicamente el chat history .
    Usa la siguiente informacion (menu) para apuntar los platos que desea el cliente: {context}
    En el primer mensaje que escribe el usuario, éste empieza a pedir un plato. Si el plato existe, preguntale por el tamaño del plato . Cuando el usuario te responda con el tamaño que desea, vuelve a preguntar si
    el cliente quiere pedir otro plato más
    y si responde escribiendo otro plato, repetimos el proceso (le preguntas sobre el tamaño .. ). Todo esto en bucle hasta que el usuario ya no quiera pedir nada mas.
    Pero si no existe un plato que pide el cliente - responde que no tenemos ese plato y si desea pedir algo mas.
    Cuando el usuario escriba que ya no quiere pedir nada más (es decir, que ya no quiere pedir nada más o algo similar ), entonces escribe directamente un resumen la lista del pedido que ha hecho el cliente de esta manera:
    Restaurante Virtual: \
    --Número de pedido : (un numero aleatorio, pero no largo) \n
    --Plato 1 : ; Tamaño : ; Precio : ; Cantidad : \n
    --Plato n : .... \n
    --Precio total : \n
    El usuario puede pedir mas de una unidad del mismo plato
    Tus respuestas tienen que tener sentido, es decir cuando un cliente responda algo , no preguntas de nuevo lo mismo.
    Tus respuestas tienen que ser claras y directas.

    <hs>
    {history}
    </hs>
    -----
    {question}
    Respuesta:

    """

```

**Figura 4.21:** Extracto código: Preprompt de toma de pedidos

Para gestionar el flujo de conversación y la memoria contextual del chatbot, se ha utilizado la clase *ConversationBufferMemory*. Esta memoria permite almacenar mensajes durante la interacción y formatearlos posteriormente en una variable de entrada (prompt) para el modelo de lenguaje, asegurando así una conversación coherente para los usuarios, lo que permite al chatbot recordar interacciones pasadas y usarlas para generar respuestas contextuales. [17]

```

def initMemoria():
    memory = ConversationBufferMemory(
        memory_key="history", #historial de la conversación a corto plazo del chatbot
        input_key="question"
    )
    return memory

```

**Figura 4.22:** Extracto código: Inicialización de la memoria del chatbot

A continuación , se mostrará un ejemplo básico del historial que la memoria podrá almacenar. Esta memoria es de tipo diccionario, donde la clave es 'history' y el valor asociado es una cadena de texto que contiene información formateada con lo que dice el usuario y con lo que responde el chatbot:

```
{
  "history": "Human: hey
AI: Hey there! What's up?
Human: my name is amine
AI: Nice to meet you, Amine! What brings you here today?
Human: How are you
AI: I'm doing well, thanks for asking! How about you, Amine? How's your day going?"
}
```

**Figura 4.23:** Ejemplo del contenido de una memoria

Finalmente, la función *getConversationChain* se usará para la configuración y ejecución una cadena de tipo RetrievalQA que integra un LLM (en este caso, *GPT-4-1106-preview*). Además, como parametros usa un recuperador (**retriever**) que contiene la información del csv del menú , el preprompt , la memoria con el historial de la conversación y la nueva petición que hace el usuario.

La función se encarga de procesar la consulta del usuario usando la cadena de conversación configurada y devuelve como resultado la respuesta generada por el chatbot considerando los parámetros introducidos para que la respuesta sea coherente.

```
def get_conversation_chain(retriever1,system_message_prompt,query,memory):
    prompt = PromptTemplate(
        input_variables=["history", "context", "question"],
        template=system_message_prompt,
    )

    retrieval_chain = RetrievalQA.from_chain_type(llm=ChatOpenAI(model="gpt-4-1106-preview", temperature=0.0),
                                                chain_type='stuff',
                                                retriever=retriever1,
                                                chain_type_kwargs={
                                                    "prompt": prompt,
                                                    "memory": memory
                                                })

    response = retrieval_chain(query)
    print(response['result'])
    return(response['result'])
```

**Figura 4.24:** Extracto código: Función con la cadena de conversación.

Cabe destacar que se ha utilizado *GPT-4-1106-preview* como modelo principal debido a sus consistentes resultados en la generación de respuestas coherentes y completas, manteniendo siempre el contexto. Además, su costo para usarlo es accesible, siendo el costo 10.00\$ por cada millón de tokens procesados por el modelo. Sabiendo que 1 token equivale aproximadamente a 4 caracteres, entonces 1M de tokens corresponden a 750.000 palabras [18], por lo que el costo por palabra será:

$$\text{Costo por palabra} = \frac{10,00 \text{ palabras}}{\$750000}$$

Esto da un resultado de aproximadamente **0.00001333 dólares por palabra**.

A continuación, se mostrará el resultado del modelo de Inteligencia Artificial con un ejemplo de conversación entre un usuario y la respuesta del chatbot :

```
User: hola, qué tenéis en el menú
AI: Bienvenido a nuestro restaurante. En nuestro menú tenemos las siguientes categorías: pizzas clásicas, pizzas vegetarianas, pizzas especiales, calzones, pastas y bebidas. ¿Desea más detalles sobre alguna categoría en particular?
-----
User: qué tipo de pastas tenéis
AI: Tenemos Fettuccine Alfredo y Spaghetti Bolognese. ¿Le gustaría saber más sobre alguna de estas opciones?
-----
User: perfecto, quiero empezar a pedir
AI: Por supuesto, ¿qué plato le gustaría pedir primero?
-----
User: Una espagueti boloñesa
AI: ¿Qué tamaño prefiere para su Spaghetti Bolognese, tamaño regular o grande?
-----
User: De tamaño grande
AI: ¿Desea pedir otro plato más o alguna bebida?
-----
User: No eso es todo
AI: Restaurante Virtual: --Número de pedido : 4721
-- Plato 1 : Spaghetti Bolognese: Tamaño grande; Precio: 12.99 ; Cantidad : 1
-- Precio total : 12.99
-----
```

**Figura 4.25:** Ejemplo de conversación con el chatbot.

## Implementación del reconocimiento de voz y conversión de texto a voz

A pesar de que el objetivo final es implementar estos sistemas en un entorno móvil, se ha comenzado a implementarlos dentro del proyecto Python para así, poder realizar pruebas en el entorno de desarrollo local.

Como se explicó en la sección 2.4.6, el sistema tendrá que capturar y procesar la petición vocal del usuario desde el micrófono, convirtiéndolo en texto, para su procesamiento por el modelo de inteligencia artificial. A continuación, la respuesta generada por la inteligencia artificial debe ser transformada de texto a voz mediante Text-to-Speech (TTS).

Para la transformación de voz a texto, se ha utilizado la librería Python *speech.recognition* , como se detalló en secciones anteriores. Para ello, al presionar el botón 'Enter', el sistema comienza a grabar la voz . Finalmente , se configuró para que se detenga la grabación cuando se detecte 4 segundos de silencio.

Por otra parte, para convertir de texto a voz, se ha utilizado la librería Python *gTTS (Google Text-to-Speech)*, la cual emplea una voz por defecto de Google . Por cada respuesta que genera el chatbot, esta se guarda como archivo MP3 y enseguida, se reproduce dicho archivo gracias a la biblioteca Python *pygame*. Cada vez que el chatbot genere una nueva respuesta, se reemplazará el archivo MP3 anterior con el nuevo y se reproducirá su contenido.

```
def text_to_speech(text):
    # Directorio del script
    script_dir = os.path.dirname(__file__)

    # Nombre del archivo de salida MP3
    mp3_file = os.path.join(script_dir, "output.mp3")

    # Crear objeto gTTS
    tts = gTTS(text, lang='es-us')

    try:
        # Guardar el archivo MP3
        tts.save(mp3_file)
        print(f"Archivo MP3 guardado en: {mp3_file}")
    except Exception as e:
        print(f"Error al guardar el archivo MP3: {e}")

    return mp3_file

def reproducir_mp3(ruta_archivo):
    # Inicializar pygame
    pygame.init()

    # Inicializar el módulo de audio
    pygame.mixer.init()

    try:
        # Cargar el archivo MP3
        pygame.mixer.music.load(ruta_archivo)

        # Reproducir el archivo MP3
        pygame.mixer.music.play()

        # Esperar hasta que termine de reproducirse
        while pygame.mixer.music.get_busy():
            pygame.time.Clock().tick(10)

    except Exception as e:
        print(f"Error al reproducir el archivo MP3: {e}")

    finally:
        # Detener la reproducción y cerrar pygame
        pygame.mixer.music.stop()
        pygame.mixer.quit()
        pygame.quit()
```

**Figura 4.26:** Extracto código: Implementación de la conversión de texto a voz

Después de haber explicado cómo funcionan los sistemas de reconocimiento de voz y conversión de texto a voz, el siguiente paso sería implementar estos sistemas en el programa principal del modelo de Inteligencia Artificial. A continuación, se mostrará un extracto del código del fichero principal que contiene el chatbot. Este código incluirá las funciones principales para la interacción con el usuario, además de las llamadas a los sistemas de reconocimiento de voz y conversión de texto a voz:

```

def initMemoria():
    memory = ConversationBufferMemory(
        memory_key="history", #historial de la conversación a corto plazo del chatbot
        input_key="question"
    )
    return memory

def main():
    load_dotenv() # load environment variables (API_KEY)
    dataset = load_dataset("menu_dataset/menus_dataset.csv")
    chunks = create_chunks(dataset, 2000, 0)

    # Create or load the vector store, calculate retriever, init memory and defaultPrompt
    vector_store = create_or_get_vector_store(chunks)
    retriever1=calculate_retriever(vector_store,dataset)
    memory=initMemoria()
    print(memory)
    prompt=system_message_prompt_info #Default prompt

    while True:
        input("Press Enter to start recording...")
        speech_recognizer =SpeechRecognizer()
        query = speech_recognizer.insert_audio()
        print(query)

        # Si la pregunta del usuario es "Quiero empezar a pedir", cambiar el estado.
        if any(word in query.strip().lower() for word in ["quiero empezar a pedir", "deseo pedir", "ya quiero pedir","quiero pedir"]) and not any(word in query.strip().lower() for word in ["no"]):
            prompt=system_message_prompt_pedido
            response = get_conversation_chain(retriever1,prompt,query,memory)

        else:
            response = get_conversation_chain(retriever1,prompt,query,memory)

        # Convertir la respuesta a voz y reproducir
        mp3_file = text_to_speech(response)
        reproducir_mp3(mp3_file)

        if "Número de pedido" in response:
            parseo=parse(response)
            insert_bdd(parseo)
            sys.exit()

if __name__ == "__main__":
    main()

```

**Figura 4.27:** Extracto código: Implementación del fichero main

El fragmento de código mostrado en la Figura 4.26 representa la implementación principal del chatbot. Una característica destacable en este código es que se ha usado un patrón de coincidencia para determinar qué preprompt utilizar según la entrada del usuario. Como se explicó anteriormente, por defecto se utiliza el **Preprompt de información del menú**, pero en el caso de que el usuario exprese su deseo de realizar un pedido, se activa el **Preprompt de pedido**. Para lograr eso, se evalúa la entrada del usuario en busca de palabras clave que indiquen su intención de realizar un pedido, como 'quiero empezar a pedir' o 'deseo pedir'. Si se detecta alguna de estas expresiones y no se encuentra la palabra 'no' en la entrada del usuario, se cambia el preprompt al de pedido y se continúa con la lógica de la conversación con el único cambio de que a partir de ese momento, el asistente empieza a apuntar los platos que desee el usuario .

Otra característica destacable de este código es que cuando el usuario indica que ya no desea realizar más pedidos , entonces el chatbot genera y devuelve el ticket correspondiente al pedido actual. Posteriormente, el código parsea automáticamente este ticket junto con la lista de platos asociados y lo carga en la

base de datos.

### Implementación de la base de datos:

Una vez implementado el modelo de Inteligencia Artificial y las funcionalidades de reconocimiento de voz, el siguiente paso sería comenzar a implementar todas las tablas de la base de datos tal y como se diseñó en la sección [4.2.4](#).

En primer lugar, se ha comenzado a analizar cómo poder conectar el proyecto con la base de datos de MongoDB. Para ello, se ha usado una librería Python llamada **pymongo** la cual permite establecer una conexión a través de un puerto específico en localhost. Esta conexión facilita la inserción de datos en tablas específicas de la base de datos.

Tras ello, el siguiente paso ha sido introducir datos ficticios del menú de un restaurante ficticio en una tabla que se llamará 'Menú'. Se ha decidido mantener esta tabla prellenada con datos ficticios para cualquier empresa que desee usar la aplicación en el futuro. Dichos datos del menú podrán ser modificados por un administrador o empleado a través de la aplicación web según sea necesario.

Finalmente, se ha creado el resto de tablas tal y como se ha explicado en la sección [4.2.4](#).

### Implementación de la aplicación móvil:

Para implementar la aplicación móvil, se ha dividido el proceso en dos grandes bloques: el desarrollo del backend y el desarrollo del frontend.

**Backend :** En el desarrollo del backend, se ha implementado una API con *Flask* que sirve de intermediaria entre la base de datos y la aplicación móvil. Dicha API servirá para gestionar las solicitudes del usuario, procesando sus consultas y devolviendo las respuestas apropiadas. Para el desarrollo de la API, se han usado dos métodos para gestionar las solicitudes del usuario:

- **Solicitud POST para comunicarse con el asistente virtual :** Este método recibe un mensaje del usuario y la memoria con el historial actual de la conversación y a continuación, devuelve la respuesta del chatbot teniendo en cuenta el contexto y actualiza la memoria con la pregunta del usuario y la respuesta dada por el chatbot. Para ello, se llama a la ruta */chat*. Como vemos en la figura de a continuación, el código es bastante similar a la figura [4.27](#), pero adaptado para que funcione en la API.

```

# Inicializar variables globales
load_dotenv() # Carga las variables de entorno
dataset = load_dataset("menu_dataset/menus_dataset.csv")
chunks = create_chunks(dataset, 2000, 0)
vector_store = create_or_get_vector_store(chunks)
retriever = calculate_retriever(vector_store, dataset)

memory = initMemoria() # Ponemos la inicialización fuera de la función del enrutador
prompt = system_message_prompt_info

@app.route("/chat", methods=["POST"])
def chat_with_history():
    global memory, prompt # Declaramos que vamos a utilizar las variables globales "memory" y "prompt"
    data = request.get_json()
    user_question = data['user_question']
    # Si la pregunta del usuario es "Quiero empezar a pedir", cambiar el estado.
    if any(word in user_question.strip().lower() for word in ["quiero empezar a pedir", "deseo pedir", "ya quiero pedir", "quiero pedir"]) and not any(word in user_question.strip().lower() for word in ["no"]):
        prompt = system_message_prompt_pedido
    try:
        response = get_conversation_chain(retriever, prompt, user_question, memory)

        # Insertar el ticket en la base de datos si la respuesta contiene 'Número de pedido'
        if "Número de pedido" in response:
            parseo=parse(response)
            insert_bbdd(parseo)

    except Exception as e:
        return jsonify({'error': str(e)}), 500
    chat_history = [{ 'message': message.content, 'type': type(message).__name__ } for message in memory.chat_memory.messages]
    return jsonify({"response": response, "chat_history": chat_history})

```

**Figura 4.28:** Extracto código: Solicitud POST comunicándose con el chatbot

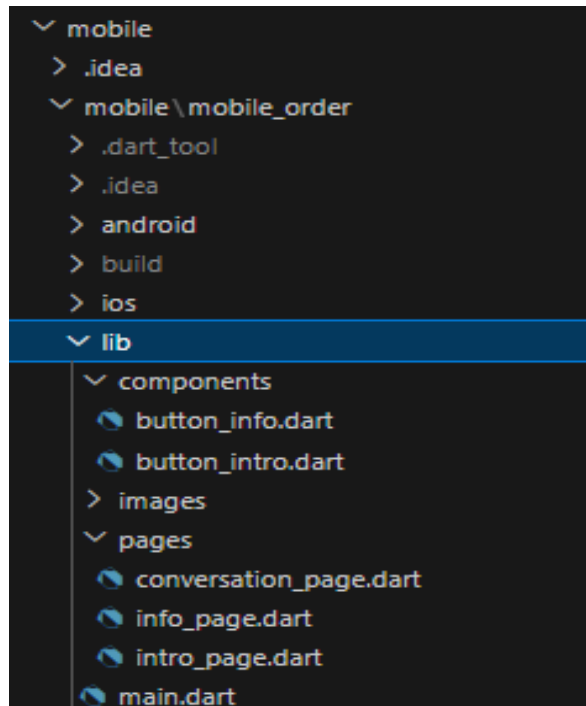
- **Solicitud POST para reiniciar la conversación:** Reinicia la memoria y el prompt a sus estados iniciales cuando llama a la ruta */reset*.

Cabe destacar que cada vez que el usuario inicia una nueva interacción con el chatbot, la memoria se reinicia automáticamente. Esto significa que cada vez que se ejecuta la aplicación, la memoria comienza vacía, garantizando que cada interacción con el chatbot inicie sin información previa de conversaciones anteriores.

**Frontend :** En paralelo, el desarrollo del frontend, se centra en la interfaz de usuario y la experiencia del usuario en la aplicación móvil. Para su desarrollo se ha usado el framework de Flutter. El primer paso ha sido la configuración de un emulador virtual para poder probar lo que se vaya haciendo de la aplicación móvil dentro de la computadora.

Una vez completada la aplicación móvil, la distribución de archivos y carpetas quedarían de la siguiente manera:





**Figura 4.29:** Distribución de las carpetas del proyecto Flutter

Como se aprecia en la figura 4.29, se han modificado únicamente las carpetas que están dentro de la carpeta *lib*. Dicha carpeta contiene la carpeta *pages*, que contiene las tres vistas que contiene la app móvil, la carpeta *images*, con todos los logos e imagenes de la aplicación y la carpeta *components*, que contiene los componentes funcionales de los botones.

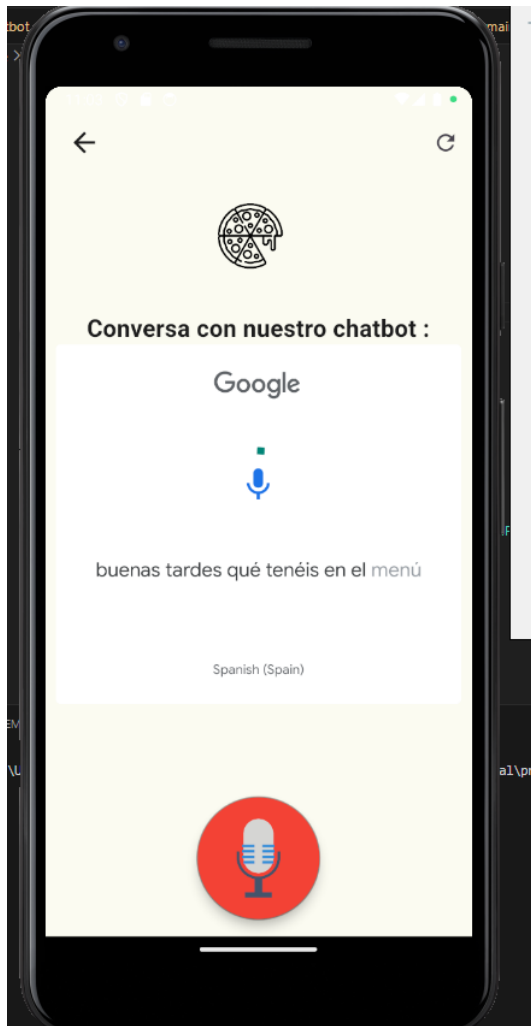
A continuación se mostrarán las vistas de la aplicación móvil. No se explicarán en detalle las funcionalidades de cada vista debido a que ya fueron discutidas anteriormente en la sección 4.2.2 :



Figura 4.30: Vista de introducción



Figura 4.31: Vista de instrucciones



**Figura 4.32:** Petición vocal al chatbot



**Figura 4.33:** Respuesta vocal del chatbot

Como se puede ver en la figura 4.32, el usuario pulsa en el botón de grabar voz, y a partir de ello, sale un modal para grabar la voz y convertirla a texto gracias a la librería *speech\_to\_text\_google\_dialog*. A partir de ello, el mensaje es procesado por la API y se obtiene una respuesta vocal y escrita del chatbot tal y como muestra en la figura 4.33.

Además y tal y como se explicaron en los requisitos, el usuario tiene la posibilidad de interrumpir lo que dice el chatbot cuando lo desee por si si desea agilizar la comunicación.

## Implementación de la aplicación web:

Igual que en la aplicación móvil, la implementación de la aplicación web se dividirá en dos grandes bloques, el frontend y el backend.

**Backend :** En el caso del backend de la aplicación, se ha implementado una nueva API con Flask que servirá de intermediario entre la aplicación móvil, la base de datos y la aplicación web.

En primer lugar, se han desarrollado métodos para gestionar los usuarios de la aplicación web. Para ello, se ha creado un método POST que permite registrar un nuevo empleado o administrador en la base de datos, y otro método POST que facilita el inicio de sesión de los usuarios en la aplicación. Además, se ha implementado un método GET que proporciona una lista de todos los empleados disponibles y otro método DELETE para borrar un usuario específico.

En relación con la gestión de la información del menú, se ha implementado un método GET para obtener todos los platos disponibles con su información, así como métodos POST, DELETE y PUT para añadir , borrar y modificar un plato respectivamente.

Más adelante, se han introducido métodos específicos para personalizar componentes de la aplicación móvil, como logotipos y títulos. Para ello, se ha implementado un método GET que devuelve dicha información (cabe destacar que los logotipos son devueltos en formato png) y también se ha implementado un método PUT para modificar dichos componentes.

Finalmente, se han creado métodos para gestionar los pedidos realizados por los clientes. Para ello, se ha introducido un método GET para obtener una lista con los detalles de todos los pedidos en curso , un método DELETE para borrar un pedido una vez esté listo para entregarse al cliente y un método POST para añadir un nuevo pedido a la tabla una vez el cliente termine de comunicarse con el chatbot.

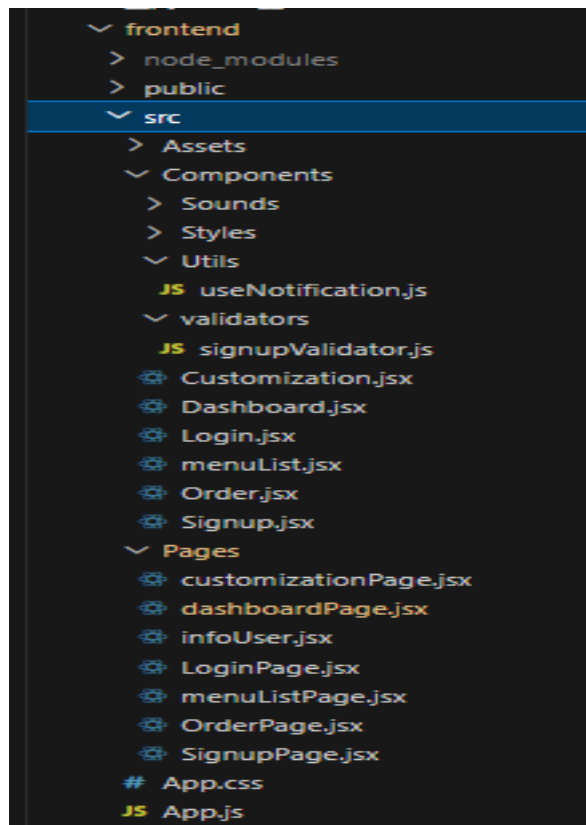
**Frontend :** Tras haber desarrollado todo el código mencionado anteriormente sobre el backend, el siguiente paso es realizar la parte visual de la aplicación web. Para ello, se ha usado ReactJS con Bootstrap con el fin de complementar la implementación del frontend. ReactJS se eligió por su capacidad de crear interfaces de usuario interactivas y dinámicas, mientras que Bootstrap facilita el diseño responsivo y la estructura visual de la aplicación web.

Antes de comenzar a programar la aplicación web, es importante detallar los proyectos contendrán dos elementos claves que se usarán para realizar la aplicación los cuales son las *Pages* y los *Components*.

En general, en nuestro proyecto, las *Pages* representan las distintas vistas de la aplicación. Estas páginas principalmente contienen la estructura HTML combinada con otros frameworks de diseño (*Bootstrap* en el caso de nuestro proyecto). Suelen manejar la lógica relacionada con las rutas .

Por otro lado, los *Components* constituyen la parte funcional y modular de la aplicación. Dichos componentes pueden ser reutilizables en diferentes partes de la aplicación.

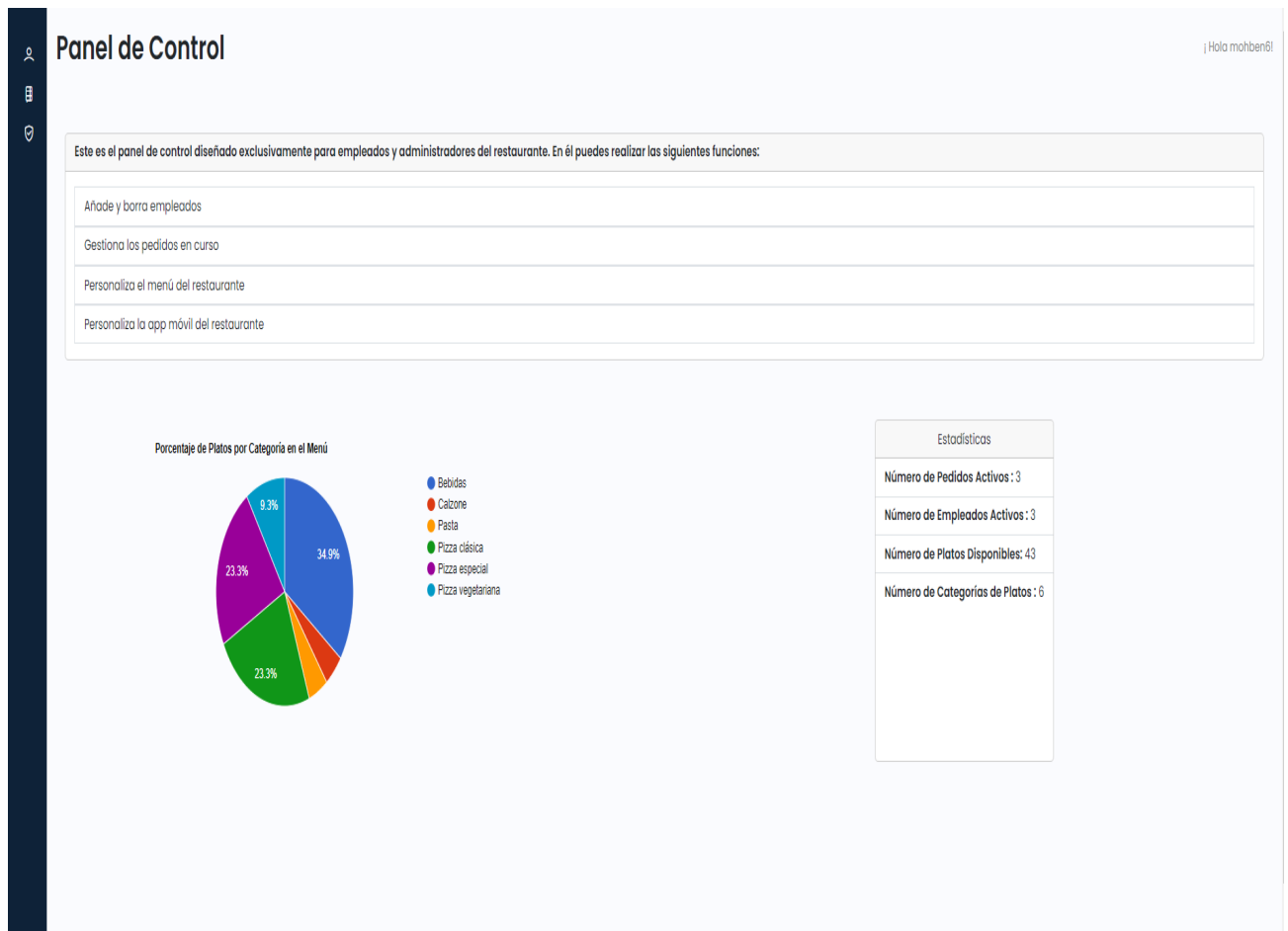
A continuación, se presenta la distribución de las carpetas tras completar la página web:



**Figura 4.34:** Distribución de las carpetas del proyecto ReactJS

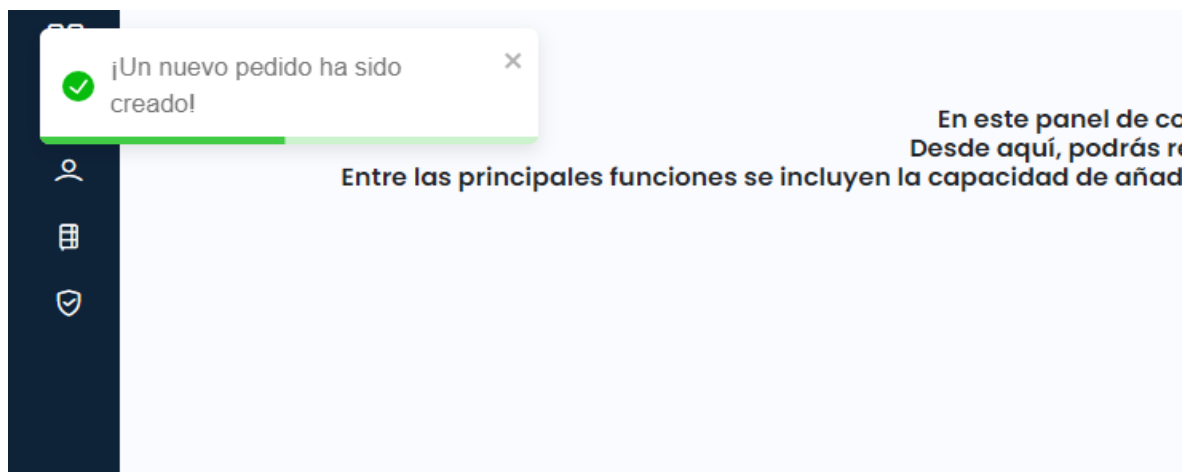
Se puede ver en la figura 4.34 que además de los components y pages mencionados anteriormente, existen otros archivos y carpetas interesantes que se deben explicar. El fichero *App.js* define las rutas y asociaciones entre las URLs y los componentes correspondientes. La carpeta *Validators* contiene funciones de validación para validar datos de formulario. La carpeta *Utils* contiene funciones reutilizables que pueden ser usadas en muchos componentes. En este proyecto, contiene la función de notificaciones que muestra notificaciones en cada una de las vistas de la aplicación, una vez un cliente realiza un pedido de manera correcta en la aplicación móvil.

A continuación se muestran las vistas más relevantes de la aplicación web y se explicarán sus funcionalidades de manera general. Esto se debe a que los detalles específicos de cada vista han sido previamente abordados en la sección 4.2.3 :



**Figura 4.35:** Panel de control de la aplicación web

La figura 4.35 representa el panel de control en la que los empleados o administradores acceden tras iniciar sesión. Se puede apreciar una barra lateral (sidebar) que proporciona acceso a todas las demás vistas disponibles. Además, aparece un mensaje de bienvenida personalizado mencionando al empleado conectado y un breve texto introductorio que describe las funcionalidades de la aplicación. Finalmente, se pueden apreciar unas estadísticas de que podrán servir al usuario .

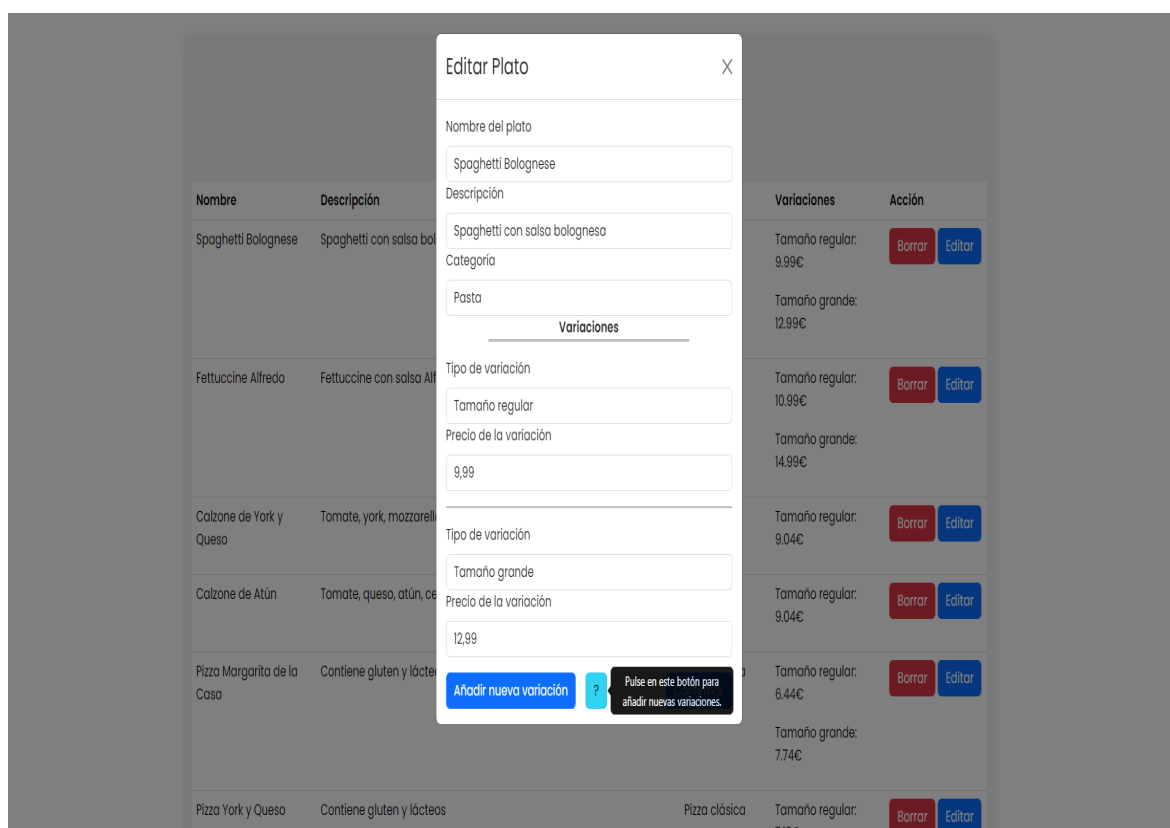


**Figura 4.36:** Notificación de un pedido

Se puede apreciar que en la figura 4.36 aparece una notificación indicando que un nuevo pedido ha sido creado. Esta notificación aparece cada vez que un cliente realiza un pedido en la aplicación móvil. Dicha notificación será visible en cada una de las vistas de la aplicación web .

← Lista de platos disponibles				
Añadir nuevo plato				
Nombre	Descripción	Categoría	Variaciones	Acción
Spaghetti Bolognese	Spaghetti con salsa bolognesa	Pasta	Tamaño regular: 9.99€  Tamaño grande: 12.99€	Borrar Editar
Fettuccine Alfredo	Fettuccine con salsa Alfredo cremosa	Pasta	Tamaño regular: 10.99€  Tamaño grande: 14.99€	Borrar Editar
Calzone de York y Queso	Tomate, york, mozzarella y ricotta	Calzone	Tamaño regular: 9.04€	Borrar Editar
Calzone de Atún	Tomate, queso, atún, cebolla, aceitunas y ricotta	Calzone	Tamaño regular: 9.04€	Borrar Editar
Pizza Margarita de la Casa	Contiene gluten y lácteos	Pizza clásica	Tamaño regular: 6.44€  Tamaño grande: 7.74€	Borrar Editar

**Figura 4.37:** Lista de platos disponibles en el menú



**Figura 4.38:** Modal para editar un plato

En las figuras 4.37 y 4.38 , se muestra como sería la vista que presenta la lista de platos disponibles en el restaurante. En la parte superior, aparece un botón para añadir nuevos platos, y a la derecha de cada plato, aparecen dos botones adicionales: uno para borrar y otro para editar las características del plato. En caso de pulsar en el botón de editar un plato, se abre un modal que permite modificar cualquier plato. Lo mismo ocurre cuando se desea añadir un nuevo plato.



← Personalización de la Aplicación Móvil

Nombre del Establecimiento

restaurantTFG

Logo Principal

PIZZA

Logo Secundario

Personalizar elementos

**Figura 4.39:** Personalización de la aplicación móvil

Editar Datos de Personalización ? No es necesario rellenar todos los campos

← Pers

Nombre del Restaurante

Nombre del Restaurante

Logo Principal

Choisir un fichier Aucun fichier choisi

Logo Secundario

Choisir un fichier Aucun fichier choisi

Cerrar Guardar Cambios

Logo Secundario

Personalizar elementos

**Figura 4.40:** Modal para personalizar

En las figuras 4.39 , se encuentra un formulario con los elementos actuales que componen la aplicación móvil (nombre del establecimiento, logo principal y logo secundario). Al pulsar en el botón de personalizar, aparece un modal (figura 4.40) donde podemos modificar dichos datos.

← Lista de pedidos en curso

Número de pedido	Platos	Cantidad	Precio Total	Acción
4729	Nombre: Botella de Agua, pequeña, Tamaño: Único	Cantidad: 1	14.29€	¡ Pedido listo !
	Nombre: Spaghetti Bolognese, Tamaño: Grande	Cantidad: 1		
4721	Nombre: Spaghetti Bolognese, Tamaño: Tamaño grande	Cantidad: 1	14.29€	¡ Pedido listo !
	Nombre: Botella de Agua, pequeña, Tamaño: Único	Cantidad: 1		

**Figura 4.41:** Lista de pedidos en curso

Finalmente, en la figura 4.41, se puede ver los pedidos en curso. Cada vez que un cliente hace un pedido, se añade un nuevo pedido en la tabla. Una vez el pedido preparado, se pulsa en el botón *Pedido listo* para borrarlo de la base de datos.

## 5. Pruebas

---

### 5.1. Introducción

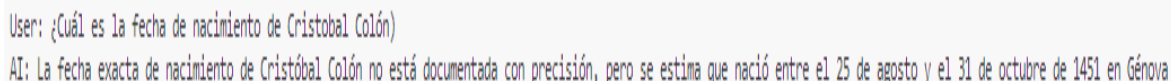
En este capítulo se detallarán las pruebas realizadas para asegurar de que cada uno de los bloques del proyecto funcione de manera correcta. Además, se explicarán algunos de los errores detectados durante dichas pruebas.

### 5.2. Pruebas del sistema

#### Pruebas en el modelo de Inteligencia Artificial :

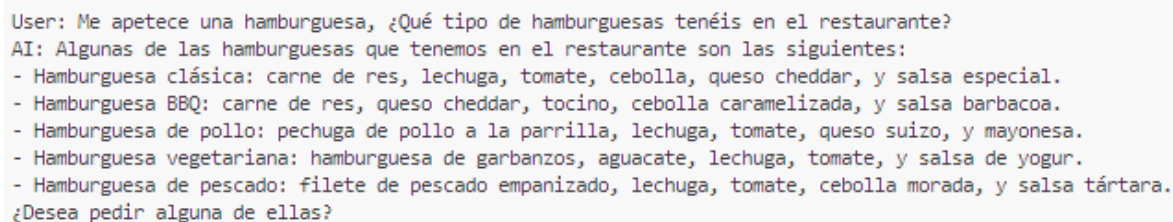
En esta fase, se realizaron varias pruebas escribiendo muchas peticiones al chatbot para analizar su capacidad de respuesta y su precisión.

Inicialmente, se ha usado como LLM *GPT-3.5 Turbo*. Sin embargo, se observó que en varias ocasiones el chatbot mostraba problemas al mantenerse dentro del contexto esperado y muchas veces respondía sobre temas que no tenían nada que ver con el establecimiento.



```
User: ¿Cuál es la fecha de nacimiento de Cristobal Colón)
AI: La fecha exacta de nacimiento de Cristóbal Colón no está documentada con precisión, pero se estima que nació entre el 25 de agosto y el 31 de octubre de 1451 en Génova
```

**Figura 5.1:** Prueba 1 con GPT-3.5 Turbo

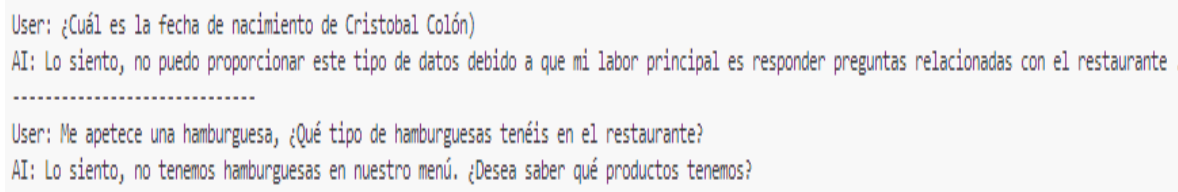


```
User: Me apetece una hamburguesa, ¿Qué tipo de hamburguesas tenéis en el restaurante?
AI: Algunas de las hamburguesas que tenemos en el restaurante son las siguientes:
- Hamburguesa clásica: carne de res, lechuga, tomate, cebolla, queso cheddar, y salsa especial.
- Hamburguesa BBQ: carne de res, queso cheddar, tocino, cebolla caramelizada, y salsa barbacoa.
- Hamburguesa de pollo: pechuga de pollo a la parrilla, lechuga, tomate, queso suizo, y mayonesa.
- Hamburguesa vegetariana: hamburguesa de garbanzos, aguacate, lechuga, tomate, y salsa de yogur.
- Hamburguesa de pescado: filete de pescado empanizado, lechuga, tomate, cebolla morada, y salsa tártara.
¿Desea pedir alguna de ellas?
```

**Figura 5.2:** Prueba 2 con GPT-3.5 Turbo

En la primera prueba de la figura 5.1, se puede ver que el chatbot responde a una pregunta que no tiene nada que ver con el tema del restaurante. Además, en la figura 5.2 se puede ver que el chatbot responde diciendo que el restaurante contiene hamburguesas y empieza a inventarse distintos tipos de hamburguesas, a pesar de que en el csv del menú no se encuentra ninguna hamburguesa. Para solucionar este problema, se ha ajustado la temperatura de precisión de los

mensajes que responde el chatbot y se ha decidido cambiar de LLM. En este intento, se ha usado como LLM *GPT-4-1106-Preview* :



```
User: ¿Cuál es la fecha de nacimiento de Cristobal Colón)
AI: Lo siento, no puedo proporcionar este tipo de datos debido a que mi labor principal es responder preguntas relacionadas con el restaurante .
-----
User: Me apetece una hamburguesa, ¿Qué tipo de hamburguesas tenéis en el restaurante?
AI: Lo siento, no tenemos hamburguesas en nuestro menú. ¿Desea saber qué productos tenemos?
```

**Figura 5.3:** Prueba con GPT-4-1106-Preview

En la figura 5.3 , se puede ver que con los nuevos ajustes el chatbot responde de manera correcta a las peticiones que hace el usuario. Además, no olvida el contexto en ningún momento, incluso cuando la conversación es larga.

### **Pruebas del sistema de reconocimiento de voz :**

Durante esta fase, las pruebas consistieron en grabar la voz en distintos lugares para garantizar una precisión óptima en la conversión de voz del usuario a texto. En primer lugar, se llevaron a cabo pruebas en un domicilio sin ruido de fondo, donde se grabó la voz con éxito, verificándose la exactitud en la transcripción de la voz del usuario a texto .

En segundo lugar, se realizaron pruebas en una sala de estudio, la cual tenía un entorno con ruido de fondo moderado. A pesar de ese nivel de ruido, se logró transcribir exactamente lo que el usuario decía a texto.

Finalmente, se realizaron pruebas en un entorno donde el ruido era bastante elevado, en este caso en una terraza de un restaurante, para simular al máximo un escenario de uso real de la aplicación. Inicialmente, se enfrentó dificultades para transcribir de manera precisa debido al ruido. Sin embargo, al usar auriculares con micrófono integrado, se consiguió una transcripción más precisa.

Por lo tanto, como conclusión de esta evaluación, se recomienda el uso de auriculares con micrófono integrado para interactuar con el chatbot para tener una experiencia óptima.

### **Pruebas en la aplicación móvil :**

Para comprobar el correcto funcionamiento de la aplicación móvil, se ha comprobado en primer lugar de que la API establece correctamente la conexión con la base de datos y el LLM . Además, se ha comprobado de que una vez el cliente termina de pedir un pedido, se actualiza la base de datos añadiendo dicho pedido. En tercer lugar, se verificó de que el botón para reiniciar el historial de la conversación actual funcionaba correctamente.

Finalmente, se evaluó la responsabilidad de la aplicación probándola en distintos dispositivos con diferentes tamaños de pantalla.

### **Pruebas en la aplicación web :**

Para comprobar el correcto funcionamiento de la aplicación web, se ha asegurado en primer lugar de que la conexión con la base de datos se hace de forma correcta. Se ha comprobado la funcionalidad de inicio de sesión y cierre de sesión, asegurando de que los usuarios puedan conectarse y desconectarse de forma exitosa.

Se verificó que una vez se borra un usuario de la base de datos, ya no puede acceder a la aplicación .

Se ha comprobado también la función de creación de un nuevo empleado de forma exitosa, asegurando que se apliquen las validaciones necesarias.

En la base de datos, se ha comprobado de que las contraseñas de los usuarios se almacenan de forma encriptada.

Se ha asegurado de que todas las páginas estén protegidas, es decir que únicamente el usuario conectado puede acceder a ella.

Se comprobó también de que todas las operaciones a las tablas funcionen correctamente (crear, borrar o modificar elementos de cada una de las tablas).

Finalmente, se confirmó que al realizar cambios, como por ejemplo modificar un elemento de la carta del restaurante , automáticamente dicha información se actualiza en la base de datos del asistente virtual para que responda correctamente a la petición del usuario.

## **5.3. Conclusión**

En este capítulo se concluye de que a lo largo del desarrollo del proyecto, se llevaron a cabo una variedad de pruebas de distinto tipo según la situación en la que nos encontrábamos. Estas pruebas fueron fundamentales para identificar fallos en el código y partes a mejorar, para así realizar correcciones y ajustes significativos en el código, con el objetivo de alcanzar los objetivos fijados.

## 6. Manual de instalación y de uso

---

Este capítulo es un extracto del *readme.md* que se podrá encontrar en el GitHub del proyecto realizado:

<https://github.com/AmineBennani7/Smart-Voice-Restaurant-Assistant>.

En primer lugar se expondrá el **manual de instalación** para poder configurar el proyecto por primera vez:

### 1. Clonar el repositorio:

```
1 git clone https://github.com/AmineBennani7/Proyecto-Reconocimiento-  
Vocal
```

### 2. Instalar Dependencias:

Ejecutar el siguiente comando en la terminal:

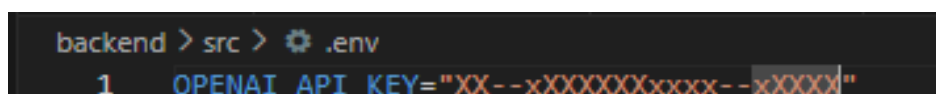
```
1 pip install -r requirements.txt
```

### 3. Generación de una clave personal en OpenAI:

Para usar el modelo GPT-4-1106-preview de OpenAI en la aplicación personal, se necesita generar una clave personal en la plataforma OpenAI. Para ello se tiene que seguir los siguientes pasos:

- Ir a la página oficial de [OpenAI](#) y crear una cuenta si aún no se tiene una.
- Ir a la sección de configuración de API y buscar la opción para **generar una nueva clave de API**. Dicha clave se tiene que mantener siempre y no se debe compartirla con nadie.
- Crear un nuevo entorno virtual **.env** e introducir la clave secreta generada en OpenAI.

```
1 cd backend/src/chatbot/ && python -m venv .env
```



```
backend > src > .env  
1 OPENAI_API_KEY="XX--xXXXXXXXXxxxx--xXXX"
```

Figura 6.1: Introducción de la clave secreta OpenAI

#### 4. Ejecutar los siguientes ficheros :

- a) Cargar los elementos del menú por defecto en la base de datos, los cuales luego el administrador podrá modificar a través de la aplicación web.

```
1 python "backend/src/database/load_data.py"
2
```

---

- b) Transformar los elementos del menú de la base de datos a un formato CSV que pueda ser leído correctamente por el chatbot.

```
1 python "backend/src/chatbot/get_menu_dataset.py"
2
```

---

En segundo lugar se expondrá el **manual de uso** para poder usar correctamente el programa cada vez que se quiera ejecutar:

##### 1. Ejecutar el backend para el correcto funcionamiento del chatbot:

```
1 python "backend/src/api/api_main.py"
2
```

---

##### 2. Ejecutar el backend para el correcto funcionamiento del servidor web:

```
1 python "web/server/app.py"
2
```

---

##### 3. Ir al siguiente directorio y ejecutar main.dart para hacer funcionar la aplicación móvil. Previamente, se debería tener configurado el emulador móvil.

```
1 cd .\mobile\mobile\mobile_order\lib
2
```

---

##### 4. Ejecutar la aplicación web para los empleados y administradores.

```
1 cd .\web\frontend ; npm start
2
```

---

# 7. Conclusiones y planes futuros

---

## 7.1. Conclusiones

Después de tantos meses de preparación e implementación, se han alcanzado los resultados que se esperaban al principio de la adjudicación de este trabajo. Como resultado final del proyecto, se ha conseguido realizar un sistema formado por una aplicación móvil que servirá para que los clientes puedan realizar peticiones a un modelo de Inteligencia Artificial y realizar pedidos de comida a un restaurante. Además, también se ha conseguido realizar con éxito una aplicación web donde los empleados o administradores realizan ajustes en el menú y en la aplicación web, además de gestionar los pedidos actuales de los clientes.

El desarrollo de este proyecto ha permitido el aprendizaje y el afianzamiento de varias tecnologías y frameworks como ReactJS, Flutter o Langchain. Además, se ha aprendido a como conectar con LLMs y realizar peticiones a ellos realizando los ajustes adecuados según los objetivos propuestos.

También, se ha ampliado el conocimiento en las bases de datos, especialmente en las no relacionales (NoSQL) usando MongoDB y comprendiendo la flexibilidad que conlleva su uso.

Otro tema muy interesante que se exploró durante este proyecto, fue el reforzamiento del concepto de las APIs y su importancia para la realización de conexiones entre servidores multiplataforma. En el proyecto, las APIs fueron utilizadas para realizar conexiones entre varios elementos clave, como la interacción entre la aplicación móvil y la base de datos para la gestión de pedidos, así como la integración con una API de LLM para habilitar funciones de Procesamiento de Lenguaje Natural en el chatbot. Se aprendió a cómo implementar y utilizar estas APIs para poder ejecutar procesos en segundo plano.

En relación al tiempo invertido para la realización del proyecto, hubo un ligero retraso respecto al tiempo planificado establecido en la sección 3.3. Para analizar las horas dedicadas a cada una de las tareas, se ha usado la extensión *Clockify*. En el caso de las horas invertidas finales, se ha trabajado 335 horas, lo cual se alinea bastante al tiempo planificado inicialmente. Sin embargo, debido a la carga con las demás asignaturas de la carrera, hubo días donde no se trabajó en el proyecto por lo que se finalizó el proyecto unos días más tarde de lo previsto. Inicialmente, se tenía previsto completar y revisar la documentación para el 26 de abril, pero finalmente se extendió hasta el **12 de mayo**.

Finalmente, según mi experiencia personal, he disfrutado mucho realizando este proyecto, aprendiendo y dominando nuevas tecnologías y conceptos. Además, ha sido un placer colaborar con el profesor Francisco Javier Ortega, quien siempre ha sido disponible para darme consejos y orientación en cualquier momento del proceso. Este proyecto me ha servido para asentar las bases para abordar y desarrollar futuros chatbots personalizados de manera eficiente.



Además, considero que este proyecto refleja el avance significativo en el desarrollo de los asistentes virtuales y su impacto en nuestra sociedad en un futuro muy cercano. Considero que los asistentes virtuales personalizados, aplicables en cualquier ámbito y no solo en el sector de la hostelería, pueden convertirse en compañeros inteligentes y omnipresentes en nuestras vidas diarias, mejorando nuestra eficiencia y nuestra experiencia en múltiples aspectos.

Pienso que en el ámbito laboral, los asistentes virtuales personalizados van a ser clave para el aumento de la eficiencia y productividad, debido a que se automatizarán procesos y se facilitará la comunicación entre equipos, lo que permitirán a los empleados dedicar más tiempo en realizar actividades más importantes.

## 7.2. Planes futuros

En un futuro, se tiene la intención de mejorar los siguientes aspectos para que el usuario tenga una mejor experiencia a la hora de usar el programa:

- Poder hacer un sistema de despliegue de la aplicación web y móvil para que las empresas o personas interesadas puedan usar el programa con toda comodidad.
- Implementar mejoras en el sistema de pattern matching para que el chatbot pueda identificar de manera más natural cuándo un usuario desea realizar un pedido, sin depender de frases clave específica. Se recuerda que actualmente, el chatbot requiere que usuario utilice una frase que contenga *'Quiero empezar a pedir'* o *'Quiero pedir'* y asegurarse de que estas frases no contengan la palabra *'no'* para iniciar el proceso de pedido de manera adecuada tal y como se explica en la sección [4.3.2](#).

# Bibliografía

---

- [1] Eleni Adamopoulou and Lefteris Moussiades. *Chatbots: History, Technology, and Applications*. 2020.
- [2] Chatbot Team. Everything you need to know about machine learning chatbot 2023. <https://www.linkedin.com/pulse/everything-you-need-know-machine-learning-chatbot-2023-chatbot-team/>.
- [3] Sanket Sanjay Patil. Seminar report on natural language processing (nlp). Internal report, 2021-2022 2022. Submitted in Partial Fulfillment for the award of the degree of Master of Computer Applications.
- [4] Neon tools community - chatbots. <https://community.neontools.io/chatbots/>.
- [5] A comparison between alice and elizabeth. <https://eprints.whiterose.ac.uk/81930/1/AComparisonBetweenAliceElizabeth.pdf>.
- [6] Eleni Adamopoulou and Lefteris Moussiades. *An Overview of Chatbot Technology*. Kavala, Greece, 2020.
- [7] Locall Host. What kind of algorithm does siri use? <https://locall.host/what-kind-of-algorithm-does-siri-use/>.
- [8] Juan Sensio. Understanding attention mechanisms in deep learning. [https://juansensio.com/blog/060\\_attention](https://juansensio.com/blog/060_attention).
- [9] Baeldung. Chatgpt model overview. <https://www.baeldung.com/cs/chatgpt-model>.
- [10] Amanatulla. Fine-tuning the model: What, why, and how. Medium, 2021. URL <https://medium.com/@amanatulla1606/fine-tuning-the-model-what-why-and-how-e7fa52bc8ddf>.
- [11] Yue Claire Xiao. Pre-training, Fine-Tuning, & Prompt Engineering. <https://www.linkedin.com/pulse/wtf-pre-training-fine-tuning-prompt-engineering-yue-claire-xiao/>, 2021.
- [12] Create a chatbot in python with langchain and rag. <https://medium.com/mlearning-ai/create-a-chatbot-in-python-with-langchain-and-rag-85bfba8c62d2>.
- [13] José Manuel Carpintheyro Sánchez. Retrieval-augmented generation (rag): Una revolución en los modelos de lenguaje. <https://www.linkedin.com/pulse/retrieval-augmented-generation-rag-una-revoluci%C3%B3n-los-jos%C3%A9-manuel/?originalSubdomain=es>.

- [14] Gustavo Espíndola. Chunk division and overlap: Understanding the process. <https://gustavo-espindola.medium.com/chunk-division-and-overlap-understanding-the-process-ade7eae1b2bd>.
- [15] SpeechRecognition Contributors. Speechrecognition. <https://pypi.org/project/SpeechRecognition/>.
- [16] Wikipedia contributors. Desarrollo en cascada. [https://es.wikipedia.org/wiki/Desarrollo\\_en\\_cascada](https://es.wikipedia.org/wiki/Desarrollo_en_cascada), 2024.
- [17] Langchain conversational memory. <https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/>.
- [18] What are Tokens and How to Count Them. <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>.