

El chatbot culinario

Nombre Estudiante

Susana José Andrade Nóbrega
Grado de Ingeniería Informática
75.629 – TFG – Inteligencia artificial

Nombre Consultor

David Isern Alarcón

Nombre Profesor responsable de la asignatura

Xavier Baró Solé

20/06/2023



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Agradecimientos

Agradezco a mi familia por todo el apoyo incondicional durante esta etapa. Especialmente, quiero agradecer a mi pareja por haber estado a mi lado, apoyándome tanto en los momentos positivos como en los más desafiantes a lo largo de este proyecto.

A mi consultor David Isern Alarcón, por su apoyo, dedicación y por compartir sus conocimientos. A mi tutora y los docentes de la Universitat Oberta de Catalunya por todos sus conocimientos transmitidos a lo largo de este trayecto.

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>El chatbot culinario</i>
Nombre del autor:	<i>Susana José Andrade Nóbrega</i>
Nombre del consultor/a:	<i>David Isern Alarcón</i>
Nombre del PRA:	<i>Xavier Baró Solé</i>
Fecha de entrega (mm/aaaa):	06/2023
Titulación:::	<i>Grado de Ingeniería Informática</i>
Área del Trabajo Final:	<i>75.629 – TFG – Inteligencia artificial</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>chatbot, aprendizaje automático, alimentación</i>

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Según la OMS, la obesidad infantil y juvenil en España es de un 14,2% [1] y en los adultos, uno de cada cinco sufre de obesidad [2]. Además, se evidencia un incremento en la carencia de prácticas alimentarias saludables y equilibradas en la sociedad, como consecuencia del estilo de vida, la falta de motivación y la limitación de tiempo.

Para enfrentar esta problemática, se propone desarrollar un chatbot que promueva la alimentación equilibrada y saludable. A diferencia de las aplicaciones que ya existen, se busca proporcionar una experiencia más personalizada al usuario, a través de una conversación en tiempo real. El chatbot será capaz de ayudar al usuario en la planificación de las comidas, considerando sus preferencias alimenticias, y proporcionar información nutricional de las recetas y alimentos.

El desarrollo del chatbot implica diferentes etapas: análisis, diseño, desarrollo, implementación y pruebas. Durante el análisis y diseño, se definen los casos de uso del chatbot y las intenciones. A través de librerías específicas de procesamiento de lenguaje natural y aprendizaje automático en Python, se construye un sistema capaz de identificar patrones en un conjunto de datos de intenciones para realizar predicciones.

Como resultado, se obtiene un chatbot que, mediante una red neuronal entrenada y las intenciones identificadas, tiene la capacidad de predecir la intención del usuario y proporcionar una de las respuestas predefinidas,

complementadas con información de la base de datos. Es posible que algunas intenciones sean incorrectamente predichas debido a la similitud de los datos, cuando hay elementos comunes en las diferentes intenciones.

Abstract (in English, 250 words or less):

According to the WHO, childhood and juvenile obesity in Spain is 14.2% [1] and in adults, one in five suffers from obesity [2]. In addition, there is evidence of an increase in the lack of healthy and balanced eating practices in society, as a consequence of lifestyle, lack of motivation and time constraints.

To address this issue, it is proposed to develop a chatbot that promotes balanced and healthy eating. Unlike existing applications, it seeks to provide a more personalized experience to the user, through a real-time conversation. The chatbot will be able to assist the user in meal planning, considering their food preferences, and provide nutritional information on recipes and food.

The development of the chatbot involves different stages: analysis, design, development, implementation and testing. During analysis and design, the chatbot use cases and intentions are defined. Through specific natural language processing and machine learning libraries in Python, a system capable of identifying patterns in a data set of intentions to make predictions is built.

As a result, a chatbot is obtained that, through a trained neural network and the identified intentions, has the ability to predict the user's intention and provide one of the predefined answers, complemented with information from the database. It is possible that some intentions are incorrectly predicted due to the similarity of the data, when there are common elements in the different intentions.

Índice

Contents

El chatbot culinario	1
1. Introducción.....	1
1.1 Contexto y justificación del Trabajo	1
1.2 Objetivos del Trabajo.....	2
1.3 Enfoque y método seguido.....	3
1.4 Planificación del Trabajo	3
1.5 Breve sumario de productos obtenidos	6
1.6 Breve descripción de los otros capítulos de la memoria.....	8
2. Estado del arte	10
3. Análisis y diseño.....	12
3.1 Definición de los posibles casos de uso	12
3.2 Elección de los datasets.....	14
3.3 Modelo de clasificación multi-etiqueta.....	15
3.4 Modelo de la base de datos	17
3.5 Cálculo de calorías y nutrientes	20
3.6 Flujo de conversión del chatbot.....	23
3.7 Definición de las intenciones: etiquetas, patrones y respuestas	24
4. Desarrollo e implementación	26
4.1 Desarrollo del modelo.....	26
4.2 Implementación del chatbot.....	28
4.3 Ajustes.....	33
5. Pruebas	33
6. Conclusiones.....	44
6.1 Conclusiones.....	44
6.2 Seguimiento de planificación y metodología	44
6.3 Líneas de trabajo futuro.....	45
7. Glosario	46
8. Bibliografía	47
9. Anexos	54
9.1 Anexo 1 – Ejemplo de algunas intenciones del archivo “intents.json”	54

Lista de figuras

Ilustración 1: Temporalización del proyecto.....	4
Ilustración 2: Diagrama de Gantt.....	4
Ilustración 3: Diagrama de casos de uso.....	12
Ilustración 4: Diagrama de frecuencia de alergias.....	16
Ilustración 5: Modelo de la base de datos.....	18
Ilustración 6: Flujo de conversación del chatbot.....	23
Ilustración 7: Interfaz del Culinary Chatbot.....	31
Ilustración 8: Ventana para que el usuario introduzca sus datos.....	32

Lista de tablas

Tabla 1: Ejemplo de chatbots11

Tabla 2: Casos de uso.....13

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente, la malnutrición sigue siendo un problema a nivel global y de difícil erradicación. La deficiencia nutricional debe ser tomada en cuenta ya que se puede dar lugar a enfermedades cardiovasculares, como el infarto de miocardio, la diabetes y la obesidad. Además, en niños, puede llevar al retraso en el crecimiento.

El estilo de vida, la pereza o la falta de tiempo pueden ser algunos de los factores asociados a que la sociedad padezca de malnutrición, debido a la carencia de nutrientes y equilibrio en la alimentación.

En la actualidad, existen varias aplicaciones con el objetivo de, fomentar una dieta saludable y equilibrada para reducir la malnutrición. Sin embargo, con el *chatbot* culinario, se busca acercar al usuario de una manera más personalizada. El *chatbot* culinario puede ser considerado también como un complemento al asesoramiento de un nutricionista. La conveniencia y accesibilidad del *chatbot* culinario pueden motivar a los usuarios a adoptar hábitos alimentarios más saludables y, a su vez, promover un uso continuo y prolongado para mejorar su salud.

Lo que se pretende con la creación de este *chatbot*, es promover los hábitos alimentarios saludables, incrementando el aporte de nutrientes y equilibrio a la alimentación del usuario. El chat será capaz de facilitar al usuario recetas e información, teniendo en cuenta sus gustos, alergias o intolerancias alimentarias. Este chat tendrá la capacidad de identificar los alimentos y nutrientes que han sido facilitados en recetas anteriores. Considerando las porciones de nutrientes recomendadas por la OMS y otros criterios, el *chatbot* podrá proporcionar recetas donde se incluyan ingredientes para completar una dieta saludable y equilibrada, evitando así el exceso o el desequilibrio de calorías y nutrientes. Además, el *chatbot* culinario será capaz de devolver recetas con un determinado alimento, el valor nutricional de la receta y de cada alimento.

1.2 Objetivos del Trabajo

Objetivos generales

- Desarrollar un *chatbot* capaz de comprender y contestar correctamente al usuario, asesorando recetas teniendo en cuenta sus preferencias y promoviendo una dieta saludable y equilibrada.
- Uso de las herramientas de inteligencia artificial aprendidas a lo largo de la carrera.

Objetivos específicos

Objetivos principales:

- Utilizar un modelo de aprendizaje automático para entrenar los datos de un dataset de recetas para clasificar las recetas en diferentes categorías como veganas, sin gluten o lactosa.
- Crear una base de datos para guardar las características nutricionales de cada alimento, las recetas, la información del usuario y el histórico de las conversaciones.
- Implementar los cálculos para calcular la cantidad de calorías que deben ser consumidas por el usuario diariamente, teniendo en cuenta su peso, altura, edad y factor de actividad.
- Implementar las funciones y cálculos necesarios para ir calculando las calorías y los nutrientes ingeridos por el usuario, teniendo en cuenta los porcentajes recomendados por la OMS.
- Implementar las funciones necesarias para ir recomendando al usuario dietas variadas, evitando así la frecuencia de ingredientes repetidos entre comidas.
- Entrenar el chatbot para que este sea capaz de entender las preguntas del usuario y contestarlas.

Objetivos secundarios:

- Crear una interfaz para simular el *chatbot*.

1.3 Enfoque y método seguido

Como se había comentado anteriormente, la idea no es que el *chatbot* sea un libro de recetas, sino que promueva una experiencia cercana y personalizada para cada usuario. Es un chat que tiene como misión proporcionar lo mejor para el usuario a nivel de alimentación.

Esta idea ha surgido con el objetivo de poder integrar el *chatbot* a sitios web de recetas, a webs de servicio de entrega de comida a domicilio, sitios web de nutrición, en los supermercados (se podría integrar el *chatbot* a los carritos de compra para que los clientes pudieran consultarlo mientras realizan la compra de alimentos), a robots de cocina y asistentes virtuales como Alexa o aplicaciones.

Durante este proyecto, se desarrollará un *chatbot* personalizado con un enfoque específico, promover una alimentación equilibrada y saludable. Con el fin de alcanzar ese propósito, el chatbot ofrecerá asesoramiento en la planificación de las comidas, proporcionará información nutricional y adaptará sus recomendaciones a las preferencias del usuario. Para poder destacarlo de otras aplicaciones similares, se ofrecerá una experiencia personalizada e una comunicación activa al usuario, captando así su atención e interés.

Para llevar a cabo esta propuesta, se contará con el apoyo de los modelos del aprendizaje automático. El lenguaje de programación utilizado será Python debido a su sencillez y su capacidad para trabajar con grandes volúmenes de datos. Además, la abundancia de herramientas y librerías serán bastante útiles en el desarrollo del *chatbot* culinario.

El chatbot se creará desde cero y estará diseñado para ajustarse a las preferencias y necesidades nutricionales del usuario. Para mantener un historial de las recetas recomendadas anteriormente y las preferencias del usuario, se implementará una base de datos que permitirá realizar un seguimiento.

1.4 Planificación del Trabajo

Para planificar el trabajo que se llevará a cabo para desarrollar el *chatbot* y presentar el trabajo realizado, se han dividido las tareas en 5 fases. A continuación, se puede observar la temporalización del proyecto con un diagrama de Gantt.

Tareas	Fecha de inicio	Fecha de finalización	Duración (días)
Fase 1: Plan de trabajo	14-Mar	28-Mar	15
Definir los objetivos	14-Mar	20-Mar	7
Elaborar una planificación temporal	21-Mar	23-Mar	3
Valorar los riesgos	24-Mar	28-Mar	5
Fase 2: Desarrollo del trabajo - Fase I	29-Mar	03-May	36
Seleccionar los modelos de entrenamiento	29-Mar	31-Mar	3
Entrenar los modelos con los datos de los datasets	01-Apr	10-Apr	10
Crear una base de datos	11-Apr	15-Apr	5
Definir los posibles casos de uso del chatbot	16-Apr	19-Apr	4
Definir las intenciones y clasificarlas	20-Apr	25-Apr	6
Elaborar los datos de entrenamiento	26-Apr	01-May	6
Elaborar informe de seguimiento	02-May	03-May	2
Fase 3: Desarrollo del trabajo - Fase II	04-May	29-May	26
Implementar las funciones para las recomendaciones	04-May	07-May	4
Implementar el chatbot	08-May	15-May	8
Entrenamiento del chatbot	16-May	19-May	4
Realizar el testeo	20-May	23-May	4
Mejorar fallos y afinar las respuestas	24-May	25-May	2
Realizar pruebas finales	26-May	28-May	3
Elaborar informe de seguimiento	29-May	29-May	1
Fase 4: Redacción de la memoria	14-Mar	20-Jun	99
Fase 5.1: Elaboración de la presentación	21-Jun	25-Jun	5
Sintetizar los aspectos más importantes del TFG	21-Jun	22-Jun	2
Elaborar la presentación	23-Jun	25-Jun	3
Fase 5.2: Defensa pública	28-Jun	05-Jul	8

Figura 1 - Temporalización del proyecto

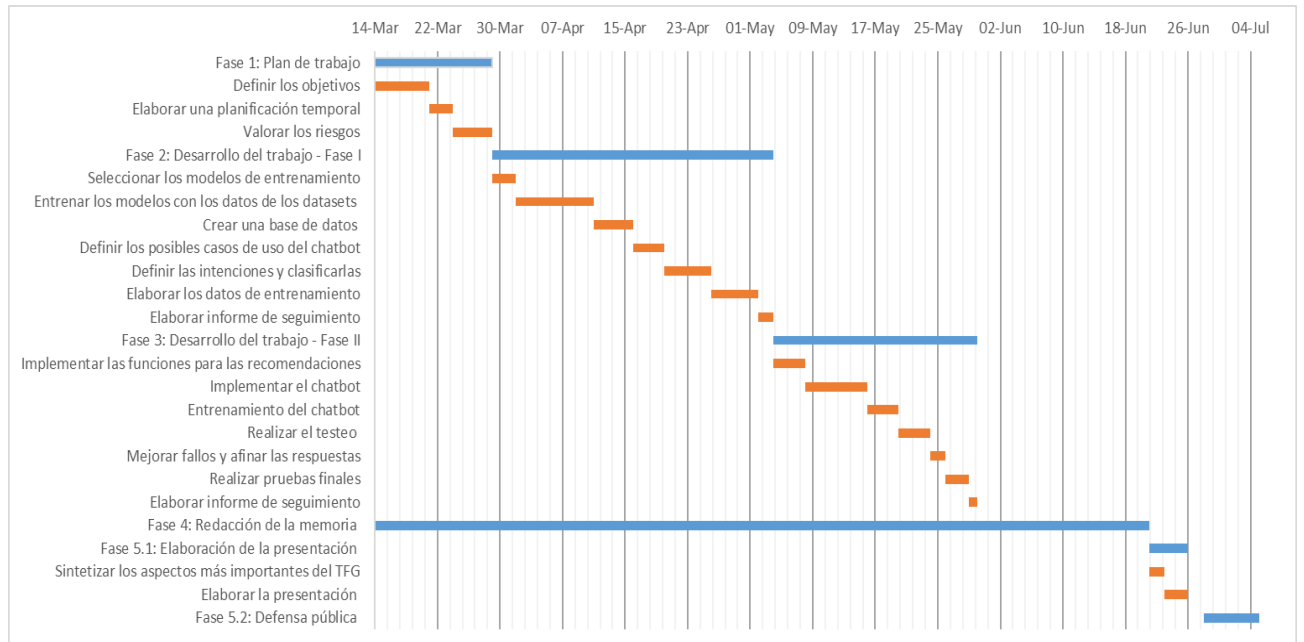


Figura 2 – Diagrama de Gantt

A continuación, se presentan las tareas y subtareas definidas para alcanzar los objetivos propuestos anteriormente:

Fase 1: Plan de trabajo

En esta fase se definen los objetivos del proyecto y se dividen las tareas en fases para poder lograr alcanzar los objetivos secuencialmente.

Además, se valorarán los riesgos que podrán surgir a lo largo del proyecto y las acciones para poder mitigarlos.

Fase 2: Desarrollo del trabajo – Fase I

Esta es una de las fases más importantes. En primer lugar, se tendrá que seleccionar los modelos de entrenamientos para clasificar las dietas (vegetariana y vegana) y las alergias. Después, se creará una base de datos para guardar toda la información sobre los datos necesarios y el historial de las conversas.

Por otro lado, también se creará un método para detectar que alimentos se han utilizado con mayor frecuencia en las recetas facilitadas por el *chatbot* teniendo en cuenta las preferencias.

Además, habrá que entrenar los modelos con los datos de los datasets.

Se otorgará mayor tiempo a la implementación del chatbot, ya que será una tarea bastante laboriosa.

Se elaborará un informe de seguimiento donde se reflexionará lo realizado hasta el momento.

Fase 3: Desarrollo del trabajo – Fase II

Esta etapa será dedicada a finalizar la implementación del modelo y a la realización de pruebas. Teniendo en cuenta los resultados obtenidos, se tomarán medidas para corregir los fallos y mejorar el desempeño del chatbot.

Además, se elaborará un informe de seguimiento similar al realizado en la fase anterior.

Fase 4: Redacción de la memoria

Esta fase será dedicada a la redacción de la memoria, respetando la plantilla de memoria propuesta por los profesores y teniendo en cuenta los aspectos de redacción de textos académicos. Es importante destacar que, a lo largo del proyecto, se irá incorporando información a la memoria a medida que se completen los objetivos.

Fase 5: Elaboración de la presentación

Fase 5.1: Elaboración de la presentación

En esta fase se elaborará la presentación, resumiendo de manera clara y precisa todo el trabajo realizado a lo largo del proyecto. Posteriormente, se preparará la presentación oral.

Fase 5.2: Defensa pública

Defender el proyecto, respondiendo a las preguntas proporcionadas por los profesores.

Además de la planificación, también se han identificado posibles riesgos que podrían surgir durante el desarrollo del proyecto y las posibles acciones de mitigación. A continuación, los enumeramos:

1. **Riesgo:** Que el *chatbot* no tenga en cuenta las preferencias del usuario. Si el *chatbot* recomienda una receta que contiene derivados lácteos y el usuario sufre de alergia a la lactosa, el *chatbot* dejaría de ser fiable.

Mitigación: Certificarse de que siempre que el *chatbot* recomiende una receta tenga en cuenta las preferencias mencionadas por el usuario. Como el histórico de recetas se guardará semanalmente/mensualmente, habrá que guardar las preferencias en la base de datos para no perderlas.

2. **Riesgo:** Incumplimiento de los plazos.

Mitigación: Revisar continuamente la temporalización del proyecto y el progreso de las tareas del proyecto.

1.5 Breve resumen de productos obtenidos

Al finalizar el proyecto, se han obtenido los siguientes productos:

- `intents.json`
En este archivo se definen las posibles intenciones del usuario. Este archivo incluye una colección de etiquetas con sus correspondientes patrones y respuestas asociadas. Las etiquetas y los patrones serán utilizados durante el entrenamiento del modelo (red neural), permitiendo realizar predicciones sobre las intenciones del usuario.
- `persistent_variables.json`
Con las variables definidas en este archivo, es posible almacenar información de forma persistente entre diferentes ejecuciones del programa.
- `culinary_chatbot.py`

En este archivo (programa principal) se importa el modelo entrenado `chatbotmodel.h5`, con el objetivo de predecir las etiquetas al analizar las preguntas del usuario. Esto permite identificar las intenciones del usuario y proporcionar respuestas adecuadas a cada intención. También se genera una simple interfaz para el usuario interactuar con el chatbot.

- `culinary_chatbot_database.py`
Se implementan funciones básicas necesarias para interactuar con la base de datos. Además, se desarrollan funciones específicas que facilitan la búsqueda de información en las tablas.
- `DataPreparation.py`
En este archivo se preparan los diferentes conjuntos de datos necesarios para este proyecto. Se incluyen también los modelos de clasificación multi-etiqueta desarrollados para clasificar las dietas y alergias en nuevas recetas.
- `nutrition_functions.py`
Contiene todas las funciones necesarias para calcular las calorías recomendadas, las calorías consumidas y las que faltan por consumir diariamente. Además, se calculan las cantidades de nutrientes recomendadas por la OMS que el usuario debe consumir diariamente. También se incluye una función para convertir el porcentaje de valor diario presente en una receta en la cantidad de gramos de nutrientes que aporta al usuario.
- `culinary_chatbot_model.py`
En este archivo se construye el modelo para entrenar el chatbot culinario.
- `chatbotmodel.h5`
Es el modelo entrenado resultante del modelo construido en `culinary_chatbot_model.py`, listo para ser utilizado en la predicción de las intenciones del usuario.
- `classes.pkl`
En este archivo se guardan las etiquetas definidas en `intents.json`.
- `words.pkl`
En este archivo se guardan todas las palabras sin duplicados y en su forma encontradas en los patrones definidos en `intents.json`.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria se divide en los siguientes capítulos:

Capítulo 1: Introducción

En este capítulo se exponen los objetivos del proyecto y se detalla la planificación de las tareas necesarias para alcanzarlos. También se proporciona un breve resumen de los productos obtenidos al finalizar el proyecto.

Capítulo 2: Estado del arte

En este apartado, se realiza un análisis del estado del arte de proyectos similares con el objetivo de identificar nuevos enfoques o metodologías que puedan ser aplicados en este proyecto.

Capítulo 3: Análisis y diseño

En este apartado se aborda el análisis y diseño del proyecto, que incluye la definición de los casos de uso y las intenciones del usuario. Se detalla la selección de los conjuntos de datos y el diseño del modelo de base de datos. Además, se describen los modelos de clasificación multi-etiqueta desarrollados para clasificar las dietas y alergias presentes en las recetas.

Capítulo 4: Desarrollo e implementación

En este capítulo se describe los pasos seguidos para el desarrollo e implementación del chatbot.

Capítulo 5: Pruebas

En este apartado se detallan las pruebas realizadas para verificar el correcto funcionamiento del chatbot, de acuerdo con las implementaciones realizadas.

Capítulo 6: Conclusiones

En las conclusiones se realiza una reflexión crítica sobre el logro de los objetivos planteados y un análisis del seguimiento de la planificación y de la metodología. Por otro lado, se exploran las líneas de trabajo futuro que quedan por abordar y las tareas pendientes de mejorar.

Capítulo 7: Glosario

En este apartado se incluyen los términos y acrónimos más relevantes utilizados dentro de la Memoria.

Capítulo 8: Bibliografía

En este capítulo se listan las referencias bibliográficas utilizadas dentro de la memoria y las que se han usado para consultar información.

Capítulo 9: Anexos

En anexos se incluyen las partes de código más extensas.

2. Estado del arte

En la actualidad, debido al aumento de la popularidad de los chatbots, es cada vez más común encontrar en los sitios web de las organizaciones un bot conversacional diseñado para proporcionar una atención al cliente eficiente. Estos chatbots tienen la capacidad de resolver las dudas de los clientes y cómo están disponibles las 24 horas del día y los 7 días de la semana, los clientes pueden obtener respuestas inmediatas a sus preguntas, eliminando así los tiempos de espera. Por lo tanto, el servicio al cliente puede mejorar significativamente. De hecho, un estudio realizado por Salesforce, reveló que el 59% de los clientes prefiere comunicarse con chatbots de marcas para despejar sus dudas debido a la mayor rapidez y agilidad en las respuestas [3].

Los asistentes virtuales impulsados por IA, ofrecen una gran cantidad de aplicaciones, ya que pueden ser personalizados para satisfacer las necesidades de cada organización. Es importante que el chatbot sea creado y adaptado para entender las necesidades de los clientes, para lo cual debe ser entrenado en los campos relevantes para abarcar las posibles preguntas y respuestas del usuario. El chatbot debe ser capaz también de guardar las preferencias del usuario y utilizarlas para mejorar su experiencia. Además, la retroalimentación por parte del usuario debe ser considerada para enriquecer el contenido del chatbot con nuevas intenciones.

A continuación, se presentan algunos ejemplos de los múltiples enfoques que pueden tener los asistentes virtuales, los cuales algunos abarcan áreas como la asistencia técnica, marketing, educación y la banca.

Autor	Enfoque
Francisco Domenes Mondragón	TFG – JackBot – El chatbot compañero [4] En este TFG se ha implementado un chatbot cuyo principal objetivo era proporcionar a las personas que se sienten solas una compañía virtual con quien puedan mantener una conversación.
Marcos Pino Gamazo	TFG – Chatbot para público infantil para clasificar sonidos del cielo [5] En este TFG se ha implementado un chatbot orientado al público infantil y a personas con problemas de visión para entretenimiento y educación.

Teresa Aguado Serrano	TFG – Tutor basado en chatbot [6] En este TFG se ha implementado un chatbot destinado a ayudar a los estudiantes de la UOC en la resolución de dudas relacionadas con sus estudios. Este sistema funciona de manera similar a un tutor.
Banco BBVA - Blue	La entidad bancaria BBVA cuenta con un chatbot diseñado para dar soporte a sus clientes, resolviendo sus dudas sobre cuestiones financieras [7].

Tabla 1 – Ejemplos de chatbots

En la actualidad, existen varias librerías de Python que brindan soporte para la creación de chatbots personalizados, entre ellas se destacan ChatterBot, Spacy, Rasa y TensorFlow [8]. Algunas de estas librerías se encuentran en constante evolución, lo que permite mejorar la calidad del chatbot. Con los nuevos avances de las tecnologías en inteligencia artificial, el chatbot puede ir evolucionando y ofreciendo nuevas funcionalidades para los usuarios [9].

Como se ha comentado anteriormente, la alimentación desequilibrada e poco saludable sigue siendo un problema en la actual sociedad. Para abordar este problema, se propone desarrollar un chatbot que promueva mejores hábitos alimenticios, ofreciendo orientación en la planificación de las comidas teniendo en cuenta sus preferencias. Además, el chatbot podrá transmitir cercanía con el usuario a través de la conversación y de la personalización de su interacción, proporcionando cualquier información que este necesite relacionada con los ingredientes y recetas. De este modo, los usuarios pueden sentirse acompañados durante su proceso de adopción de nuevos hábitos alimenticios. Como el chatbot culinario es fácil de usar, puede ser utilizado por personas de todas las edades y niveles de experiencia con la tecnología, destacándose así positivamente frente a las aplicaciones orientadas al mismo contexto.

3. Análisis y diseño

3.1 Definición de los posibles casos de uso

El chatbot culinario, será diseñado con una serie de casos de uso específicos relacionados con la alimentación y lo que engloba este tema. Estos casos de uso definen las competencias que el chatbot debe tener para proporcionar una experiencia eficaz al usuario. A continuación, se detallan algunos de los casos de uso identificados para el proyecto del chatbot culinario:

- Saludar el usuario.
- Recomendar recetas.
- Facilitar información nutricional relativa a las recetas e ingredientes.
- Planificación de las comidas.
- Control de las calorías y nutrientes ingeridos por el usuario.

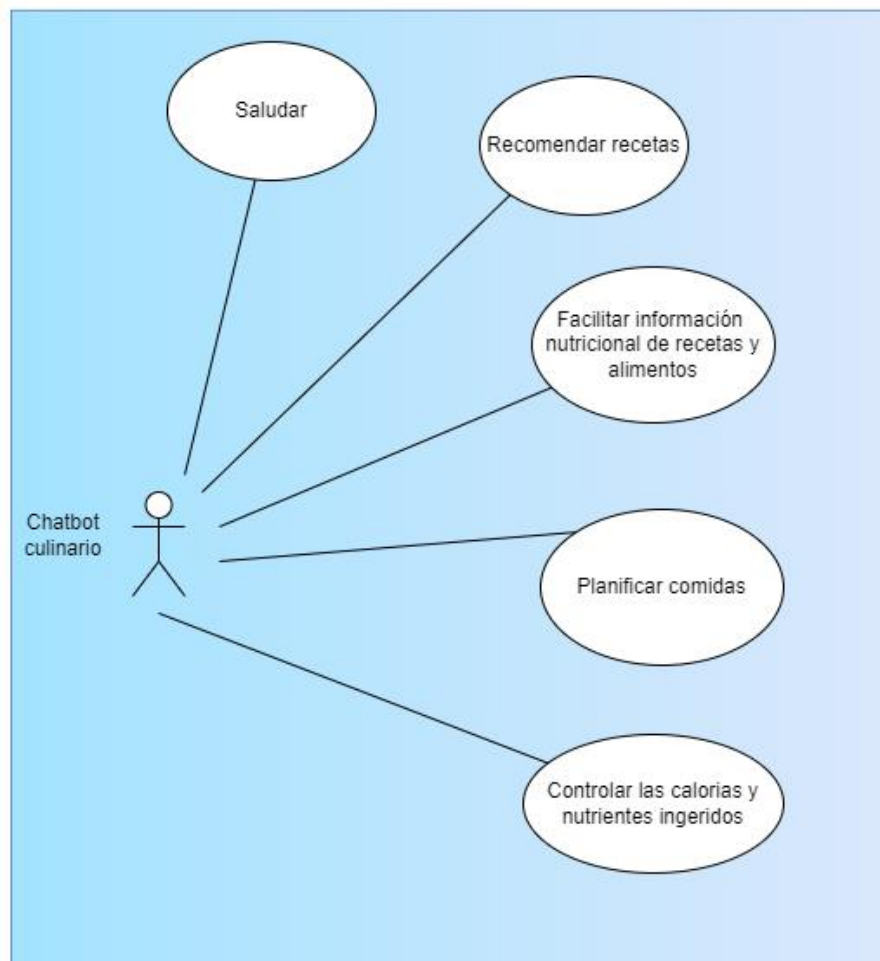


Figura 3 – Diagrama de casos de uso

En las tablas que se encuentran a continuación, se detallan los casos de uso.

Caso de uso	Descripción
Saludar	Aunque sea una intención básica, el saludo es vital para establecer una conexión entre el usuario y el chatbot al ser la primera iteración entre estos.
Recomendar recetas	<p>El chatbot debe ser capaz de recomendar recetas teniendo en cuenta las preferencias del usuario como pueden ser sus intolerancias o alergias y la cantidad de calorías ya consumidas.</p> <p>Al recomendar la receta, el chatbot debe incluir en su respuesta las siguientes características:</p> <ul style="list-style-type: none"> • nombre de la receta; • los ingredientes con sus respectivas cantidades; • los pasos para preparar la receta.
Facilitar información nutricional relativa a las recetas e ingredientes	<p>Facilitar información relativa a las recetas:</p> <ul style="list-style-type: none"> • información nutricional <ul style="list-style-type: none"> → Calorías; → Proteínas, → Grasa saturada, → Carbohidratos. <p>Facilitar información relativa a los alimentos:</p> <ul style="list-style-type: none"> • información nutricional por cada 100g: <ul style="list-style-type: none"> → Calorías; → Grasa saturada; → Proteínas; → Carbohidratos.
Planificar comidas	Planificar las comidas del usuario teniendo en cuenta tanto las necesidades diarias de nutrientes como las preferencias alimentarias, garantizando así un equilibrio adecuado de

	nutrientes en las recetas e ingredientes seleccionados.
Controlar las calorías y nutrientes ingeridos por el usuario	El control de las calorías y nutrientes puede ser realizado diariamente. Este control servirá de apoyo para planificar las comidas.

Tabla 2 – Casos de uso

3.2 Elección de los datasets

Una vez definido el objetivo y los casos de uso del chatbot, es esencial recopilar los datos necesarios y relevantes. Los chatbots requieren un proceso de entrenamiento riguroso para que sean capaces de entender y contestar de manera concisa y efectiva a las consultas del usuario, por lo que es fundamental que los datos sean precisos y de alta calidad.

En este apartado, se presentará información específica sobre los datos que se utilizarán como base de conocimiento para el chatbot culinario, incluyendo recetas, ingredientes, nutrientes y alergias relevantes. En el apartado de definición de intenciones, se profundizará en el proceso de entrenamiento de las preguntas y respuestas, que se complementarán con los conjuntos de datos mencionados en este apartado.

Se han evaluado otros datasets de recetas para el proyecto, sin embargo, algunos carecían de información relevante sobre los nutrientes, el tipo de dieta o la presencia de alergias. Además, algunos de estos datasets resultaron ser demasiado pequeños en términos de volumen de datos.

A continuación, se enumeran y se detallan los diferentes conjuntos de datos seleccionados para el proyecto:

- Food.com Recipes and Interactions [10] – recipes.csv
Este conjunto de datos, compuesto por 231637 filas, incluye información sobre las recetas, como los ingredientes, las instrucciones de preparación, las etiquetas (postre, lunch, vegetariano, ...) y los valores nutricionales.
- Food.com Recipes with Search Terms and Tags [11] – recipesTerms.csv
Este conjunto complementa el anterior, con información adicional sobre las recetas, como las porciones, la cantidad de los ingredientes y los términos de búsqueda.

Se combinan estos conjuntos de datos, ya que cada uno aporta características diferentes de cada receta que serán utilizadas por el chatbot para realizar

recomendaciones. Para referirse a este conjunto de datos, se utilizará el nombre “**recipes**”. Dado el tamaño considerable del conjunto de datos, se ha optado por limitar a las primeras 10000 filas.

- Nutrientes Análisis exploratorio (EDA) [12] – nutritionIngredients.csv
Este conjunto, compuesto por 8790 filas, contiene información sobre los nutrientes de una variedad de alimentos y será utilizado por el chatbot para proporcionar información nutricional a los usuarios. Para referirse a este conjunto de datos, se utilizará el nombre “**nutrition_ingredients**”.
- Food: Allergens and Allergies [13] – intolerances.csv
El conjunto de datos, compuesto por 184 filas, contiene información sobre los alérgenos presentes en diferentes alimentos. A partir de esta información, se podrá clasificar las alergias o intolerancias presentes en las recetas y proporcionar así recomendaciones seguras a los usuarios que padecen alguna alergia alimentaria. Para referirse a este conjunto de datos, se utilizará el nombre “**intolerances**”.

Posteriormente, se realizará una limpieza de los datos, eliminando aquellas columnas que no son relevantes para el dominio del chatbot y verificando que no existan filas duplicadas.

La columna *tag* de *recipes* será utilizada para clasificar las recetas en vegetarianas y/o veganas en función de las etiquetas. Los ingredientes de las recetas, en conjunto con el dataset *intolerances*, permitirá clasificar las alergias e intolerancias presentes en las recetas, revisando todos los ingredientes de las mismas.

3.3 Modelo de clasificación multi-etiqueta

Una vez que los conjuntos de datos hayan sido sometidos a procesos de limpieza y preprocesamiento, se procede al desarrollo de los modelos de clasificación multi-etiqueta. Estos modelos tienen como objetivo predecir las alergias y el tipo de dieta presentes en las recetas.

Para implementar estos modelos, se siguen varios pasos. En primer lugar, se extraen las etiquetas de alergias presentes en el conjunto de datos *recipes*. Utilizando las librerías *NLTK* [14] y *matplotlib* [15], se obtiene un gráfico de frecuencias que nos permite visualizar cuales son las alergias presentes en mayor cantidad en el conjunto de datos.

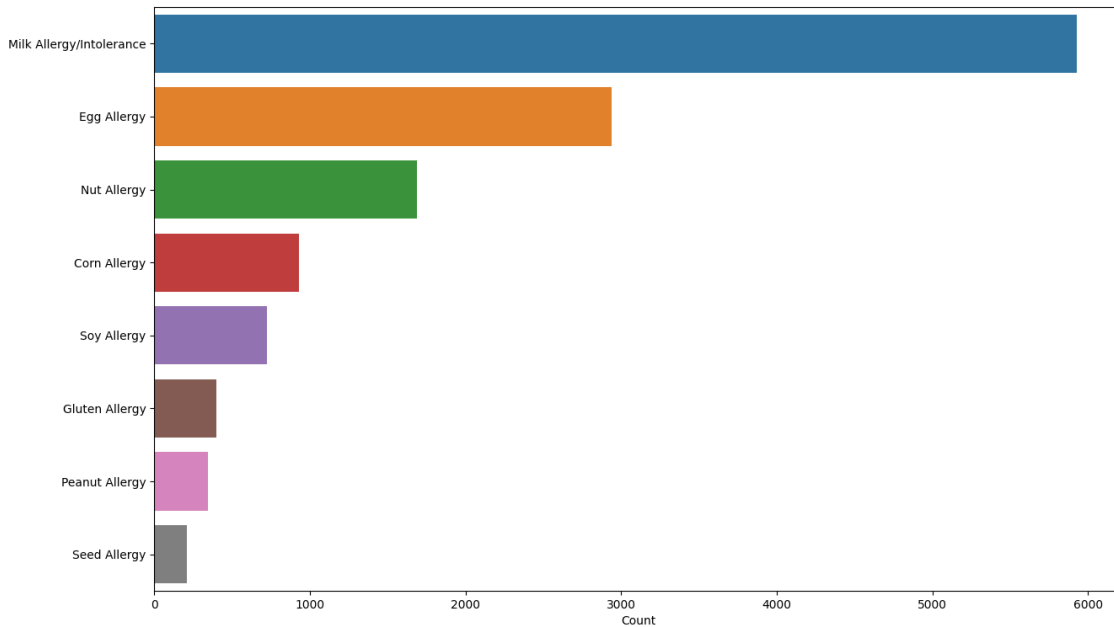


Figura 4 – Diagrama de frecuencia de alergias

El diagrama fue generado utilizando el conjunto de datos *recipes*, que consta de 10000 recetas. Se puede observar claramente que en el conjunto de datos hay una gran cantidad de recetas que contienen productos lácteos o sus derivados en comparación con otras categorías de alimentos. Estos datos son comprensibles dado que ingredientes lácteos como mantequilla, yogur, queso y leche son comunes en muchas recetas. Sin embargo, también se dispone de una cantidad razonable de recetas aptas para personas con alergias o intolerancias a los productos lácteos.

A continuación, se procede a crear el modelo de clasificación multi-etiqueta utilizando la librería *Scikit-Learn* [16]. Primero, se convierte el texto de la columna *allergies* y *diets* del dataset *recipes* en *features* y se transforman las etiquetas en un formato adecuado. Posteriormente, se divide el conjunto de datos en dos conjuntos, uno de entrenamiento y uno de validación. Para este caso, se opta por utilizar el algoritmo de Regresión Logística (*LogisticRegression* [17]) que permite estimar la probabilidad de que un ingrediente se encuentre en una categoría, es decir, este asociado a una alergia o dieta. Para el entrenamiento y predicción de las etiquetas de alergias y dietas en las recetas, se emplea la clase *OneVsRestClassifier* [18], que convierte un problema de clasificación multi-etiqueta en un conjunto de problemas de clasificación binaria para cada etiqueta (alergia/dieta)

Después de entrenar el modelo de clasificación, se ha evaluado el modelo utilizando las métricas de *F1* y precisión (*accuracy*). Con el conjunto de datos *recipes*, se ha obtenido una precisión del 89% en la predicción de alergias y un 81% en la predicción de dietas.

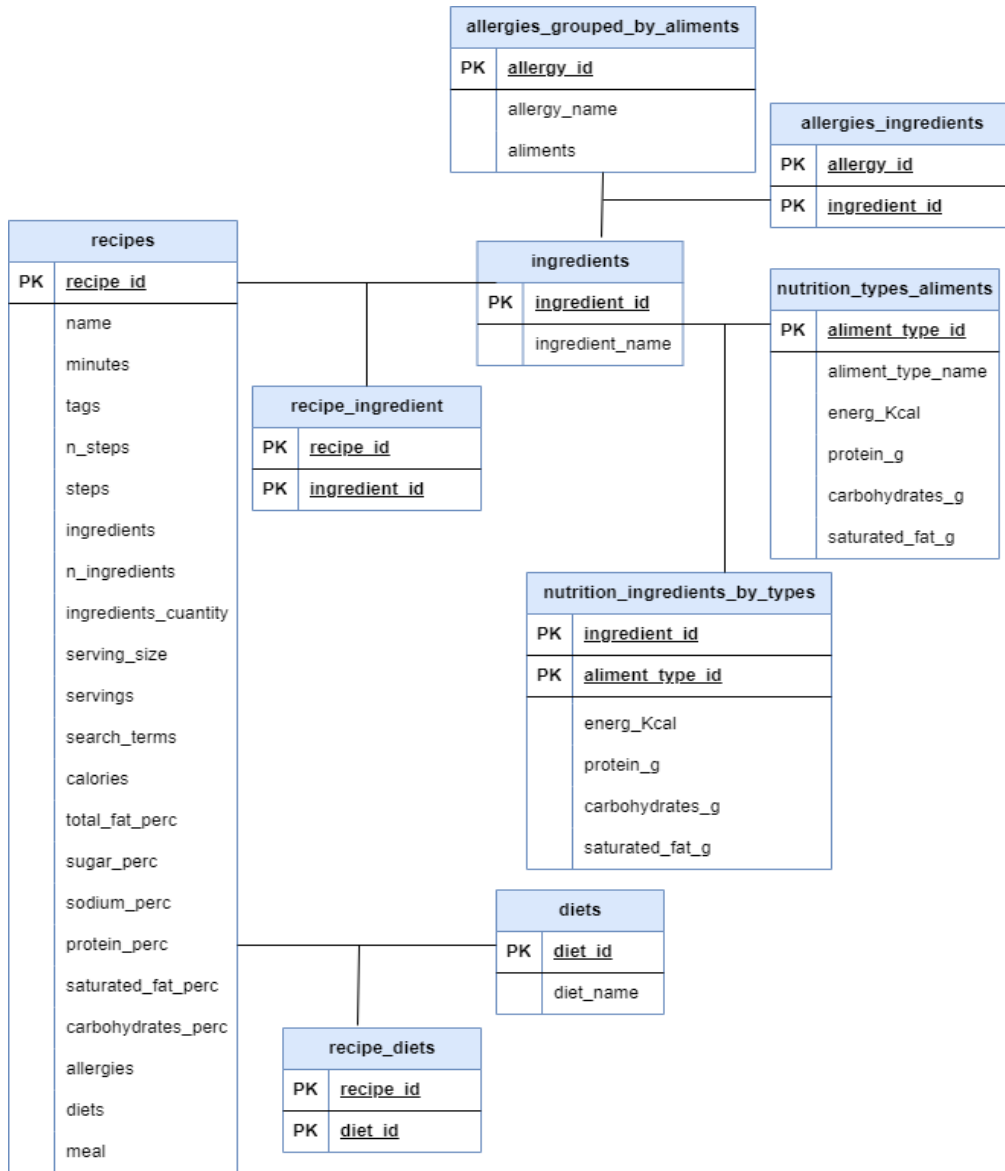
Se ha podido comprobar que, probando con un conjunto de datos mayor, exactamente, con las primeras 90000 recetas, se ha obtenido una mayor

precisión. Se obtuvo una precisión del 99% en la predicción de alergias y un 86% en la predicción de dietas.

3.4 Modelo de la base de datos

Para almacenar toda la información relativa a las recetas, ingredientes, nutrientes y usuarios, es necesario crear una base de datos para poder consultar y manipular estos datos. Por eso, se ha optado por utilizar SQLite [19], ya que ofrece una gran facilidad de manipulación para actualizar y borrar registros. Además, la base de datos puede ser ejecutada en cualquier plataforma sin requerir configuraciones [20]. Se ha implementado en el archivo *culinary_chatbot_database.py* métodos para añadir, consultar y eliminar tablas. Además, se han implementado métodos que permiten la inserción, actualización y eliminación de filas en las tablas de la base de datos. Adicionalmente, se han definido funciones que permiten obtener la información de las tablas de manera más eficiente.

A continuación, se presenta el modelo de la base de datos CulinaryDB:



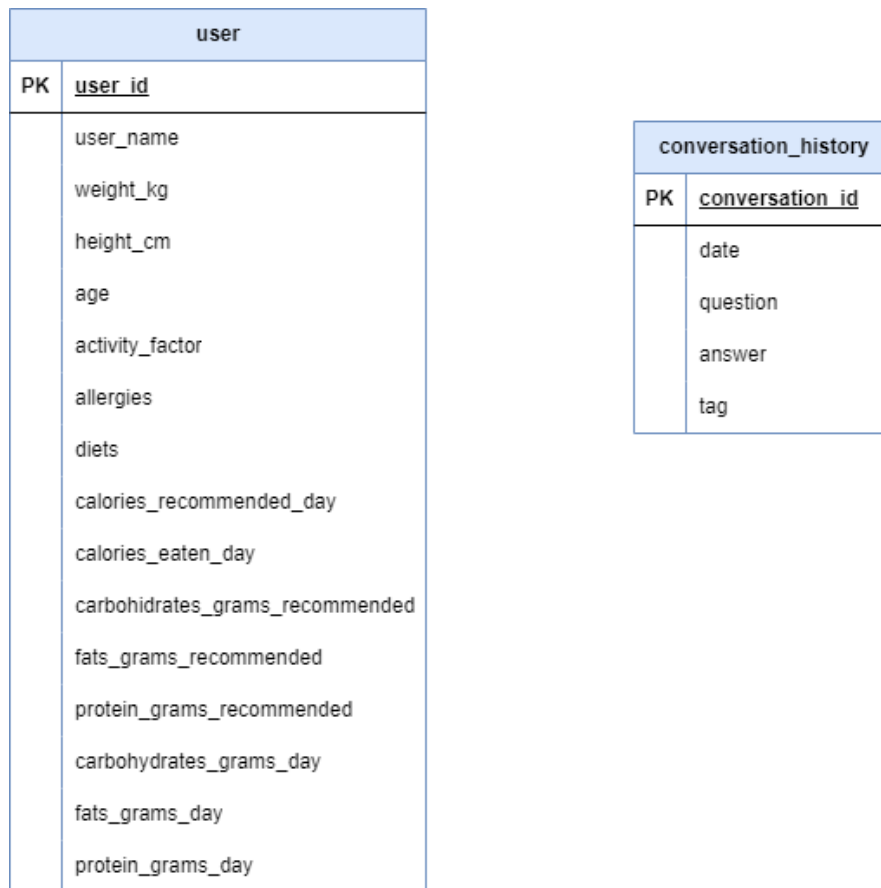


Figura 5 – Modelo de la base de datos

A continuación, se resume la información guardada en cada tabla:

- **recipes:** Contiene la información básica de cada receta, como el nombre, los ingredientes con sus respectivas cantidades, los pasos de preparación, las calorías, y los porcentajes de aporte de cada nutriente por porción de la receta.
- **ingredients:** Almacena todos los ingredientes utilizados en las recetas.
- **recipe_ingredient:** Establece la relación entre las tablas *recipes* e *ingredients*.
- **nutrition_types_aliments:** Contiene diversos alimentos y sus valores nutricionales en diferentes estados de preparación (cocido, crudo, frito) o partes específicas (por ejemplo, yema o clara del huevo) o tipos (por ejemplo, huevo de gallina o huevo de codorniz).
- **nutrition_ingredients_by_types:** Establece la relación entre las tablas *ingredients* y *nutrition_types_aliments*, asociando los ingredientes con los diferentes estados o partes específicas y sus respectivos valores nutricionales.
- **allergies_grouped_by_aliments:** Almacena las diferentes alergias y los alimentos que las pueden provocar.
- **allergies_ingredients:** Establece la relación entre las tablas *ingredients* y *allergies_grouped_by_aliments*, identificando los ingredientes que pueden causar alguna alergia.

- **diets:** Contiene los distintos tipos de dietas que, en este caso, serán vegetariana y vegana.
- **recipe_diets:** Establece la relación entre las tablas *recipes* y *diets*.
- **conversation_history:** Almacena las conversaciones realizadas entre el usuario y el chatbot, incluyendo las etiquetas predichas por el modelo y la fecha de la conversación.
- **user:** Almacena información relevante del usuario como el peso, altura, factor de actividad, edad, preferencias dietéticas y alergias. Se registran las recomendaciones diarias de consumo de nutrientes en gramos y de calorías siguiendo las recomendaciones de la OMS. Adicionalmente, se realiza un seguimiento de las calorías ingeridas y las pendientes de consumir diariamente, actualizando las variables correspondientes.

Analizando el esquema anterior, se pueden observar las relaciones entre las diferentes tablas de la base de datos *CulinaryDB.db*. Se destaca lo siguiente:

- Entre la tabla *recipes* y *ingredients* se establece una relación de muchos a muchos, ya que una receta puede contener muchos ingredientes y un ingrediente puede estar presente en varias recetas.
- La relación entre la tabla *nutrition_types_aliments* y *ingredients* muestra una relación de uno a muchos. Un ingrediente puede tener varios tipos o presentarse en diferentes estados (cocido, crudo, frito), pero un ingrediente en un estado específico (por ejemplo, huevo cocido) solo tendrá una correspondencia en la tabla *ingredients* (huevo).
- Entre la tabla *ingredients* y *allergies_grouped_by_aliments* se percibe una relación de uno a muchos. Un ingrediente solo estará asociado a un tipo de alergia, pero una alergia puede tener asociados uno o más alimentos. Por ejemplo, el ingrediente “mantequilla” estará asociado únicamente a la alergia “lácteos”, mientras que la alergia a los lácteos puede estar asociada a varios alimentos como leche, queso, mantequilla, entre otros.
- Entre *recipes* y *diets*, se verifica una relación de muchos a muchos, ya que una receta puede estar asociada a diferentes tipos de dietas, y una dieta puede estar presente en varias recetas.

3.5 Cálculo de calorías y nutrientes

En el archivo “*nutrition_functions.py*”, se han creado funciones para calcular las calorías y nutrientes. Estos cálculos son necesarios para que el usuario pueda mantener una dieta equilibrada y saludable. En la primera interacción entre el chatbot y el usuario, se solicitan algunos datos, como peso, altura, factor de actividad y edad. El factor de actividad [42], indica la actividad física del usuario:

- 1,2 si hace muy poco o ningún de ejercicio.
- 1,375 si hace ejercicio 1-3 días a la semana.
- 1,55 si hace ejercicio 3-5 días a la semana.

- 1,725 si hace ejercicio 6-7 días a la semana.
- 1.9 si hace entrenamientos muy duros, dos veces al día.

Con estos datos, se calcula la cantidad de calorías recomendadas diariamente con la fórmula de Harris Benedict [42]:

Para individuos de género femenino, la fórmula es la siguiente:

$$\text{TMB} = (10 \times \text{peso en kg}) + (6,25 \times \text{altura en cm}) - (5 \times \text{edad en años}) - 161$$

$$\text{Calorías diarias necesarias} = \text{TMB} \times \text{factor de actividad}$$

Para individuos de género masculino, la fórmula es la siguiente:

$$\text{TMB} = (10 \times \text{peso en kg}) + (6,25 \times \text{altura en cm}) - (5 \times \text{edad en años}) + 5$$

$$\text{Calorías diarias necesarias} = \text{TMB} \times \text{factor de actividad}$$

TMB* Tasa metabólica basal

Según la OMS [43], se recomienda los siguientes porcentajes de ingesta diaria de nutrientes:

- 50-55% de carbohidratos:
 - carbohidratos_min = 0.50
 - carbohidratos_max = 0.55
- 30-35% de grasas:
 - grasas_min = 0.50
 - grasas_max = 0.55
- 12-15% de proteínas:
 - proteínas_min = 0.50
 - proteínas_max = 0.55
- 1 gramo de carbohidratos aporta 4 kcal → carbohidratos_kcal = 4
- 1 gramo de grasas aporta 9 kcal → grasas_kcal = 9
- 1 gramo de proteínas aporta 4 kcal → proteínas_kcal = 4

A partir de estos valores, se calculan las cantidades de carbohidratos, grasas y proteínas recomendadas consumir diariamente en gramos:

Carbohidratos:

$$\text{Kcal} = \text{calorias_usuario} \times \text{carbohidratos_min}$$

$$\text{carbohidratos_gramos_recomendados} = \text{kcal} / \text{carbohidratos_kcal}$$

Grasas:

$$\text{Kcal} = \text{calorias_usuario} \times \text{grasas_min}$$

$$\text{grasas_gramos_recomendados} = \text{kcal} / \text{grasas_kcal}$$

Carbohidratos:

$Kcal = calorías_usuario * proteínas_min$

$proteínas_gramos_recomendados = kcal / proteínas_kcal$

Cada vez que el usuario solicite una receta o un plan de comidas al chatbot, se asume que el usuario tiene la intención de consumirlos. Esto implica que las calorías de la receta se sumarán a la columna *calories_eaten_day*, lo que permite hacer un seguimiento de las calorías consumidas a lo largo del día y evitar exceder las recomendaciones. Además, se realizará un seguimiento de las cantidades de nutrientes en gramos que el usuario va consumiendo a lo largo del día en las columnas *carbohidrates_grams_day*, *fats_grams_day* y *protein_grams_day*. Para ello, es necesario obtener la cantidad en gramos de nutrientes que aporta la receta y ajustar esa cantidad a las calorías recomendadas del usuario.

Con el seguimiento de las calorías y la cantidad de nutrientes que el usuario va consumiendo a lo largo del día, el chatbot puede recomendar recetas de acuerdo con las cantidades de calorías y nutrientes que faltan por consumir. Sin embargo, es importante destacar que una alimentación saludable va más allá de estos nutrientes específicos. Por lo tanto, cada vez que el usuario solicite una receta o un plan de comidas, también se resaltarán la importancia de consumir 5 raciones diarias de verduras, hortalizas y frutas. Estos alimentos son fundamentales para obtener fibra, minerales y vitaminas necesarios para mantener una dieta equilibrada.

3.6 Flujo de conversión del chatbot

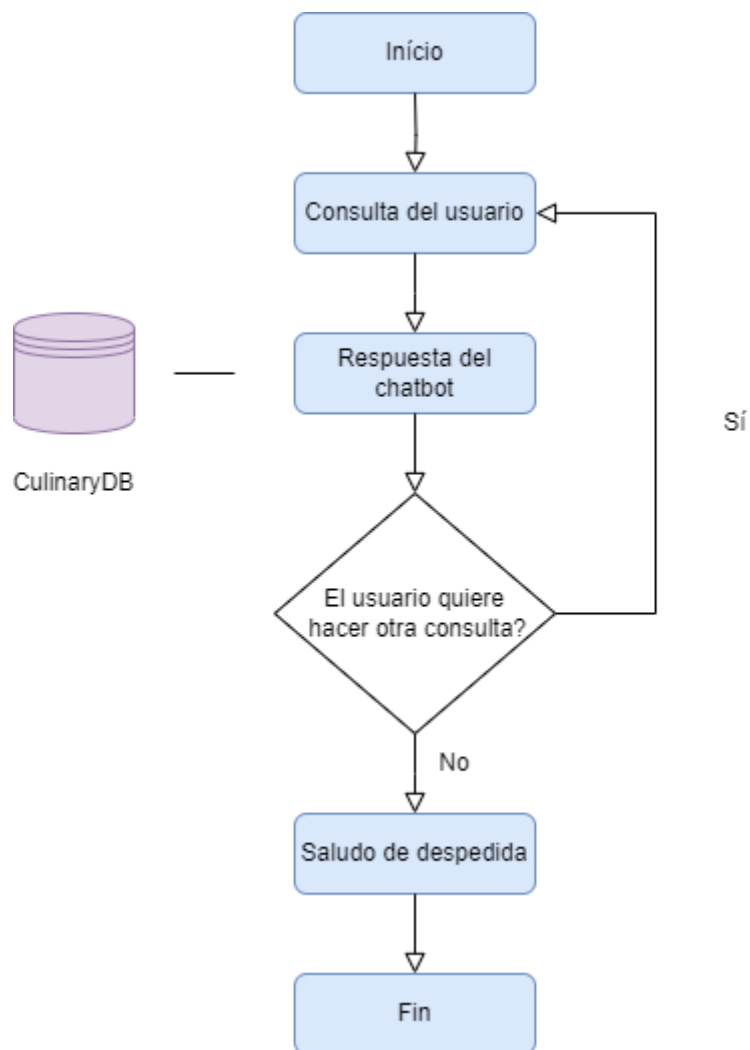


Figura 6 – Flujo de conversación del chatbot

El lenguaje utilizado por el chatbot debe ser sencillo y amigable para que la experiencia del usuario sea agradable. El saludo, al ser la primera impresión que el usuario captará, debe transmitir la sensación de bienvenida y establecer una conexión con el usuario. El chatbot puede preguntar, incluso, el nombre del usuario, para poder establecer una conexión más estrecha con el usuario. En la primera interacción con el usuario, el chatbot solicitará información relevante que permitirá al chatbot recomendar recetas o planes de comidas personalizados, teniendo en cuenta las preferencias del usuario. Además, se utilizará la información del usuario para calcular las calorías y la cantidad de nutrientes en gramos recomendadas diariamente.

Las posibles consultas y respuestas del usuario están relacionadas con los casos de uso mencionados en el apartado anterior. El chatbot debe ser capaz

de demostrar sus conocimientos de manera apropiada y eficiente, utilizando un lenguaje claro y conciso.

Al terminar las consultas del usuario, el chatbot debe despedirse de él con un cordial saludo.

A pesar de la relación cercana que se busca establecer entre el chatbot y el usuario, es importante dejar en claro que el chatbot es una máquina y no un ser humano. De esta manera, en caso de que el chatbot no sea capaz de comprender o contestar a alguna de sus consultas, el usuario será más comprensible.

3.7 Definición de las intenciones: etiquetas, patrones y respuestas

Durante la etapa de diseño, se definen las posibles intenciones del usuario mediante la creación de etiquetas, con los correspondientes patrones y respuestas. Para ello, se ha utilizado como punto de partida un ejemplo existente de intenciones [22], y se han realizado modificaciones para adaptarlo al caso específico del proyecto. Algunas intenciones innecesarias han sido eliminadas, y se han añadido nuevas intenciones relacionadas con las necesidades del usuario y propósitos de este chatbot culinario.

En la sección de anexos de la memoria (Anexo 1), se podrá observar una pequeña parte de la estructura del archivo "intents.json". Este archivo, escrito en formato JSON[23], contiene las intenciones definidas, que se componen de los siguientes elementos:

- Etiqueta (*tag*): La etiqueta permite identificar las intenciones del usuario.
- Patrones (*patterns*): Los patrones representan las frases o preguntas que el usuario puede realizar.
- Respuestas (*responses*): Contiene las posibles respuestas que el chatbot dará al usuario. Estas respuestas se complementarán posteriormente con información de la base de datos, según la pregunta del usuario.

A continuación, se presentan las etiquetas definidas:

- ***greeting***: Indica la intención del usuario de saludar al chatbot.
- ***goodbye***: Representa la intención del usuario de despedirse y finalizar la interacción con el chatbot.
- ***thanks***: Indica la intención del usuario de agradecer.
- ***options***: Indica que el usuario busca información sobre las funcionalidades del chatbot.
- ***recipe***: Representa la intención del usuario de solicitar una receta.
- ***recipe_ingredient***: Representa la intención del usuario de solicitar una receta que contenga uno o varios ingredientes específicos.
- ***recipe_calories***: Indica la intención del usuario de querer conocer las calorías de una receta.

- ***recipe_nutrition_values***: Representa la intención del usuario de querer conocer los valores nutricionales de una receta.
- ***aliment_nutrition_values***: Representa la intención del usuario de querer conocer los valores nutricionales de un determinado alimento.
- ***plan_meal_day***: Representa la intención del usuario de solicitar un plan de comidas para el día.
- ***user_calories_left***: Indica la intención del usuario de querer saber cuántas calorías le quedan por consumir en el día.
- ***user_calories_consumed***: Representa la intención del usuario de querer saber cuántas calorías ha consumido en el día.
- ***user_calories***: Representa la intención del usuario de querer saber cuántas calorías le son recomendadas consumir diariamente según la OMS.

Las etiquetas y los patrones definidos desempeñarán un papel importante en la siguiente fase, donde se desarrollará la red neuronal utilizando el modelo secuencial de Keras [24]. El objetivo será capacitar el modelo para encontrar “el patrón entre los patrones y las etiquetas, de manera que cuando el usuario realice una pregunta, el chatbot pueda identificar la etiqueta apropiada y proporcionar una de las respuestas definidas como respuesta” [25].

4. Desarrollo e implementación

4.1 Desarrollo del modelo

El desarrollo del modelo que se mencionará y explicará en este apartado, se encuentra en el archivo “culinary_chatbot_model.py”. Para su construcción, se ha elegido el lenguaje de programación Python, ya que es el lenguaje “preferido para proyectos de datos, proyectos de aprendizaje automático y proyectos de chatbots” [26]. Esto se debe a su sencillez y capacidad para trabajar con grandes volúmenes de datos. Además, “Python también tiene una amplia selección de bibliotecas para aprendizaje automático y procesamiento de lenguaje natural (NLP), incluido el poderoso Natural Language Toolkit (*NLTK*), que muchos desarrolladores consideran la mejor biblioteca de NLP.” [26]

Para desarrollar el modelo, se han seguido las siguientes fases:

1. Importar y cargar el archivo de datos
2. Preprocesamiento de los datos
3. Crear datos de entrenamiento y de prueba
4. Construir el modelo

Importar librerías y cargar el archivo de datos

Existían varias librerías por elegir, como Chatterbot [27] o Rasa [28], sin embargo, se ha tomado la decisión de utilizar las librerías *Keras* [24] y *NLTK* [14] debido a su simplicidad. “La librería *Keras* proporciona un acceso de alto nivel mediante una API a las funcionalidades de la librería *tensorflow*, usada para la creación a bajo nivel de modelos de *Deep learning*” [30]. Al ser una API “intuitiva, modular y fácil de ampliar” [31], es posible construir y entrenar redes neuronales fácilmente. La librería *NLTK* [14], también conocida como Natural Language Toolkit, es una librería diseñada para trabajar con el lenguaje humano. Con esta librería, es posible realizar tareas de clasificación, *stemming*, lematización, análisis sintáctico, tokenización, razonamiento semántico y la exclusión de *stop words* que facilitan el procesamiento y análisis del texto.

Aparte de estas librerías, necesitaremos otras para trabajar con los datos y los archivos como:

- NumPy [32]: Se utiliza para trabajar con vectores y matrices. Además, ofrece varias funciones matemáticas para trabajar con este tipo de datos.
- json [35]: Se utiliza como codificador y decodificador JSON [23].
- pickle [33]: Se utiliza para la “serialización de objetos Python” [33].
- random [34]: Se utiliza para “generar números pseudoaleatorios” [34].

Para iniciar la construcción de la red neuronal, se carga el archivo “intents.json”, con el apoyo de la librería json, que contiene las intenciones del usuario.

Preprocesamiento de los datos

Antes de empezar a trabajar con los datos de texto, es necesario realizar un preprocesamiento de los mismos. En primer lugar, se realiza la tokenización de los patrones definidos en “intents.json”. La tokenización consiste en dividir las sentencias en partes más pequeñas conocidas como *tokens*. Se utiliza la función *nlk.word_tokenize()*, que divide el texto en palabras individuales. Durante este proceso, se van registrando todas las palabras/*tokens* en una lista “words”. Además, se crean otras dos listas. La primera, conocida como “classes”, para almacenar todas las clases (etiquetas) identificadas en el archivo “intents.json”. La segunda lista “documents”, para guardar la asociación entre la lista de palabras de cada patrón con la etiqueta correspondiente. Por ejemplo, un elemento de esta lista podría tener la siguiente forma:

```
(['How', 'are', 'you'], 'greeting')], donde 'How', 'are', 'you' son los tokens resultantes de realizar la tokenización del patrón “How are you” y 'greeting' es la etiqueta asociada a ese patrón.
```

En segundo lugar, y para finalizar el proceso de preprocesamiento de los datos, se utiliza la instancia de *WordNetLemmatizer()* [36] previamente creada para aplicar la lematización a cada palabra. Se elige la lematización en lugar del stemming debido a que en este caso el análisis del contexto y obtener una mayor precisión son más importantes que la rapidez. El stemming puede generar palabras incorrectas o con significados diferentes, ya que se basa en la eliminación de afijos, o sea, relaciona los afijos de las palabras para obtener la raíz de la palabra” [37]. La lematización, por otro lado, convierte las palabras a su forma base, conocida como lema, relacionando “una palabra flexionada o derivada con su forma canónica o lema” [37].

Se guardan las listas “words” y “classes” en archivos binarios (“words.pkl” y “classes.pkl”, respectivamente), los cuales se utilizarán para predecir las etiquetas.

Crear datos de entrenamiento y de prueba

En esta fase, se generan datos de entrenamiento y prueba para la red neuronal. La capa de entrada de la red neural recibirá como datos de entrada los patrones y la capa de salida devolverá la predicción, es decir, las etiquetas asociadas a cada patrón.

Es necesario transformar los datos en un formato numérico para usarlos en la red neuronal. Para cada patrón (transformado en tokens), se aplica la función *lemmatize()* para obtener el lema de cada palabra en el patrón. A continuación, se recorre la lista de lemas “words” y se verifica si algún lema de “words” coincide con los lemas del patrón. Si hay coincidencia, se agrega un 1 a la lista “bag_of_words” en la posición correspondiente al lema en la lista “words”; de lo contrario, se agrega un 0.

Por ejemplo, si tenemos el patrón “Calories recommended” y los lemas obtenidos con la función *lemmatize()* son ['calorie', 'recommended']. Sabiendo que el lema 'calorie' se encuentra en la posición 24 en la lista “words” y el lema 'recommended' en la posición 55, entonces se obtendrá un 1 en las posiciones 24 y 55 mientras que en las demás posiciones un 0, ya que no coinciden los lemas.

Para la etiqueta asociada al patrón, se crea una lista “output_row” con la longitud de la lista “classes”, que contiene todas las etiquetas sin repeticiones. Se coloca un 1 en la posición correspondiente a la etiqueta que coincide con la etiqueta del patrón y un 0 en las demás posiciones.

Para finalizar, se crea una nueva lista “training”, donde se asocia cada lista “bag_of_words” con su etiqueta correspondiente representada en valores binarios (output_row). Posteriormente, se dividen los datos de “training” en dos listas, “train_x” y “train_y”. La lista “train_x” contiene los patrones codificados, y “train_y”, las etiquetas.

Construir el modelo

El modelo de la red neuronal se compone de tres capas, la primera capa con 128 neuronas, la segunda con 64 y la última capa con el número de neuronas igual al número de clases/etiquetas. Para construir este modelo, se utiliza el modelo *Sequential()*, que permite definir un modelo como una secuencia de capas, “donde cada capa se añade de una en una utilizando el método *add()*.” [38] “Las capas pueden instanciarse con diferentes tipos de neuronas,” [38] que en nuestro caso será *Dense* (totalmente conectadas). En las dos primeras capas, se utiliza la función de activación “ReLU” y para la capa de salida la función de activación “Softmax”. Además, se incluyen las capas *Dropout* que ayudan a reducir el sobreajuste.

Una vez que el modelo está definido, se compila el modelo detallando la función de pérdida de entropía cruzada, la función de optimización *sgd* (Stochastic Gradient Descent) y la métrica de precisión (*accuracy*). Después, se entrena el modelo utilizando *fit()* durante 200 épocas. Con estas 200 épocas, se ha logrado alcanzar hasta una precisión del 93%.

A continuación, se muestra el código del modelo:

```
# Create a Sequential machine learning model 3 layers. First layer 128
neurons,
# second layer 64 neurons and 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile and fit the model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200,
batch_size=5, verbose=1)
```

Al final, se guarda el modelo como “chatbotmodel.h5”, que se utilizará en la implementación del chatbot para predecir las etiquetas.

4.2 Implementación del chatbot y generación de la interfaz

En el archivo “culinary_chatbot.py” se encuentra la implementación del chatbot, que se explicarán este apartado. Se incluyen todas las funciones necesarias para crear el chatbot y cargar el modelo entrenado de la fase anterior, el cual se utilizará para predecir las etiquetas analizando la entrada del usuario.

Para llevar a cabo la implementación, se han importado algunas librerías para cargar los archivos y el procesamiento de los datos. Además, se ha añadido la librería *datetime* [40] que permite trabajar con los tipos de datos *date*, *time* y *datetime*.

Igualmente, se han importado los siguientes archivos, que contienen funciones necesarias para consultar y guardar información en la base de datos, así como funciones para calcular los nutrientes y calorías recomendadas:

- **“culinary_chatbot_database.py”**: Este archivo contiene funciones básicas para interactuar con la base de datos, como añadir, consultar y eliminar tablas. Además, se han implementado métodos que permiten insertar, actualizar y borrar filas. También incluye otras funciones específicas que permiten obtener información de las tablas de manera más eficiente.
- **“nutrition_functions.py”**: Contiene todas las funciones necesarias para calcular las calorías recomendadas, las calorías consumidas y las que faltan por consumir diariamente. Además, se calculan las cantidades de nutrientes recomendadas por la OMS que el usuario deber consumir diariamente. También se incluye una función para convertir el porcentaje de valor diario presente en una receta en la cantidad de gramos de nutrientes que aporta al usuario.

Después, se han definido las funciones necesarias para la implementación del chatbot y generación de la interfaz:

- **save_persistent_data**: Esta función se encarga de actualizar las variables persistentes en el archivo “persisten_variables.json” cuando se modifican.
- **load_persistent_data**: Carga las variables almacenadas del archivo “persisten_variables.json”.

- **clean_up_sentences:** En esta función se realiza el preprocesamiento del input del usuario, realizando la tokenización y la lematización del texto.
- **bag_of_words:** Esta función genera la “bag of words” de la sentencia (input del usuario).
- **predict_class:** Realiza la predicción de la etiqueta correspondiente al input del usuario y devuelve una lista con las etiquetas ordenadas por probabilidad.
- **get_response:** Escogiendo la etiqueta predicha con mayor probabilidad, se obtiene una respuesta aleatoria de las respuestas (responses) definidas en el archivo “intents.json”.
- **send_message:** Define la interacción entre el usuario y el chatbot.
- **open_update_info_window:** Esta función crea una ventana donde el usuario puede introducir sus datos.
- **update_info:** Permite añadir la información del usuario a la tabla *user* si es la primera vez que se proporciona, y en caso contrario, actualiza la información del usuario en la tabla correspondiente.

A posteriori, se realiza la configuración (*setup*) del chatbot llamando a la clase *DataPreparation()*, que se encarga de preparar los datos de los conjuntos de datos seleccionados para el proyecto. Seguidamente, se crean las tablas necesarias y se añaden los datos preparados.

A continuación, se carga el archivo con las intenciones, “intents.json”, así como los archivos generados previamente en el archivo “culinary_chatbot_model.py”:

- **“words.pkl”:** Contiene todas las palabras únicas encontradas en los patrones definidos en “intents.json”, en su forma base y sin duplicados.
- **“classes.pkl”:** Contiene las etiquetas definidas en “intents.json”.
- **“chatbotmodel.h5”:** Este archivo contiene el modelo entrenado que se utiliza para predecir las intenciones del usuario utilizando el método *model.predict()*, que recibe como parámetro la “bag of words” generada a partir de la entrada del usuario.

Por fin, se crea la interfaz utilizando la sencilla librería Tkinter [39] de Python. Se crea la ventana principal (Figura 7), con un cuadro de texto donde el usuario puede introducir sus preguntas y un botón "Update user info" que le dirige a otra ventana adicional (Figura 8) para introducir su información. Con la función *mainloop()*, se inicia un bucle que permite que el chatbot funcione de manera continua a través de la interfaz Culinary Chatbot.

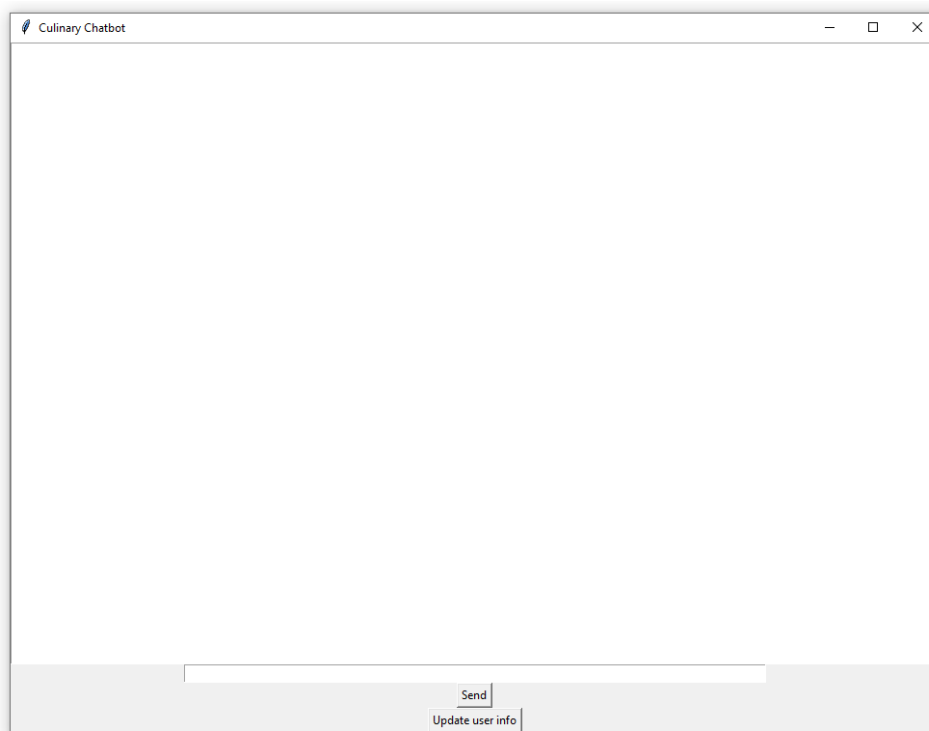


Figura 7 – Interfaz del Culinary Chatbot

En la primera interacción entre el usuario y el chatbot, se solicitarán algunos datos a través de una nueva ventana (Figura 8). Estos datos incluyen el nombre, peso, altura, edad, género, factor de actividad, alergias y preferencias dietéticas. Una vez que el usuario proporcione esta información, se calcularán las calorías recomendadas y las cantidades de nutrientes (carbohidratos, proteínas y grasas) en gramos recomendadas utilizando las funciones definidas en el archivo "nutrition_functions.py". Posteriormente, se añaden estos datos a la tabla *user*.

En la Figura 7 se puede observar la inclusión de un botón "Update user info" que permite al usuario modificar sus datos en cualquier momento. Al presionar este botón, se abrirá una nueva ventana (Figura 8) que permitirá al usuario introducir y actualizar sus datos. Después de realizar los cambios, se volverá a calcular las calorías y nutrientes recomendados y se actualizarán los datos en la base de datos para que queden reflejados los cambios.

Figura 8 – Ventana para que el usuario introduzca sus datos

Se ha creado un archivo “persistent_variables.json” que almacena algunas variables persistentes para mantener la información entre diferentes ejecuciones del programa. A continuación, se describe el propósito de cada una de estas variables:

- **"is_first_time_user_info"**: Esta variable indica si es la primera vez que el usuario introduce su información. Se utiliza para determinar si se debe mostrar la ventana (Figura 8) en primer plano para que el usuario ingrese sus datos por primera vez. Una vez que los datos son ingresados y el usuario presiona “submit”, el usuario podrá realizar preguntas al chatbot.
- **"is_first_time_running"**: Se utiliza para controlar la creación de tablas cada vez que se ejecuta el chatbot, evitando así errores en la base de datos.

- **“current_date”**: Se utiliza para identificar un nuevo día. Al ser un nuevo día, se reinician los valores nutricionales diarios del usuario, como “calories_eaten_day”, “carbohidrates_grams_day”, “fats_grams_day, protein_grams_day). Así, se garantiza que los valores se reinician diariamente y se pueda realizar un seguimiento de la cantidad de nutrientes y calorías ingeridos.

4.3 Ajustes

Se ha reducido el número de épocas de 1000 a 200 para evitar el sobreajuste. Un número demasiado alto de épocas puede llevar “al modelo de entrenamiento a aprender patrones que son específicos de los datos de muestra en gran medida. Esto hace que el modelo sea incapaz de obtener buenos resultados en un nuevo conjunto de datos. Este modelo ofrece una gran precisión en el conjunto de entrenamiento (datos de muestra), pero no consigue una buena precisión en el conjunto de prueba.” [41]

Inicialmente, se tenía la intención de utilizar las primeras 90000 recetas del dataset “recipes”. Sin embargo, debido a la gran cantidad de datos y el tiempo requerido para procesarlos, se ha tomado la decisión de reducir la muestra y utilizar solo las primeras 10000 recetas del dataset. Esta reducción permite agilizar el procesamiento y análisis de los datos.

5. Pruebas

En este apartado se realizarán pruebas para cada etiqueta o intención definida, con el objetivo de evaluar si el chatbot cumple con todas las funcionalidades implementadas y verificar si se están prediciendo correctamente las intenciones del usuario.

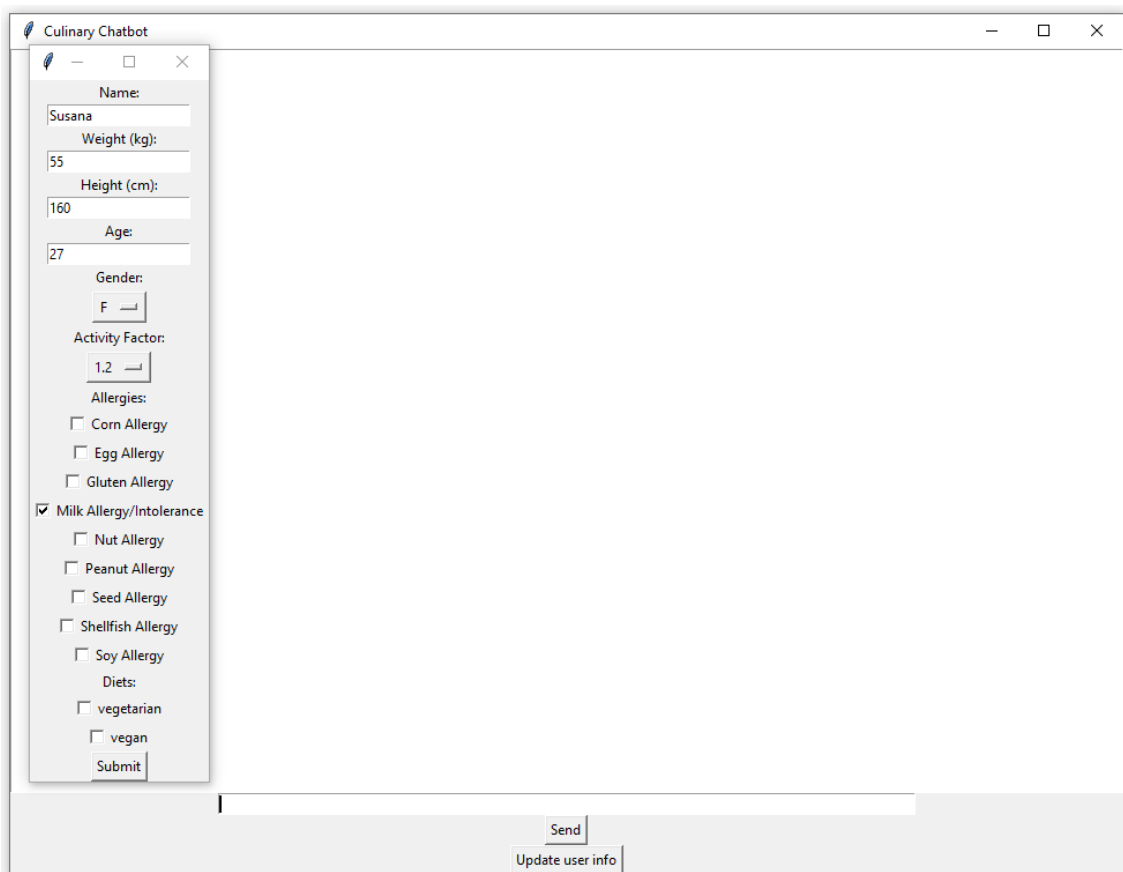
Antes de iniciar las pruebas, es importante tener en cuenta algunas notas informativas para aclarar el funcionamiento del chatbot:

- En el cálculo de los nutrientes, se consideran únicamente los carbohidratos, proteínas y grasas, ya que solo conocemos los porcentajes recomendados por la OMS de estos nutrientes. Sin embargo, se informará al usuario de la necesidad de consumir 5 raciones diarias de verduras, hortalizas y frutas para conseguir obtener fibra, minerales y vitaminas necesarias.
- Actualmente, al recomendar una receta o planificar las comidas para un día completo, no se considera que el usuario rechace la receta o el plan.

Se asume que el usuario consumirá esa receta o seguirá ese plan de comidas.

- En relación a las etiquetas “recipe_calories” y “recipe_nutrition_values”, se considerará que hacen referencia a la receta presente en la última respuesta del chatbot (ultima respuesta guardada en tabla conversation_history con la etiqueta “recipe”).
- Para poder comprobar que ha pasado un día, se puede modificar en el fichero “persistent_variables.json” la fecha de “current_date”, ya que cambiar la fecha en el reloj del sistema solo se aplica al detener y comenzar de nuevo la ejecución del programa “culinary_chatbot.py”.

El usuario que participará en la prueba proporcionará la siguiente información:



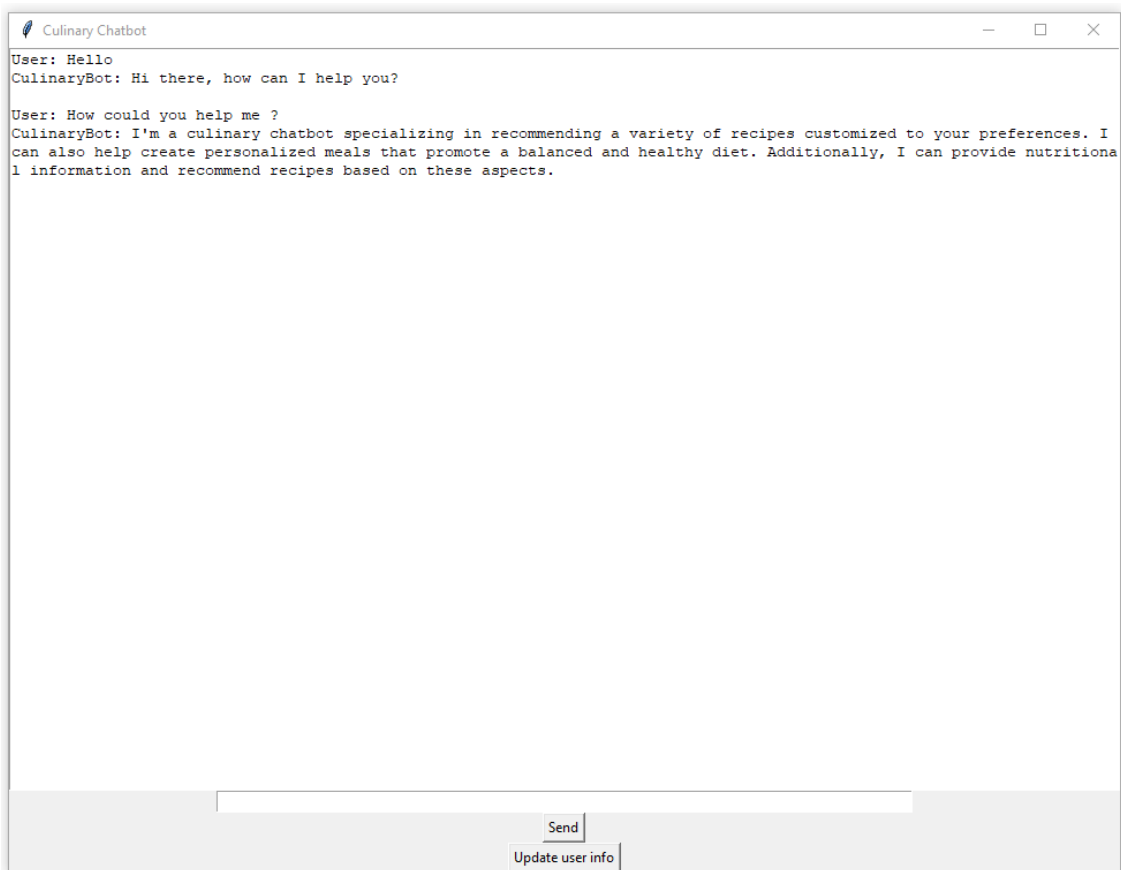
The image shows a web form titled "Culinary Chatbot" for user registration. The form is organized into several sections:

- Name:** Text input field containing "Susana".
- Weight (kg):** Text input field containing "55".
- Height (cm):** Text input field containing "160".
- Age:** Text input field containing "27".
- Gender:** Dropdown menu with "F" selected.
- Activity Factor:** Text input field containing "1.2".
- Allergies:** A list of checkboxes for various allergies:
 - Corn Allergy
 - Egg Allergy
 - Gluten Allergy
 - Milk Allergy/Intolerance
 - Nut Allergy
 - Peanut Allergy
 - Seed Allergy
 - Shellfish Allergy
 - Soy Allergy
- Diets:** A list of checkboxes for dietary preferences:
 - vegetarian
 - vegan

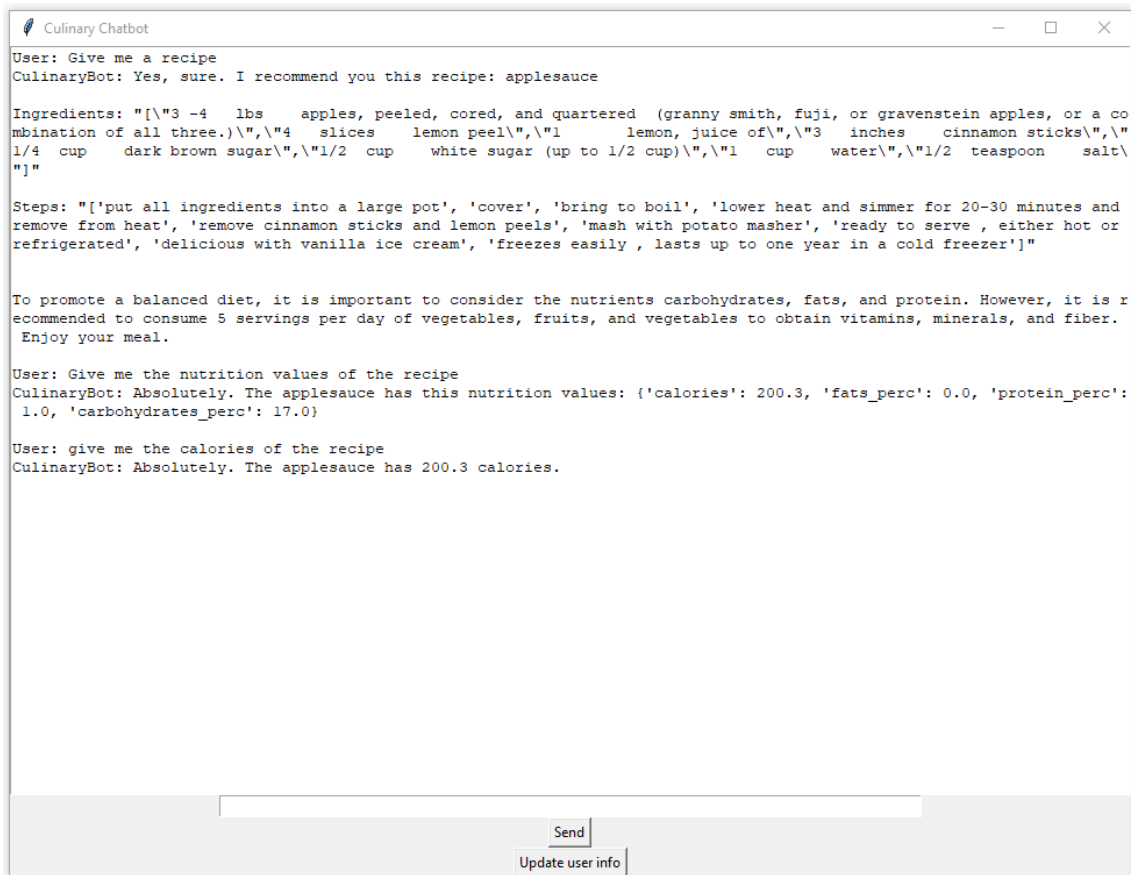
At the bottom of the form is a "Submit" button. Below the form area, there are two buttons: "Send" and "Update user info".

Cuando la usuaria presiona “Submit”, se cambia el estado de la variable “is_first_time_user_info” a “false”.

Prueba 1: Etiqueta “greeting” y “options”

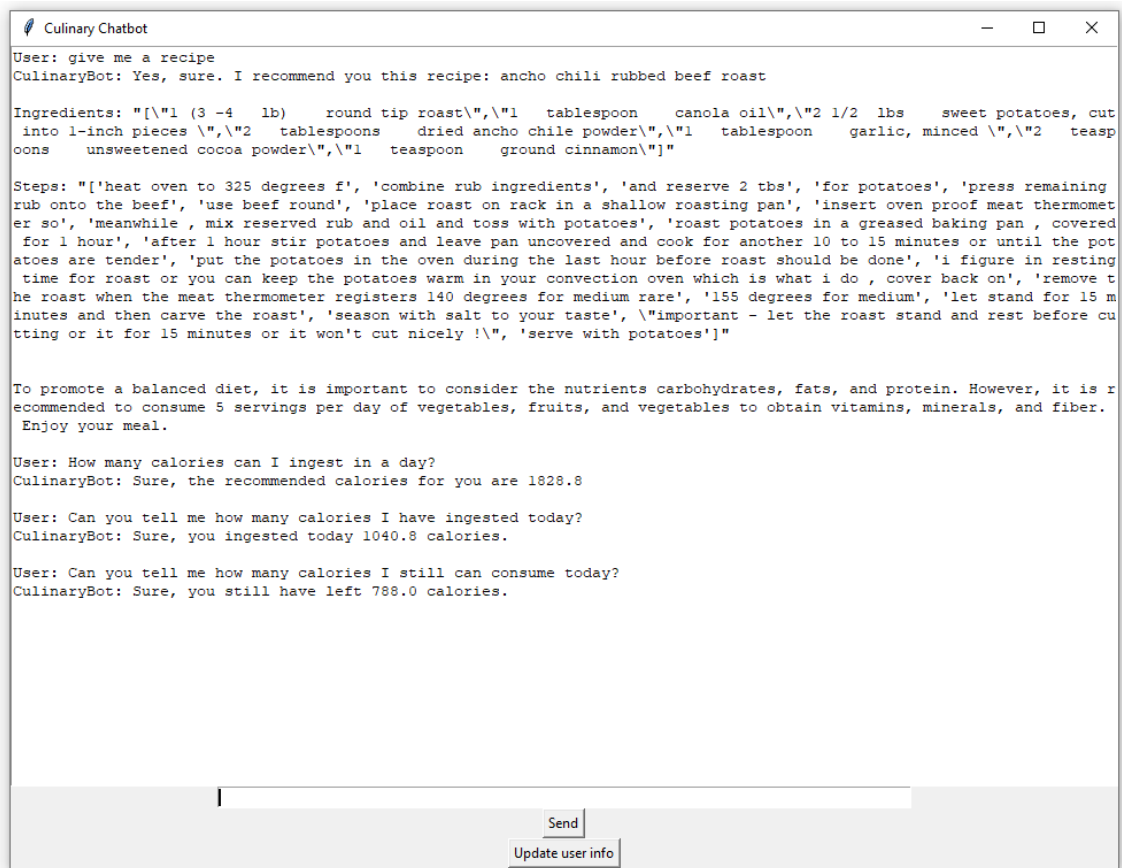


Prueba 2: Etiqueta "recipe", etiqueta "recipe_calories" y "recipe_nutrition_values"



En esta prueba, el usuario solicita al chatbot una receta y este le devuelve una receta teniendo en cuenta su alergia a los lácteos, así como las calorías y nutrientes que ha consumido durante el día. Después de recibir la receta, el usuario le solicita información sobre los nutrientes y las calorías de la receta.

Prueba 3: Etiqueta “user_calories”, “user_calories_consumed”, y “user_calories_left”



En esta prueba, el usuario le pide al chatbot que le proporcione información sobre las calorías recomendadas consumir diariamente. Además, el usuario pregunta al chatbot sobre las calorías que ha ingerido hasta el momento en ese día y cuántas le faltan por consumir para cumplir con su objetivo diario.

Prueba 4: Etiqueta “plan_meal_day”

```
Culinary Chatbot
User: I want a meal plan for today please
CulinaryBot: Here is a meal plan for today: breakfast: 5 a day smoothie

Ingredients:
"[\"1 seedless orange\", \"1 banana\", \"1 sweet potato\", \"1 carrot, chopped \", \"1 cup orange juice\"]"

Steps:
"[\"microwave sweet potato until soft\", \"scoop out pulp and put in blender\", \"peel orange and add to blender\", \"add banana, chopped carrot, and orange juice\", \"blend until smooth and creamy\", \"add ice cubes and water until desired thickness is achieved\", \"pour into glass and enjoy !\"]"

breakfast: almond peach smoothie

Ingredients:
"[\"1 cup skim milk\", \"1 cup sliced peach\", \"1/2 teaspoon almond extract\", \"1 cup non-fat vanilla frozen yogurt\"]"

Steps:
"[\"combine all ingredients in blender and process until smooth and creamy\", \"this makes a great breakfast\"]"

lunch: asparagus with thyme

Ingredients:
"[\"1 clove garlic, halved \", \"1 1/2 lbs asparagus spears\", \"2 teaspoons olive oil\", \"1/4 teaspoon dried thyme\", \"salt and pepper\"]"

Steps:
"[\"preheat oven to 400 degrees\", \"rub the cut sides of the garlic over a 13 x 9 inch baking dish, place garlic in dish\", \"snap off tough ends of asparagus\", \"add asparagus to dish, drizzle with oil\", \"sprinkle asparagus with thyme and salt and fresh ground pepper to taste\", \"toss gently and bake at 400 degrees for 20 minutes, stirring once\"]"

dessert: ancient honey and sesame fritters arxaies tiganites me meli k

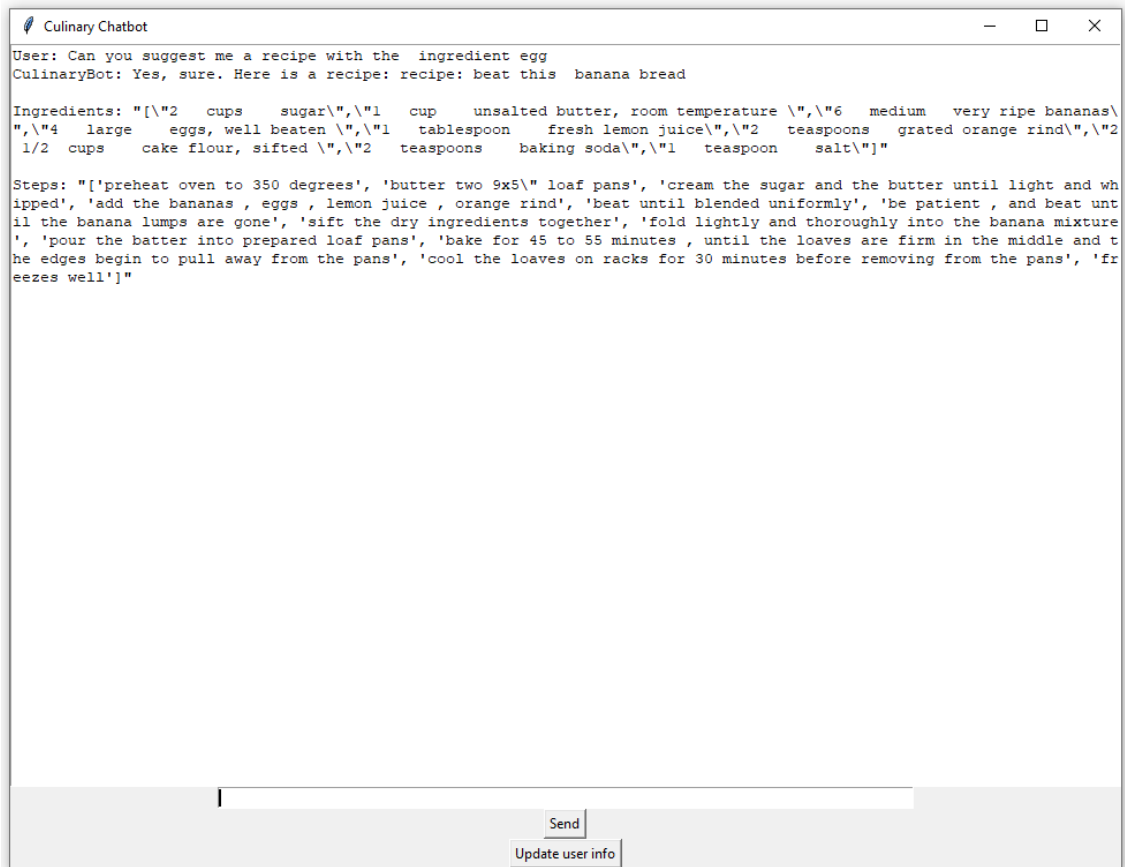
Ingredients:
"[\"120 g flour\", \"225 ml water\", \"2 tablespoons honey\", \"oil (for frying)\", \"15 g toasted sesame seeds\"]"

Steps:
"[\"start heating the oil\", \"15\", \"mix the flour, the water and one spoon of honey and make a dough\", \"heat 2 spoons oil\"]"

Send
Update user info
```

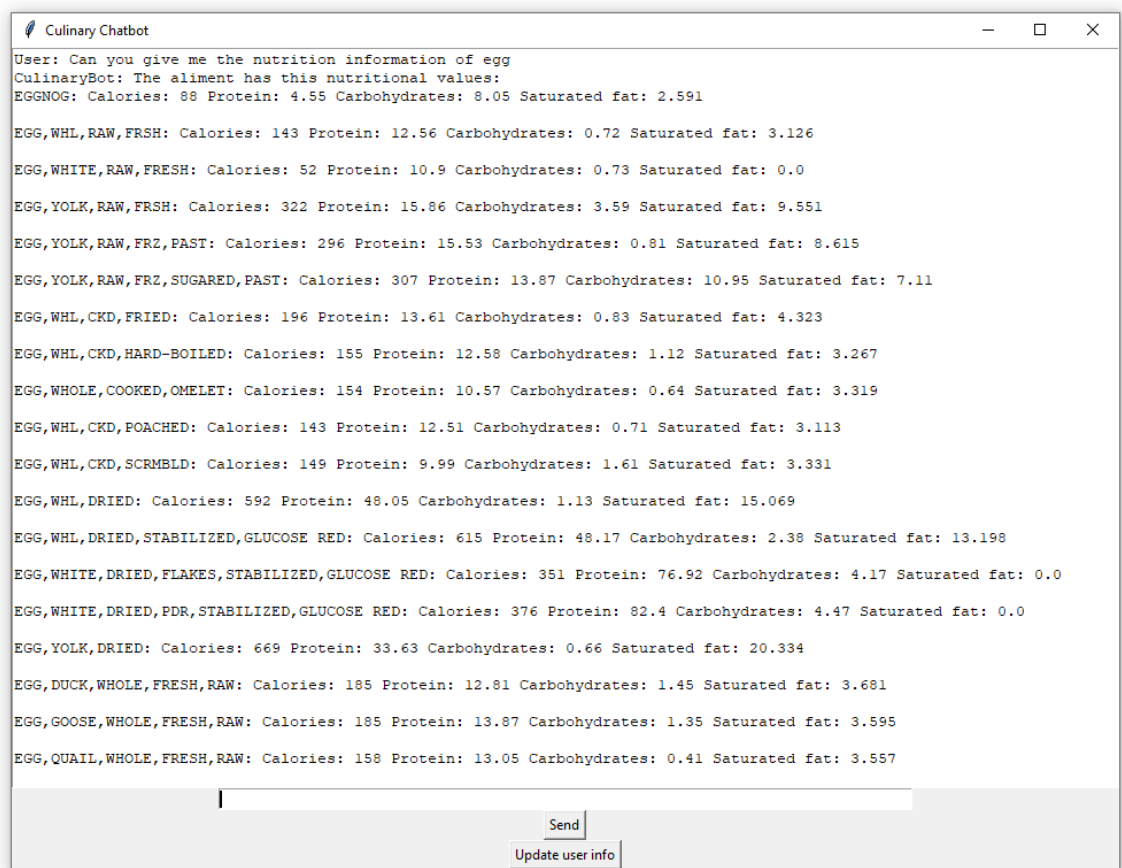
En esta prueba, el usuario solicita al chatbot un plan de comidas para ese día y el chatbot devuelve un plan (desayuno, almuerzo, comida, postre, merienda y cena) teniendo en cuenta su alergia a los lácteos, así como las calorías y nutrientes que va acumulando cada receta.

Prueba 5: Etiqueta "recipe_ingredient"



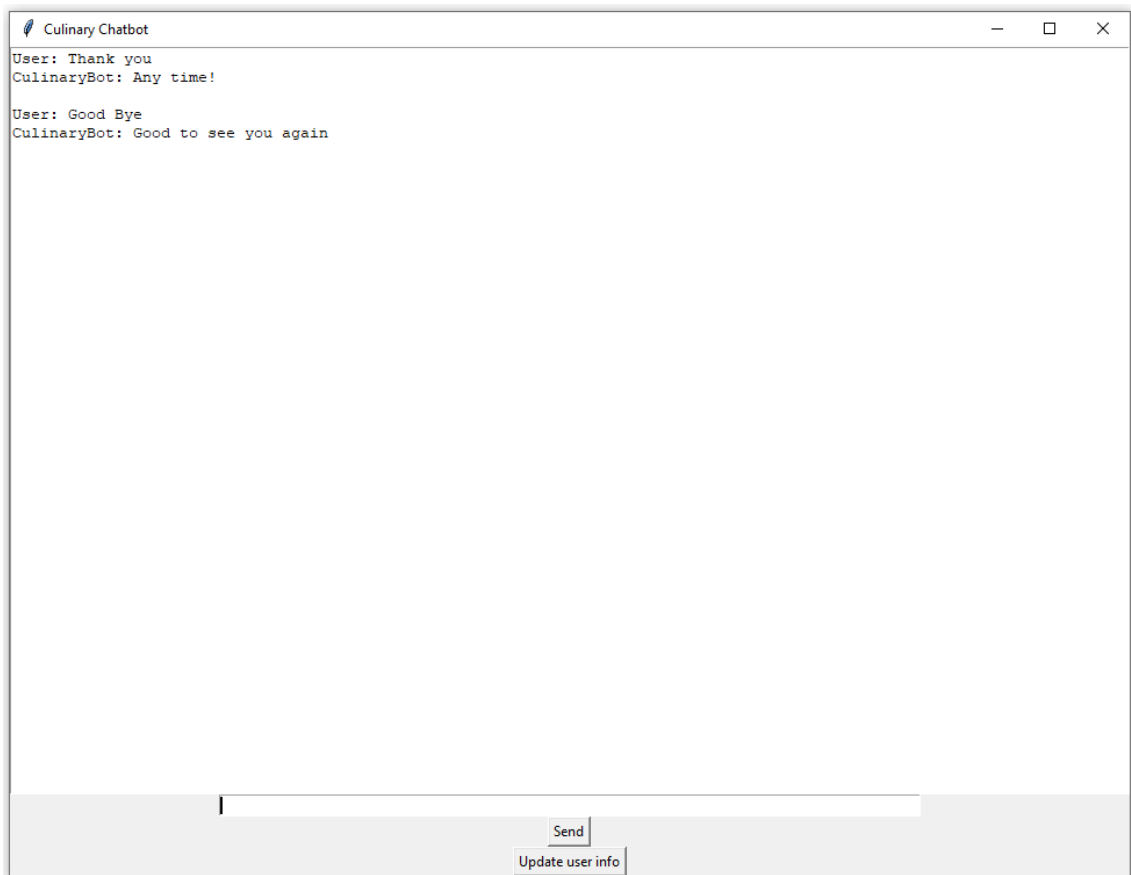
En esta prueba, el usuario le pide al chatbot una receta con el ingrediente "huevo" y el chatbot le recomienda la receta "beat this banana bread".

Prueba 6: Etiqueta “aliment_nutrition_values”



En esta prueba, se pide al chatbot los valores nutricionales de “huevo” y este devuelve el alimento huevo en sus diferentes estados o partes específicas con los valores nutricionales.

Prueba 7: Etiqueta “thanks” y “goodbye”



El usuario agradece al chatbot y, al despedirse, se predijo incorrectamente la etiqueta. Se predijo la etiqueta “greeting” en lugar de “goodbye”.

Conclusiones extraídas de las pruebas:

Cuando el usuario solicita una receta, una receta con un ingrediente específico o un plan de comidas, se verifica que el chatbot tarda en responder al usuario. Este retraso se debe a la necesidad de realizar consultas en un conjunto de datos grande. Durante estas consultas es necesario hacer varias comprobaciones, se verifica si las recetas cumplen con las preferencias del usuario, lo cual implica comprobar si se ajustan a su tipo de dieta y si no contienen alimentos que puedan provocar alergias. Además, es necesario realizar cálculos para asegurarse de que la cantidad de calorías y nutrientes que aporta la receta no exceda las recomendaciones diarias.

Para mejorar el rendimiento del chatbot, se debería considerar en el futuro mover el sistema a la nube. Esta migración permitiría reducir los tiempos de respuesta, ya que los servidores en la nube son más potentes en comparación con los procesadores de un ordenador personal, lo cual aumentaría la rapidez y agilidad en las consultas.

Adicionalmente, al migrar a la nube, es posible contar con medidas de seguridad mejores para proteger la información del usuario.

Cada vez que se recomienda una receta o plan de comidas, se tienen en cuenta las preferencias del usuario. Sin embargo, es posible que en algunas consultas el chatbot recomiende una receta que contenga algún alimento alergénico para el usuario.

Actualmente, para recomendar una receta, se realiza una comparación entre el nombre de los alimentos que pueden causar alergias y los nombres de los ingredientes de la receta. Por lo tanto, si existe algún error ortográfico, por más pequeño que sea, como, por ejemplo, “buter” en lugar de “butter”, puede llevar el sistema a considerar erróneamente que la receta es apta para el usuario.

Por otro lado, debido a la forma en que se comparan los nombres, es posible que algunas recetas que cumplen con las restricciones sean consideradas no recomendadas debido a la presencia del nombre de un alimento que provoca una alergia. Por ejemplo, si tenemos el ingrediente “soya milk”, se considerará inapropiado para personas con alergia a los productos lácteos debido a la palabra “milk” en el nombre del ingrediente de la receta.

Para enfrentar esta problemática, es necesario explorar métodos alternativos de verificación de alérgenos en las recetas. Se ha desarrollado un modelo de clasificación multi-etiqueta para identificar las alergias en cada receta. Inicialmente, se utilizarán las primeras 90000 recetas del dataset *recipes*. Concretamente, se asignó el 20 % de estos datos para la validación y se logró una precisión del 99%. Se había considerado utilizar este modelo para identificar las alergias en nuevas recetas que se agregarían a la base de datos, aunque debido a limitaciones de tiempo, no se llegó a implementar para este propósito específico. Sin embargo, hubiera sido conveniente optar por este modelo para verificar que una receta no contenga ningún alérgeno, comprobando que las etiquetas clasificadas por el modelo no incluyen la alergia a la cual el usuario es alérgico.

En el futuro, sería recomendable implementar este modelo en lugar del método actual de comparación entre los nombres de los alimentos que provocan la

alergia y los nombres de los ingredientes de la receta. Para el entrenamiento del modelo, se utilizaría una parte del dataset *recipes*, reservando otra porción para su evaluación con el fin de evitar sesgos.

Aunque este modelo no se haya implementado, habría sido la elección más acertada debido a su alta precisión y mayor confiabilidad en comparación con la simple comparación de palabras. Además, sería interesante explorar otros modelos de clasificación, como, por ejemplo, las redes neuronales artificiales, para comparar los resultados obtenidos con diferentes modelos.

Sería necesario mejorar la clasificación del tipo de receta, ya que, en algunas ocasiones, como se puede observar en la siguiente figura, en el plan de comidas se recomienda para cenar una bebida.

```
dinner: after eight shooter

Ingredients:
["1/2 ounce Tia Maria","1/2 ounce creme de menthe","1/2 ounce Baileys Irish Cream"]

Steps:
["carefully layer the tia maria , creme-de-menthe , and bailey's irish cream , in that order , into a sherry glass , ensuring they don't mix"]

To promote a balanced diet, it is important to consider the nutrients carbohydrates, fats, and protein. However, it is recommended to consume 5 servings per day of vegetables, fruits, and vegetables to obtain vitamins, minerals, and fiber..
```

Al generar un plan de comidas para un día, se ha observado que la cantidad total de carbohidratos es muy baja, mientras que la cantidad de grasas y proteínas se acerca a las cantidades recomendadas. El motivo porque no se llega a la cantidad deseada de carbohidratos es porque las recetas contienen muy pocos carbohidratos y se hace imposible, o muy difícil llegar a la cantidad recomendada de carbohidratos.

Para resolver este problema, se podría considerar ampliar la base de datos para disponer de más opciones de recetas. Además, se podrían establecer medidas específicas de distribución de nutrientes que el usuario debe consumir en cada comida. Por ejemplo, se podría fijar un porcentaje objetivo de carbohidratos para cada momento del día, como el 15% para el desayuno, 10% para el almuerzo y merienda, 20% para la comida y un 5% para la cena. Así, se podrían realizar búsquedas más precisas en la base de datos ofreciendo opciones de platos que se ajusten las cantidades recomendadas de carbohidratos para cada momento del día.

En algunas pruebas se ha podido comprobar como el modelo tiende a predecir erróneamente la etiqueta "greeting" en lugar de la etiqueta "goodbye" y la etiqueta "recipe" en lugar de "recipe_ingredient". Esto se debe a la similitud de los datos, cuando hay elementos comunes en las diferentes intenciones.

Para resolver este problema, se recomienda en el futuro realizar un análisis semántico que permitirá comprender el significado de la consulta del usuario. De esta manera, se podrá analizar la intención del usuario en base al contexto de la frase y, así, se espera distinguir con mayor precisión las diferentes intenciones.

6. Conclusiones

6.1 Conclusiones

A lo largo del proyecto, se ha desarrollado un chatbot con el objetivo de fomentar y mantener hábitos alimenticios adecuados, ofreciendo una experiencia personalizada al usuario a través de una conversación en tiempo real. El chatbot permite a los usuarios interactuar y recibir sugerencias de recetas, planes de comidas e información relevante para ayudarles a alcanzar y mantener una dieta equilibrada, satisfaciendo sus necesidades nutricionales. En las recomendaciones, el chatbot tiene en cuenta las preferencias del usuario, adaptando las recetas a sus necesidades específicas, como alergias alimentarias y tipo de dieta seguido.

Para cumplir todas las funcionalidades requeridas, se ha desarrollado una base de datos que almacena las recetas, información nutricional de los alimentos, el historial de conversaciones y la información del usuario. Con la base de datos, el chatbot consigue facilitar recomendaciones precisas al realizar consultas específicas. Adicionalmente, se ha diseñado una interfaz intuitiva que facilita la interacción con el chatbot.

Al final, se ha logrado alcanzar el producto final deseado y cumplir con todos los objetivos definidos inicialmente. Sin embargo, dada la amplitud del dominio de la alimentación, se reconoce la necesidad de considerar y cubrir muchas más intenciones del usuario para perfeccionar el chatbot y ofrecer una experiencia aún mejor al usuario.

6.2 Seguimiento de planificación y metodología

Durante el desarrollo del proyecto, fue necesario introducir cambios en la planificación debido a una distribución ineficiente de las tareas, a un cálculo incorrecto de los tiempos para su ejecución y la aparición de imprevistos que dificultaron el cumplimiento de los plazos establecidos inicialmente. Por eso, algunas tareas tuvieron que ser reprogramadas para completarse en la siguiente fase.

A continuación, se detallan los imprevistos y dificultades que llevaron a alterar algunos objetivos y buscar soluciones alternativas:

- Uno de los objetivos iniciales era recomendar recetas teniendo en cuenta las porciones que se debían ingerir de cada grupo de la pirámide alimenticia diariamente. Sin embargo, no se encontró un dataset que contuviera los ingredientes asociados a su grupo en la pirámide alimentaria, lo que obligó a modificar este objetivo. En su lugar, se optó por tomar en consideración las recomendaciones de la OMS respecto a la cantidad de nutrientes (carbohidratos, proteínas y grasas) que deben ser ingeridos diariamente. De esta manera, se utilizó el dataset “recipes”

que contiene toda la información nutricional necesaria para calcular las cantidades de nutrientes consumidas por el usuario.

- No fue posible desarrollar la interfaz en el tiempo asignado durante la PEC3. Por tanto, se llevó a cabo en la PEC4, donde inicialmente solo se había programado la redacción de la memoria.

6.3 Líneas de trabajo futuro

De cara al futuro, existen diversas funcionalidades que podrían mejorar el chatbot y la experiencia del usuario. Una de las sugerencias sería agregar una función de voz que permita a los usuarios utilizar el micrófono de sus dispositivos para interactuar con el chatbot. Esta funcionalidad adicional proporcionaría una experiencia más cómoda y práctica para el usuario.

En futuras etapas del proyecto, se podría trabajar en mejorar algunos aspectos que no han quedado completamente resueltos. Para lograr un prototipo de chatbot más robusto y efectivo, es esencial cubrir más intenciones del usuario y contemplar todos los posibles escenarios. Además, un mayor aporte de datos contribuiría a mejorar los resultados de la red neuronal, lo que a su vez aumentaría la precisión del modelo. Por otro lado, se debería considerar la retroalimentación por parte del usuario para enriquecer el contenido del chatbot con nuevas intenciones.

En relación a la recomendación de recetas y la planificación de comidas, sería necesario considerar la posibilidad de que los usuarios rechacen las recetas o planes propuestos.

Actualmente, se ha establecido una relación entre dietas y recetas en la base de datos, pero habría sido más apropiado establecer una relación entre ingredientes y dietas. Sin embargo, debido a las limitaciones de tiempo, no fue posible establecer esta última relación.

Es importante destacar que, al generar un plan de comidas para un día, se ha observado que la cantidad total de carbohidratos es muy baja mientras que la cantidad de grasas y proteínas se acerca a las cantidades recomendadas. El motivo porque no se llega a la cantidad deseada de carbohidratos es porque las recetas contienen muy pocos carbohidratos y se hace imposible, o muy difícil llegar a la cantidad recomendada de carbohidratos. Este aspecto debería ser considerado y ajustado, para garantizar que el usuario consume la cantidad de carbohidratos necesaria diariamente.

7. Glosario

Etiqueta: La etiqueta identifica la intención del usuario.

Json: Librería codificar y decodificar archivos con formato JSON.

JSON: Formato para el intercambio de datos.

Keras: Una API de Deep Learning diseñada para implementar redes neuronales fácilmente.

Lematización: Convierte las palabras a su forma base, conocida como lema.

Matplotlib: Es una librería de Python que permite crear gráficos con listas o arrays.

NLTK: Sigla para Natural Language Toolkit. Conjunto de librerías y programas diseñados para el procesamiento del lenguaje natural.

NumPy: Librería para trabajar con vectores y matrices. Además, ofrece varias funciones matemáticas para trabajar con este tipo de datos.

OMS: Sigla para la Organización Mundial de la Salud.

Patrones: Los patrones representan las frases o preguntas que el usuario puede realizar.

Pickle: Librería para la serialización de objetos Python.

Random: Librería para generar números pseudoaleatorios.

Respuestas: Contiene las posibles respuestas que el chatbot dará al usuario.

Scikit-learn: Es una librería de Python con algoritmos de clasificación, regresión y clustering.

SQLite: Sistema de gestión de bases de datos relacional.

Stemming: Convierte una palabra a su raíz eliminando los afijos de las palabras.

Tokenización: División de las frases en partes más pequeñas conocidas como tokens.

Visual Studio Code: Editor de código fuente.

8. Bibliografía

Bibliografía utilizada en la memoria del trabajo:

1. Martín, Atala. (31 de marzo de 2022). “La salud de nuestros hijos está en su alimentación”. [en línea]. La Nueva España. [consultado el 17 de abril de 2023]. Disponible en:
<https://www.lne.es/sociedad/2022/03/31/salud-hijos-alimentacion-64490170.html>
2. Cañizares, Francisco. (23 de junio de 2022). “El tremendo peso de la obesidad en España”. [en línea]. El País. [consultado el 17 de abril de 2023]. Disponible en:
<https://elpais.com/sociedad/el-peso-de-la-salud/2022-06-23/el-tremendo-peso-de-la-obesidad-en-espana.html>
3. (12 de enero de 2023). “Un 59% de los clientes prefiere comunicarse con chatbots de las marcas para solucionar sus dudas”. [en línea]. PORTALTIC. [consultado el 20 de abril de 2023]. Disponible en:
<https://www.europapress.es/portaltic/empresas/noticia-59-clientes-prefiere-comunicarse-chatbots-marcas-solucionar-dudas-20230112092948.html>
4. Domenes Mondragón, Francisco. (26 de diciembre de 2022). Trabajo Fin de Grado “JackBot – El chatbot compañero”. [en línea]. [consultado el 23 de abril de 2023]. Disponible en:
<https://openaccess.uoc.edu/bitstream/10609/147346/5/fdomenesTFG0123memoria.pdf>
5. Pino Gamazo, Marcos. (febrero 2021). Trabajo Fin de Grado “Desarrollo de un Chatbot para Público Infantil para Classificar Sonidos del Cielo. Versión 2”. [en línea]. [consultado el 23 de abril de 2023]. Disponible en:
https://oa.upm.es/66306/1/TFG_MARCOS_PINO_GAMAZO.pdf
6. Aguado Serrano, Teresa. (junio de 2018). Trabajo Fin de Grado “Tutor basado en chatbot”. [en línea]. [consultado el 23 de abril de 2023]. Disponible en:
<https://openaccess.uoc.edu/bitstream/10609/81274/7/taserranoTFG0618memoria.pdf>
7. “Blue”. [en línea]. BBVA. [consultado el 23 de abril de 2023]. Disponible en:
<https://www.bbva.es/personas/banca-online/chatbots.html>
8. (5 de febrero de 2023). “What are the ways or libraries to train Chatbot in Python”. [en línea]. AAHFOO. [consultado el 23 de abril de 2023]. Disponible en:

<https://www.aahfoo.com/2023/02/05/internet/what-are-the-ways-or-libraries-to-train-chatbot-in-python/>

9. (30 de julio de 2018). “La inteligencia artificial acelera la evolución de los ‘chatbots’”. [en línea]. BBVA. [consultado el 23 de abril de 2023]. Disponible en:
<https://www.bbva.com/es/innovacion/inteligencia-artificial-acelera-evolucion-chatbots/>
10. Li, Shuyang. Majumder Prasad, Bodhisattwa. “Food.com Recipes and Interactions”. [en línea]. Kaggle. [consultado el 23 de abril de 2023]. Disponible en:
https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions?select=RAW_recipes.csv
11. Li, Shuyang. “Food.com Recipes with Search Terms and Tags”. [en línea]. Kaggle. [consultado el 23 de abril de 2023]. Disponible en:
<https://www.kaggle.com/datasets/shuyangli94/foodcom-recipes-with-search-terms-and-tags>
12. Genzano, Leo. “Nutrientes Análisis exploratorio (EDA)”. [en línea]. Kaggle. [consultado el 24 de abril de 2023]. Disponible en:
<https://www.kaggle.com/code/leogenzano/nutrientes-an-lisis-exploratorio-eda/input>
13. Aadya. “Food: Allergens and Allergies”. [en línea]. Kaggle. [consultado el 24 de abril de 2023]. Disponible en:
<https://www.kaggle.com/datasets/boltcutters/food-allergens-and-allergies>
14. NLTK [en línea]. [consulta: 16 de abril de 2023]. Disponible en:
<https://www.nltk.org/>
15. Matplotlib [en línea]. [consulta: 16 de abril de 2023]. Disponible en:
<https://matplotlib.org/>
16. Scikit-learn [en línea]. [consulta: 17 de abril de 2023]. Disponible en:
<https://scikit-learn.org/stable/>
17. LogisticRegression [en línea]. [consulta: 19 de abril de 2023]. Disponible en:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

18. OneVsRestClassifier [en línea]. [consulta: 19 de abril de 2023]. Disponible en: <https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html>
19. SQLite [en línea]. [consulta: 22 de abril de 2023]. Disponible en: <https://sqlite.org/index.html>
20. Ravikiran A S. (16 de febrero de 2023). [en línea]. simplilearn. [consultado el 27 de abril de 2023]. Disponible en: https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite#why_use_sqlite
21. Visual Studio Code [en línea]. [consulta: 4 de abril de 2023]. Disponible en: <https://code.visualstudio.com/>
22. Malik, Karan. (15 de junio de 2020). intents.json. [en línea]. Github. [consulta: 4 de abril de 2023]. Disponible en: https://github.com/Karan-Malik/Chatbot/blob/master/chatbot_codes/intents.json
23. JSON [en línea]. [consulta: 13 de junio de 2023]. Disponible en: <https://www.json.org/json-en.html>
24. Keras [en línea]. [consulta: 13 de junio de 2023]. Disponible en: <https://keras.io/>
25. Nagpal, Manika. (24 de abril de 2023). "Python Chatbot Project-Learn to build a chatbot from Scratch". [en línea]. ProjectPro. [consulta: 13 de junio de 2023]. Disponible en: <https://www.projectpro.io/article/python-chatbot-project-learn-to-build-a-chatbot-from-scratch/429#toc-8>
26. Miller, Stephan. (26 de agosto de 2022). "Top 6 Programming Languages for Chatbot Development". [en línea]. codecademy. [consulta: 16 de junio de 2023]. Disponible en: <https://www.codecademy.com/resources/blog/top-6-programming-languages-for-chatbot-development/#:~:text=Python%20is%20a%20preferred%20language,easy%20to%20read%20and%20understand.>
27. Chatterbot [en línea]. [consulta: 13 de junio de 2023]. Disponible en: <https://pypi.org/project/ChatterBot/>

28. Rasa [en línea]. [consulta: 13 de junio de 2023]. Disponible en: <https://pypi.org/project/rasa/>
29. TensorFlow [en línea]. [consulta: 13 de junio de 2023]. Disponible en: <https://www.tensorflow.org/?hl=es-419>
30. “Espacio de recursos de ciencia de datos” – PID_00275836. [en línea]. UOC. [consulta: 13 de junio de 2023]. Disponible en: <http://datascience.recursos.uoc.edu/es/keras-tensorflow/>
31. (31 de enero de 2022). “Keras: todo sobre la API de Deep Learning”. [en línea]. datascientest. [consulta: 16 de junio de 2023]. Disponible en: <https://datascientest.com/es/keras-la-api-de-deep-learning>
32. NumPy [en línea]. [consulta: 16 de junio de 2023]. Disponible en: <https://numpy.org/>
33. Pickle [en línea]. [consulta: 16 de junio de 2023]. Disponible en: <https://docs.python.org/es/3/library/pickle.html>
34. Random [en línea]. [consulta: 16 de junio de 2023]. Disponible en: <https://docs.python.org/es/3.10/library/random.html#:~:text=Se%20utiliza%20para%20el%20muestreo%20aleatorio%20sin%20reemplazo.>
35. Json [en línea]. [consulta: 16 de junio de 2023]. Disponible en: <https://docs.python.org/es/3/library/json.html>
36. Documentation. NLTK. [en línea]. [consulta: 16 de junio de 2023]. Disponible en: https://www.nltk.org/_modules/nltk/stem/wordnet.html
37. “Espacio de recursos de ciencia de datos” – PID_00275836. [en línea]. UOC. [consulta: 13 de junio de 2023]. Disponible en: <http://datascience.recursos.uoc.edu/es/procesamiento-del-lenguaje-natural-nlp/>
38. Kumar, Dhiraj. “Chat Bot using Python, Machine Learning, NLP, Keras and NLTK”. [en línea]. BOT BARK. [consulta: 17 de junio de 2023]. Disponible en: <https://botbark.com/2023/03/13/chat-bot-using-python-machine-learning-nlp-keras-and-nltk/>
39. Tkinter [en línea]. [consulta: 17 de junio de 2023]. Disponible en: <https://docs.python.org/es/3/library/tkinter.html>
40. datetime [en línea]. [consulta: 17 de junio de 2023]. Disponible en: <https://docs.python.org/es/3/library/datetime.html>

41. Manmayi. (28 de febrero de 2023). "Choose optimal number of epochs to train a neural network in Keras". [en línea]. geeksforgeeks. [consulta: 17 de junio de 2023]. Disponible en: <https://www.geeksforgeeks.org/choose-optimal-number-of-epochs-to-train-a-neural-network-in-keras/>
42. "Ecuación de Harris-Benedict". [en línea]. Wikipedia. [consulta: 17 de junio de 2023]. Disponible en: https://es.wikipedia.org/wiki/Ecuaci%C3%B3n_de_Harris-Benedict
43. "pirámide alimentaria". [en línea]. guía metabólica. [consulta: 17 de junio de 2023]. Disponible en: <https://metabolicas.sjdhospitalbarcelona.org/etiquetas/piramide-alimentaria>

Documentos consultados para obtener información:

- Aguado Serrano, Teresa. (junio de 2018). Trabajo Fin de Grado "Tutor basado en chatbot". [en línea]. [consultado el 23 de abril de 2023]. Disponible en: <https://openaccess.uoc.edu/bitstream/10609/81274/7/taserranoTFG0618memoria.pdf>
- Álvarez Ruiz, Anesti. (enero 2018). Trabajo Fin de Grado "Estudi i creació d' un assistent virtual (xatbot)". [en línea]. [consultado el 10 de abril de 2023]. Disponible en: <https://openaccess.uoc.edu/bitstream/10609/71725/9/aalvarezruTFG0118mem%c3%b2ria.pdf>

Este archivo ha sido utilizado como referencia para diseñar el flujo de conversación del chatbot.

- Kariuki, Charles. "Multi-Label Classification with Scikit-MultiLearn". [en línea]. Section. [consultado el 10 de abril de 2023]. Disponible en: <https://www.section.io/engineering-education/multi-label-classification-with-scikit-multilearn/>
- Joshi, Prateek. (19 de julio de 2022). "Predicting Movie Genres using NLP – An Awesome Introduction to Multi-Label Classification". [en línea]. Analytics Vidhya. [consultado el 10 de abril de 2023]. Disponible en: <https://www.analyticsvidhya.com/blog/2019/04/predicting-movie-genres-nlp-multi-label-classification/>
- Browniee, Jason. (13 de abril de 2020). "One-vs-Rest and One-vs-One for Multi-Class Classification". [en línea]. Machine Learning Mastery. [consultado el 15 de abril de 2023]. Disponible en:

<https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/>

- Quintana, Ambar. (23 de octubre de 2020). “Relaciones entre tablas de bases de datos”. [en línea]. Styde. [consultado el 20 de mayo de 2023]. Disponible en: <https://styde.net/relaciones-entre-tablas-de-bases-de-datos/>
- Vincent Abba, Ihechikara. (20 de septiembre de 2022). “Python Datetime.now() – How to Get Today's Date and Time”. [en línea]. freeCodeCamp. [consultado el 22 de mayo de 2023]. Disponible en: <https://www.freecodecamp.org/news/python-datetime-now-how-to-get-todays-date-and-time/>
- Kozaczko, Dominik. (26 de junio de 2018). “8 best Python Natural Language Processing (NLP) libraries”. [en línea]. sunscrapers. [consultado el 10 de abril de 2023]. Disponible en: <https://sunscrapers.com/blog/8-best-python-natural-language-processing-nlp/>
- (2 de agosto de 2022). “10 Librerías populares de procesamiento del lenguaje natural”. [en línea]. Gobierno de España. [consultado el 15 de abril de 2023]. Disponible en: <https://datos.gob.es/es/blog/10-librerias-populares-de-procesamiento-del-lenguaje-natural>
- Wali, Kartik. (4 de mayo de 2022). “Explained: Stemming vs lemmatization in NLP”. [en línea]. analyticsindiamag. [consultado el 16 de abril de 2023]. Disponible en: <https://analyticsindiamag.com/explained-stemming-vs-lemmatization-in-nlp/#:~:text=Lemmatization%20has%20higher%20accuracy%20than,the%20context%20is%20not%20important.>
- Utrera Burgal, Jesús. (20 de junio de 2018). “Deep Learning básico con Keras (Parte 1)”. [en línea]. enmilocalfunciona. [consultado el 16 de abril de 2023]. Disponible en: <https://www.enmilocalfunciona.io/deep-learning-basico-con-keras-parte-1/>
- Rodríguez, Vicente. (9 de noviembre de 2018). “Dropout y Batch Normalization”. [en línea]. vincentblog. [consultado el 16 de abril de 2023]. Disponible en: <https://vincentblog.xyz/posts/dropout-y-batch-normalization>

- “How do I write JSON data to a file?”. [en línea]. Stack overflow. [consultado el 14 de mayo de 2023]. Disponible en: <https://stackoverflow.com/questions/12309269/how-do-i-write-json-data-to-a-file>
- Muthuraman, Vishwanath. (30 de diciembre de 2022). “Building a Chatbot with Python and tkinter library for the GUI”. [en línea]. medium. [consultado el 13 de junio de 2023]. Disponible en: https://medium.com/@vishwanathmuthuraman_92476/building-a-chatbot-with-python-and-tkinter-library-for-the-gui-390a747dadf6

Este archivo ha sido utilizado como referencia para crear la interfaz.

- Quora. [en línea]. [consultado el 28 de junio de 2023]. Disponible en: <https://www.quora.com/Why-does-a-query-run-faster-on-the-cloud>

Para las ilustraciones se han utilizado las siguientes herramientas:

Ilustración 1: Excel

Ilustración 2: Excel

Ilustración 3: Draw.io

Ilustración 4:

Se obtuvo el diagrama en la función `_model_multi_label_classification_allergies` de la clase `DataPreparation()` utilizando las librerías `NLTK` [14] y `matplotlib` [15] y con el editor de texto Visual Studio Code [21].

Ilustración 5: Draw.io

Ilustración 6: Draw.io

Ilustración 7: Librería Tkinter.

Ilustración 8: Librería Tkinter.

9. Anexos

En este espacio https://drive.google.com/drive/folders/1f_DWIJAt55Bt-vEx_wGv3C-qXmM0MQtq?usp=sharing, se encuentran todos los archivos mencionados.

Además, se pueden consultar los archivos y vídeos de pruebas realizadas que muestran el funcionamiento del sistema en detalle en GitLab: <https://gitlab.com/susananobrega/TFG-EIChatbotCulinario.git>

9.1 Anexo 1 – Ejemplo de algunas intenciones del archivo “intents.json”

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": [
        "Hi",
        "How are you",
        "Hello"
      ],
      "responses": [
        "Hello",
        "Good to see you again",
        "Hi there, how can I help you?"
      ],
      "context": [
        ""
      ]
    },
    {
      "tag": "recipe_ingredient",
      "patterns": [
        "Can you give me a recipe with the {ingredients}.",
        "Can you suggest me a recipe with the {ingredients}.",
        "I want a recipe with {ingredients}.",
        "Give me a recipe with {ingredients}."
      ],
      "responses": [
        "Yes, sure. Here is a recipe: {recipe}"
      ]
    },
    # ...
  ]
}
```

