
Chatbot para la recuperación de información personal



Trabajo de Fin de Grado
Curso 2021–2022

Autora
Lucía Latorre Magaz

Directores
Gonzalo Méndez Pozo
Pablo Gervás Gómez-Navarro

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Chatbot para la recuperación de información personal

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Ingeniería del Software e Inteligencia
Artificial

Autora
Lucía Latorre Magaz

Directores
Gonzalo Méndez Pozo
Pablo Gervás Gómez-Navarro

Dirigida por el Doctor
Gonzalo Méndez Pozo
Pablo Gervás Gómez-Navarro

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

23 de diciembre de 2022

Dedicatoria

Texto de la dedicatoria...

Agradecimientos

Texto de los agradecimientos

Resumen

Resumen en español del trabajo

Palabras clave

Máximo 10 palabras clave separadas por comas chatbot, reminiscencia, demencia, recuerdo

Abstract

Abstract in English.

Keywords

10 keywords max., separated by commas.

Índice

1. Introduction	1
1. Introducción	3
1.1. Motivación	3
1.2. Objetivos	3
1.3. Plan de trabajo	4
2. Estado de la Cuestión	5
2.1. Demencia	5
2.2. Terapia ocupacional basada en reminiscencia	5
2.3. Trabajo previo	6
2.4. Procesamiento del lenguaje natural	6
2.4.1. Modelos para el PLN	7
2.4.2. Componentes del PLN	7
2.4.3. Aplicaciones del procesamiento del lenguaje natural . .	8
2.4.4. Ventajas del PLN	8
2.4.5. Chatbots	9
2.4.6. Análisis de sentimiento	11
3. Arquitectura del Chatbot	13
3.1. Clasificación de los recuerdos	13
3.1.1. Pruebas del análisis de sentimiento	15
3.1.2. Clasificación en etapas de vida	18
3.2. Procesamiento del texto para encadenar preguntas y respuestas	22
3.2.1. Pruebas	23
3.3. Base de datos MongoDB	25
4. Aplicación Web	27
4.1. Prototipo inicial	27

4.2.	Bases de datos	32
4.2.1.	MySQL	32
4.3.	Implementación final	35
4.3.1.	Funcionalidad inicio de sesión y registro	37
4.3.2.	Funcionalidad chatbot	38
4.3.3.	Funcionalidad usuario terapeuta vs usuario paciente .	38
4.3.4.	Funcionalidad consulta de pacientes registrados por parte del terapeuta	39
4.3.5.	Funcionalidad crear nuevo paciente	39
4.3.6.	Funcionalidad historia de vida de un paciente	39
5.	Conclusiones y Trabajo Futuro	43
5.	Conclusions and Future Work	45
	Bibliografía	47

Índice de figuras

4.1. Prototipo inicial de la aplicación para el usuario con demencia	28
4.2. Prototipo inicial de la aplicación para el terapeuta: Consultar los pacientes registrados	29
4.3. Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 1	29
4.4. Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 2	30
4.5. Prototipo inicial de la aplicación para el terapeuta: Crear y consultar las terapias creadas	30
4.6. Prototipo inicial de la aplicación para el terapeuta: Consultar y reproducir una terapia concreta	31
4.7. Diagrama de entidad-relación MySQL	33
4.8. Diagrama de la tabla de usuarios	34
4.9. Diagrama de la tabla de terapias	35
4.10. Funcionalidad consultar pacientes registrados	39
4.11. Funcionalidad consultar pacientes registrados	39
4.12. Funcionalidad consultar pacientes registrados	40
4.13. Funcionalidad crear nuevo paciente	40
4.14. Funcionalidad crear nuevo paciente	41

Índice de tablas

Chapter 1

Introduction

Introduction to the subject area.

Introducción

“Frase célebre dicha por alguien inteligente”

— Autor

Introducción temporal

Las personas con Alzheimer u otros tipos de demencia pueden beneficiarse del uso de la llamada terapia basada en reminiscencia, que se basa en la construcción de un libro de vida del paciente que recopila recuerdos positivos de su vida que se pueden utilizar posteriormente para ejercitarse su memoria y retrasar el deterioro, además de permitir aumentar el bienestar de los pacientes.

En el presente proyecto se propone el desarrollo de un chatbot que permita recopilar y estructurar esta información para ayudar a los terapeutas en la elaboración de los libros de vida.

1.1. Motivación

Introducción al tema del TFG.

1.2. Objetivos

Este proyecto tiene como objetivo desarrollar un chatbot mediante el cual recabar información personal sobre la vida del paciente con demencia, clasificarla y estructurarla siguiendo un esquema que pueda facilitar la tarea de los terapeutas a la hora de construir un libro de vida.

La clasificación de recuerdos se hará en base a unos criterios predefinidos por expertos en terapia ocupacional del proyecto CANTOR. Se clasificará en función de:

- **Emoción:** Se clasificarán los recuerdos en positivos y negativos, siendo estos últimos para identificar qué recuerdos no deben tratarse en las

terapias por afligir al paciente. Los positivos se puntuarán de 0 a 10 en función de la felicidad que le traen al paciente.

- **Etapa:** Los recuerdos pertenecerán a una de las siguientes etapas: infancia, adolescencia, edad adulta o tercera edad según el periodo temporal en que aconteció.
- **Categorías:** Cada recuerdo entrará dentro de una o varias categorías que recojan una característica del recuerdo. Ejemplos de categorías: guerra civil, bailes, ocio, familia, aficiones...

1.3. Plan de trabajo

El plan de trabajo ha consistido de tres etapas:

- Investigación y construcción de prototipo en la que se ha consolidado la idea del TFG, investigado sobre demencia y terapia ocupacional basada en reminiscencia, elegido tecnologías y creado un prototipo de análisis de texto usando spaCy para identificar si un recuerdo es positivo o negativo.
- Programación del Chatbot y desarrollo de la memoria
- Pruebas, revisión de la memoria y entrega

Capítulo 2

Estado de la Cuestión

Introducción de lo que voy a hablar en el estado de la cuestión y por qué

2.1. Demencia

La demencia¹ es una condición neurodegenerativa progresiva, caracterizada por un deterioro cognitivo que interfiere con la vida cotidiana afectando a la memoria, al pensamiento, al lenguaje, al juicio y al comportamiento. La demencia no es una enfermedad específica aunque la mayor parte de los casos de demencia son provocados por la enfermedad de Alzheimer. Muchas veces se confunde la demencia con una consecuencia más del envejecimiento, cuando no tiene por qué ser así.

Hay muchos síntomas asociados a la demencia pero, en este trabajo, nos vamos a centrar en la pérdida de memoria. Trabajar los recuerdos de una persona que sufre demencia ayuda a retrasar los efectos de la misma. Hablaremos para ello de la terapia ocupacional basada en reminiscencia.

2.2. Terapia ocupacional basada en reminiscencia

La terapia ocupacional² se centra en que el paciente sea capaz de participar en las actividades de la vida cotidiana. Es decir, se basa en ayudar al individuo a llevar una vida lo más normal posible adaptando las tareas cotidianas a realizar o el entorno para que pueda llevarlas a cabo.

(Sacar información del seminario cantor) La terapia ocupacional basada en reminiscencia se centra en mejorar la calidad de vida de la persona con demencia. Se trata de una técnica basada en la recuperación de recuerdos

¹<https://www.alz.org/alzheimer-demencia/que-es-la-demencia?lang=es-MX>

²<https://aptoca.org/terapia-ocupacional/que-es-la-terapia-ocupacional-2/>

dentro de un periodo de tiempo en la vida de la persona con el objetivo de construir la historia de vida del sujeto. La historia de vida surge de la sucesión de acontecimientos que componen la totalidad de la vivencia del sujeto.

2.3. Trabajo previo

Herramienta de ayuda guiada para la reminiscencia (rem) : Generación de historias a partir de una base de conocimiento: recomendación de temas a tratar en la terapia + aplicación web que enlaza situaciones y vivencias mediante grafos y luego permite añadir recursos fotográficos asociado a un tema. (Más como un chatbot que va sugiriendo temas a tratar)

Sistema de asistencia para cuidados de enfermos del Alzheimer (asi) : Página que guarda información sobre pacientes y terapeutas asociados. Información relevante + historia de vida formada por instancias de recuerdos

Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning (pre) : chat desplegado con telegram. De las fotos que tiene archivadas va preguntando al usuario cosas relacionadas con la imagen

Generación de resúmenes de video-entrevistas utilizando redes neuronales (res) : transcripción de video-entrevistas a texto

Extracción de información personal a partir de redes sociales para la creación de un libro de vida (rrs)

Alameda Salas, María Cristina (2022) Generación de historias de vida usando técnicas de Deep Learning.

Barquilla Blanco, Cristina y Díez García, Patricia y Mulas López, Santiago Marco y Verdú Rodríguez, Eva (2022) Recuérdame: Aplicación de apoyo para el tratamiento de personas con problema de memoria mediante terapias basadas en reminiscencia.

García González, Hugo (2022) Extracción de recuerdos de vídeos de entrevistas con personas con problemas de memoria.

2.4. Procesamiento del lenguaje natural

El Procesamiento del Lenguaje Natural³ es un campo de la Inteligencia Artificial que estudia las interacciones entre personas y máquinas mediante el uso del lenguaje natural, es decir, investiga cómo pueden comunicarse las computadoras y los humanos de forma eficiente. El PLN es un campo que se ha estado desarrollando durante los últimos 50 años y que, aunque tuvo poco éxito en un principio, en la actualidad, se emplea en muchos ámbitos. Es por intereses económicos y prácticos que los desarrollos en este

³<https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/>

área de conocimiento se hayan realizado para las lenguas más habladas como el inglés, alemán, español y chino. No obstante, existen muchas herramientas muy potentes que trabajan muchos idiomas como por ejemplo, el traductor de Google. El desarrollo de técnicas de procesamiento de lenguaje natural es vital también para el funcionamiento de los chatbots (tema en el que está centrado este TFG) como podrían ser los tan reconocibles Siri de Apple y Alexa de Amazon. Es importante tener en cuenta que se avanza mucho más en el lenguaje por escrito ya que hay muchos más datos y es más fácil de guardar en formato electrónico.

2.4.1. Modelos para el PLN

Para que las máquinas puedan tratar el lenguaje natural es necesario describirlo en términos matemáticos porque los ordenadores solo entienden de bytes. Existen dos formas de modelar el lenguaje:

1. **Modelo gramatical:** Esta formado por reglas de reconocimiento de patrones estructurales relacionados con la fonética, la morfología, la semántica, la sintaxis etc. Estas reglas las definen expertos lingüistas y son las que permiten a las máquinas reconocer lo que solicita la persona.
2. **Modelo probabilístico:** Se aplican técnicas matemáticas para extraer el conocimiento. En vez de usar reglas gramaticales, se recoge una gran cantidad de ejemplos y datos para calcular la frecuencia de aparición de las diferentes unidades lingüísticas (letras, palabras, oraciones) y su probabilidad de aparecer en un contexto determinado. Con estas probabilidades se puede predecir mucha información. Este modelo es lo que se denomina aprendizaje automático.

2.4.2. Componentes del PLN

Existen varios tipos de análisis para extraer información del lenguaje natural y sus usos dependerán del objetivo de la aplicación:

- **Análisis morfológico:** Se analiza la estructura interna de las palabras para clasificarlas en categorías (sustantivos, verbos, adjetivos, etc.) y extraer los lemas
- **Análisis sintáctico:** Consiste en estudiar la estructura sintáctica de las oraciones a partir de una gramática de la lengua en cuestión.
- **Análisis semántico:** Se utiliza para extraer el significado de las oraciones.
- **Análisis pragmático:** Añade al análisis el contexto de uso del lenguaje para mejorar la interpretación.

2.4.3. Aplicaciones del procesamiento del lenguaje natural

⁴ Algunas son:

Traducción automática de textos: actualmente se ha llegado a un nivel de traducción bastante razonable, que permite traducir textos en internet de textos en distintos idiomas al nuestro. Incluso existen algunos navegadores que traducen automáticamente de un idioma al otro. Ejemplo: Google Chrome puede traducir automáticamente páginas web del neerlandés a inglés o al español. Suelen ser más fiables ciertas parejas de idiomas. Por ejemplo, el navegador de Google traduce mejor del neerlandés al inglés que al español, probablemente por tener más uso y, por tanto, desarrollo por parte de esta empresa.

Aún no se ha conseguido la traducción compleja de documentos o aquellos que tienen muchos matices, ni siquiera de un idioma principal a otro. Para dichos textos seguirá siendo necesaria la intervención de un humano experto en traducciones.

Sistemas conversacionales con PLN: ejemplos son Siri de Apple o el Asistente de Google que entablan pequeñas conversaciones con el usuario y resuelven dudas, aunque a veces no con mucho éxito aún.

Respuestas automáticas a preguntas: el primero que se hizo famoso fue el sistema Watson, desarrollado por IBM y que ganó algunos concursos en la televisión contra humanos.

Análisis de sentimiento en redes sociales: algunas aplicaciones como la desarrollada por el Instituto de Ingeniería del Conocimiento (ver referencia abajo) analizan opiniones sobre personas, productos y temas varios. También se utilizan para analizar cómo interactúan los usuarios de dichas redes.

Resúmenes de textos automáticos: en el mundo de hoy con tanta información online, muchos textos quedan sin leer por falta de tiempo. Estos resúmenes ayudan a determinar si un texto merece ser leído al completo.

Clasificación de documentos por categorías: ayuda a dirigir la información contenida en dichos documentos a los usuarios interesados, ahorrando tiempo.

2.4.4. Ventajas del PLN

Ahorro de tiempo al usuario. Ahorro de costes (traducciones automáticas en vez de acudir a traductor profesional). Mejora de la comunicación entre personas de distintas culturas que hablan distintas lenguas. Facilita el turismo (el móvil me traduce a mi idioma un texto que veo en japonés con la cámara de mi móvil, por ejemplo). Agiliza el etiquetado manual de documentos. Ayuda a tomar decisiones de negocio. Por ejemplo: el análisis

⁴<https://www.iic.uam.es/procesamiento-del-lenguaje-natural/aplicaciones-procesamiento-lenguaje-natural/>

automático de redes sociales permite detectar una posible crisis de reputación con rapidez y atajarla con mayor brevedad.

2.4.5. Chatbots

Un Chatbot o asistente virtual inteligente⁵ es un programa informático capaz de mantener una conversación real con un usuario en lenguaje natural. Dan respuesta a dudas y tareas planteadas por los internautas. Para desarrollar un Chatbot se suele usar Procesamiento del Lenguaje Natural (PLN) y “Machine Learning”. Algunos ejemplos de asistentes virtuales serían los que recomiendan viajes y lugares turísticos, para la compra de billetes de avión, para aprender idiomas y mejorar las habilidades lingüísticas, gestiones en banca, para consultar dudas sobre contratos de telefonía móvil, etc.

Muchas empresas disponen de un Chatbot para atender a sus clientes y resolver las dudas más frecuentes. Así, de paso, ahorran costes y dejan las preguntas más difíciles para los “call centres” o los chats con agentes humanos. Se consigue así contratar menos agentes, ya que muchas de las preguntas que realizan los usuarios son repetitivas. Son capaces de interpretar lo que el usuario pide a través del texto que introduce o lo que dice y de mantener una conversación y dar respuestas concretas.

2.4.5.1. Tipos

Hay distintos tipos de Chatbots:

Aquellos que resuelven temas concretos relativamente complejos dentro de una misma temática de una empresa en particular como, por ejemplo, la compra de billetes de tren, la compra de entradas, resolver dudas sobre contrato y ofertas de telefonía móvil (ejemplo: asistente del operador Orange), etc.

Algunos de ellos se han desarrollado para contestar por WhatsApp, lo que los hace más fáciles e intuitivos de usar para el usuario.

Otros se han diseñado para escuchar al usuario en conversaciones cortas como el Siri de Apple que interpreta lo que pide el usuario y resuelve dudas como qué tiempo va a hacer, quién ha ganado un partido de fútbol o ponen canciones.

Finalmente existen otros aún más complejos que mantienen conversaciones con los usuarios como si fueran personas con sentimientos, empatía, conocimiento, etc.

Todos estos asistentes virtuales necesitan entender el lenguaje natural para recibir las peticiones por un lado y por otro ser capaces de generar lenguaje natural para contestar.

⁵<https://www.iic.uam.es/procesamiento-del-lenguaje-natural/como-crear-chatbot-con-machine-learning-y-pln>

Desarrollo Existen varias herramientas disponibles en la web de empresas reconocidas que ayudan a crearlos. Ejemplo son Language Understanding (LUIS) de Microsoft, Google Dialogflow o Watson Assistant de IBM.

Dichas herramientas ayudan a crear el agente virtual para extraer los datos de las conversaciones, buscar en bases de datos, dar respuestas y entrenar al agente para que cada vez funcione mejor.

Con entrenar al agente nos referimos a que detecte la petición del usuario en el texto que introduce o dice. Para ello, proporcionaremos a la herramienta ejemplos de oraciones que los clientes pueden introducir. Cuando el usuario real introduzca una frase con una petición similar, el agente será capaz de deducir lo que pide.

Para algunas de las peticiones, el Chatbot dará una respuesta inmediata y directa, contestando, por ejemplo, al horario de una tienda física.

En otros casos el agente requerirá más información al usuario y planteará una pregunta al mismo para recabar dicha información. Por ejemplo, le puede pedir el tipo de producto demandado.

Se suele hablar de “intención” para saber lo que el usuario quiere, y para ello se entrena a la herramienta.

Se habla de “entidades” para indicar los distintos valores de producto demandado por el usuario (ejemplo, tipo de fruta en una frutería). Para ello, será necesaria una base de datos con los distintos valores que puede tomar el producto (pera, manzana, naranja, etc).

Se suele hablar de “flujo de diálogo” para indicar lo que va guiando el mismo, en función de los datos que introduce el cliente y de la intención y entidades que se descubran en función de dichos inputs del usuario.

A cada intención detectada se le asigna un nivel de confianza en función de la probabilidad de que se haya detectado correctamente la petición concreta del usuario. Si la probabilidad calculada es baja se le hará otra pregunta al usuario para mejorar dicha confianza.

Se definen unos umbrales de confianza para pasar a la siguiente fase o pregunta (fruta concreta demandada, por ejemplo). El objetivo es dar respuestas incorrectas al usuario, contestando algo que no pide.

Se debe entrenar mucho al asistente para llegar a un nivel de confianza óptimo (80 %, por ejemplo) y que dicho asistente sea eficaz y útil al cliente.

Cuando el umbral de confianza sea bajo, se le pedirá más información al usuario indicando, por ejemplo, “no te he entendido, ¿puedes repetir la pregunta?”. Se trata de intentar que el usuario formule la petición de otro modo que sea más comprehensible para el Chatbot. Durante la interacción con el usuario en dicho flujo de diálogo pueden ser necesarias varias preguntas como por ejemplo tipo de fruta, cuántos kilos, si necesita bolsa, etc. A veces, se hace un resumen del pedido al final, para que el cliente confirme que toda la información es correcta.

Muchas veces los chatbots incluyen una pequeña encuesta al final para

saber si el usuario ha quedado satisfecho. Si no ha quedado contento se le pide información y esto ayudará a los desarrolladores a mejorar dicho asistente.

Meter herramientas de creacion de chatbots

2.4.6. Análisis de sentimiento

El análisis de sentimiento⁶ es un método para identificar las emociones que se esconden tras un mensaje concreto y forma parte del procesamiento de lenguaje natural (PLN). Consiste en analizar las frases para extraer de ellas las opiniones o sentimientos acerca de un tema o producto.

Con este análisis⁷ se pretende determinar quién es el sujeto del sentimiento, sobre qué o quién tiene ese sentimiento y categorizar esos sentimientos como positivos, negativos o neutros.

Tras realizar el análisis del sentimiento se puede averiguar qué se esconde detrás de información subjetiva.

Una de las muchas aplicaciones de esta tecnología está enfocada al marketing, de forma que permite a las empresas averiguar qué es lo que quieren sus consumidores mediante el escrutinio de opiniones en redes sociales u otros medios.

Estos sistemas tienen limitaciones, ya que no pueden detectar toda la complejidad del lenguaje humano. Se encuentran problemas a la hora de comprender el contexto en el que se encuentra un texto o para entender la ironía o el sarcasmo.

Las principales herramientas para el análisis del sentimiento son:

- Lingmotif⁸ → Se trata de una herramienta de análisis de sentimiento desarrollada por la universidad de Málaga. Permite obtener valores precisos de las opiniones y sentimientos dentro de un texto.
- Opinion Finder⁹ → Sistema desarrollado por investigadores de la Universidad de Pittsburgh, Cornell y Utah. Permite identificar la subjetividad de frases y varios aspectos de la subjetividad dentro de las propias frases mediante el procesamiento de documentos. Funciona en Inglés.
- LIWC¹⁰ → “Linguistic Inquiry and Word Count” es un programa capaz de analizar textos para calcular el uso que hacen las personas de distintas categorías de palabras. Permite saber si los emisores transmiten un mensaje con palabras positivas o negativas entre otras muchas opciones.

⁶<https://gaeapeople.com/marketing-estrategia/sentiment-analysis>

⁷<https://blog.pangeanic.es/funcionamiento-herramientas-analisis-sentimiento-basadas-en-inteligencia-artificial>

⁸<https://ltl.uma.es>

⁹<https://mpqa.cs.pitt.edu/opinionfinder/>

¹⁰<https://www.liwc.app>

Capítulo 3

Arquitectura del Chatbot

Los sistemas de análisis automático de texto tienen como objetivo entender la información no estructurada hablada por los humanos y convertirla en información estructurada como podría ser un resumen del contenido o la clasificación de un documento por temática. En este caso, el objetivo del chatbot es conseguir esta información estructurada a partir de la mayor cantidad de recuerdos posibles para ayudar a los terapeutas a construir la historia de vida de un paciente que es mucho más complicada de tratar en bruto. El análisis se va a realizar en varios sentidos, por un lado, se clasificarán los recuerdos en etapas de vida, para estructurar la información lo máximo posible. Por otro lado, se determinará si el texto contiene connotaciones positivas o negativas, de tal forma que en la terapia se puedan evitar los recuerdos negativos que pueden agravar la situación del paciente. Por último, se utilizará el procesamiento del lenguaje natural mediante el análisis morfológico de las palabras para comprender lo que quiere decir el paciente y poder generar la respuesta más acertada y relacionada con la temática que se está tratando.

3.1. Clasificación de los recuerdos

La clasificación de los recuerdos en positivos y negativos y la clasificación en etapas de vida se consiguen utilizando el mismo método. Se ha utilizado el componente “TextCategorizer”¹ de la librería spacy de python. Se trata de entrenar un modelo para saber identificar si un recuerdo es positivo o negativo o para poder clasificar el recuerdo en diferentes etapas de la vida. Este método de clasificación se ha probado primero para el análisis de sentimiento de los recuerdos y es por eso que se empezará a explicar cómo funciona el categorizador para el *sentiment analysis*. Más adelante en esta sección se explicará cómo se adaptó para la clasificación en etapas.

¹<https://spacy.io/api/textcategorizer>

Para conseguir que funcione el categorizador de textos se ha utilizado como guía el código del artículo *How to Train Text Classification Model in spaCy*² para probar cómo se podría entrenar el modelo para que supiese diferenciar los recuerdos negativos de los positivos y así tener la primera clasificación que pidieron los terapeutas especializados del proyecto CAN-TOR. Antes de probar con recuerdos se probó con un dataset muy grande de reseñas de moda que se ponía como ejemplo en el artículo del que hablábamos anteriormente. Después, se ha probado usando una batería menor de recuerdos positivos encontrados entre los archivos del TFG de Laura Castillo (rem). También había que incluir recuerdos negativos y, como no se ha encontrado ningún dataset ni textos relacionados, se han usado frases negativas inventadas pensadas para entrenar el modelo de la mejor manera posible.

El análisis de sentimiento de los recuerdos funciona siguiendo una serie de pasos. En primer lugar, se añade el componente TextCategorizer (textcat) a un modelo en blanco del idioma español. Un modelo en blanco es un modelo que no tiene ningún componente de spaCy definido (como serían NER que explicaré más adelante), es decir, el texto que se almacena en los documentos de spaCy no es analizado por ninguna cadena de procesos porque la tubería de componentes está vacía. Si hubiésemos cogido un modelo pre-entrenado como “es_core_news_sm”, el clasificador de textos se sumaría al trabajo previo de los procesos que analizan el texto como serían [‘tok2vec’, ‘morphologizer’, ‘parser’, ‘attribute_ruler’, ‘lemmatizer’, ‘ner’]. Por ejemplo, el componente NER, más reconocido como *Named Entity Recognizer*, reconoce entidades dentro del texto como nombres de personas, fechas o lugares. Al usar el modelo en blanco empezamos con el español de cero, sin analizar. Tras añadir el componente textcat, al crear un documento de spaCy, automáticamente, pasará por el proceso de clasificación que le indiquemos. Para que textcat funcione como categorizador de recuerdos, se le añade a nuestro nuevo componente dos etiquetas, NEGATIVO y POSITIVO, que definirán cuánto de negativo es un recuerdo y cuánto de positivo. Para seguir configurándolo, el prototipo coge el texto que usaremos para entrenar nuestro modelo y lo prepara adecuándolo al formato que entiende el clasificador, siendo éste una lista de tuplas (texto, etiqueta). Además, cogerá un porcentaje de esta batería de recuerdos (% que previamente hemos definido) y lo reservará para la evaluación de las predicciones, es decir, para analizar cómo de bien predice nuestro modelo tras entrenarlo.

Ya tenemos casi todo preparado para comenzar a entrenar el modelo, se dividen los casos de entrenamiento en lotes que se analizarán y evaluarán un número definido de iteraciones para asegurar que el entrenamiento es lo más preciso posible sin pasarnos de vueltas para que sea óptimo. Se trata de un proceso iterativo en el que las predicciones del modelo se comparan con las etiquetas de referencia para estimar el gradiente de la pérdida. El modelo se

²<https://www.machinelearningplus.com/nlp/custom-text-classification-spacy/>

entrena utilizando una función que lo analiza y actualiza en cada iteración, la función `update()`. También se comprueban las predicciones del modelo en cada vuelta, se comparan con las etiquetas de referencia para estimar la desviación de la pérdida.

Una vez afinado el modelo mediante el entreno previo, ya podemos ponerlo a prueba. El prototipo sacará la probabilidad de que un texto procesado por spacy sea un recuerdo positivo y la probabilidad de que sea uno negativo. En lo relativo al chatbot, se ha añadido se ha añadido el archivo “analyze_answer” que alberga esta funcionalidad de análisis de sentimientos que, como se ha explicado, entrena primero al modelo con textos positivos y negativos y luego evalúa el texto que le llega del usuario e identifica si es negativo o positivo. Esto se utilizará como una de las categorías que se le asignan a cada respuesta recibida y que se mete en una base de datos junto a la respuesta que haya dado el usuario al chatbot. Este análisis de sentimiento también servirá para identificar temáticas dolorosas para el interrogado que no deberían ser usadas en las terapias.

Más adelante en el proyecto, se observó que, al analizar el sentimiento de la frase, había recuerdos o simplemente, datos que no encajaban en ninguna de las dos categorías de positivo y negativo. Las emociones neutras son aquellas que no son desagradables ni agradables, es decir, ni negativas ni positivas. Es por esto que se probó a añadir una nueva categoría para los datos neutro como por ejemplo, indicar tu edad, explicar dónde vivías en cierto momento de tu vida, etc. No suponía un gran esfuerzo porque el programa estaba pensado para poder añadir todas las categorías que se considerasen. Sin embargo, a niveles prácticos, se pudo comprobar que no funcionaba igual de bien que la clasificación binaria. El nivel de entrenamiento que necesitaría el modelo para poder distinguir también las frases neutras es mucho mayor, es decir, se necesitarían muchos más casos, ejemplos, para que el modelo aprendiese de ellos. En nuestro caso, solo se añadieron alrededor de 40 recuerdos neutros, de nuevo inventados, por la falta de recursos encontrados en Internet que no proporcionaba ningún dataset parecido al que se buscaba. En la siguiente sección de pruebas se podrá comprobar la diferencia entre meter la categoría neutra y no meterla.

3.1.1. Pruebas del análisis de sentimiento

Para comprobar la precisión y el correcto funcionamiento del módulo de análisis de sentimiento, se hicieron bastantes pruebas que corroboraron que la clasificación en recuerdos negativos y positivos funcionaba bastante bien pero que podría mejorarse metiendo muchos más casos de entrenamiento para afinar el modelo. A continuación se muestran algunos ejemplos de frases que se han analizado con el categorizador de textos y las conclusiones que se han sacado. Entre llaves se muestra la probabilidad sobre 1 de que la frase sea positiva y la probabilidad sobre 1 de que sea negativa. Entre corche-

tes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad las dos:

- “Mi abuela se murió en 1998”

```
{'POSITIVO': 0.09926079958677292, 'NEGATIVO': 0.9007392525672913}
['negativo']
```

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'POSITIVO': 0.992607593536377, 'NEGATIVO': 0.007392475381493568}
['positivo']
```

- “Tengo 23 años”

```
{'POSITIVO': 0.9999983310699463, 'NEGATIVO': 1.6568293403906864e-06}
['positivo']
```

- “Me dolío muchísimo cuando me rompé una pierna”

```
{'POSITIVO': 2.0322097043390386e-05, 'NEGATIVO': 0.9999797344207764}
['negativo']
```

- “Mi hermano y yo nos pasabamos las tardes haciendo puzzles”

```
{'POSITIVO': 0.9681606888771057, 'NEGATIVO': 0.031839337199926376}
['positivo']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'POSITIVO': 0.9882893562316895, 'NEGATIVO': 0.011710633523762226}
['positivo']
```

- “Mi pareja sufrió depresión después del parto”

```
{'POSITIVO': 0.07562904059886932, 'NEGATIVO': 0.9243709444999695}
['negativo']
```

Por otro lado, también se probó cómo funcionaba el programa añadiendo la clasificación de neutro. Podemos comparar así ambas formas de clasificación y comprobar que, efectivamente, funciona mejor el binario. De nuevo, entre llaves se muestra la probabilidad sobre 1 de que la frase sea positiva, la probabilidad sobre 1 de que sea negativa y la probabilidad sobre 1 de que sea neutra. Entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad de todas:

- “Mi abuela se murió en 1998”

```
{'POSITIVO': 0.000133998051751405, 'NEGATIVO': 0.00039583168108947575,  
'NEUTRO': 0.9994701743125916}  
['neutro']
```

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'POSITIVO': 0.0003367967437952757, 'NEGATIVO': 0.9894788861274719,  
'NEUTRO': 0.010184396989643574}  
['negativo']
```

- “Tengo 23 años”

```
{'POSITIVO': 0.016445694491267204, 'NEGATIVO': 0.0031412760727107525,  
'NEUTRO': 0.980413019657135}  
['neutro']
```

- “Me dolió muchísimo cuando me rompí una pierna”

```
{'POSITIVO': 0.0006958534941077232, 'NEGATIVO': 0.9989858269691467,  
'NEUTRO': 0.0003183614171575755}  
['negativo']
```

- “Mi hermano y yo nos pasabamos las tardes haciendo puzzles”

```
{'POSITIVO': 2.354631942580454e-05, 'NEGATIVO': 0.9972598552703857,  
'NEUTRO': 0.0027166458312422037}  
['negativo']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'POSITIVO': 0.051713500171899796, 'NEGATIVO': 0.9064642786979675,  
'NEUTRO': 0.041822321712970734}  
['negativo']
```

- “Mi pareja sufrió depresión después del parto”

```
{'POSITIVO': 2.2260337573243305e-05, 'NEGATIVO': 0.9954761862754822,  
'NEUTRO': 0.004501543939113617}  
['negativo']
```

3.1.2. Clasificación en etapas de vida

Otra de las categorías que se añadirá junto a la respuesta en la base de datos de recuerdos será la de la etapa de vida a la que corresponde el recuerdo. Primero había que definir esas etapas y para ello, en un principio se utilizó de forma orientativa la clasificación que hace el Ministerio de Salud de Colombia ?? porque no se encontraron clasificaciones tan claras para España y parecía una forma sensata de clasificación. Como en el caso del análisis de sentimiento, se han probado varias combinaciones de Le he hecho algunas modificaciones para adecuarla más a lo que quería sacar en claro de la información que se me presentaba. Las etapas serían las siguientes:

Otra de las categorías que se añadirá junto a la respuesta en la base de datos de recuerdos será la de la etapa de vida a la que corresponde el recuerdo. En un principio, había que definir las etapas y se decidió usar de forma orientativa la clasificación que hace el Ministerio de Salud de Colombia ?? porque no se encontraron fuentes fiables españolas que diesen una clasificación tan clara y sensata. Aún así se le hicieron algunas adecuaciones para que se ajustase más a lo que se buscaba y lo que se quería sacara en claro. Además, como en el caso del análisis de sentimiento con el neutro, se decidió desde un principio añadir la clasificación “Indeterminado” para aquellos recuerdos que no encajasen en ninguna etapa porque fuesen datos generales. Aquí se veía más claro que había información de los usuarios que era muy general y que por defecto no podía meterse en ninguna otra etapa. Sin embargo, pasaba lo mismo que con el neutro, la clasificación era mucho menos acertada. Más adelante se explicará con más detalle. A continuación se muestra la clasificación inicial que se eligió:

- Infancia de 0 a 11 años
- Adolescencia de 12 a 17 años
- Juventud de 18 a 26 años
- Etapa adulta de 27 a 59 años
- Vejez de 60 años o más
- Indeterminado para aquellos textos en los que no se pueda distinguir la etapa

Una vez elegidos los períodos de tiempo en los que se divide la vida de las personas, el objetivo era clasificar recuerdos, dados en forma de respuesta a una pregunta, según la etapa de la vida a la que pertenecía el recuerdo de la persona interrogada. Para ello, también se ha utilizado la misma tecnología que para el análisis de sentimientos. Además de entrenarse en recuerdos positivos y negativos, ahora el modelo se entrena para distinguir etapas vitales

en las que ocurren los recuerdos para así, de cara a la elaboración de un libro de vida, toda la información esté bien estructurada en períodos de tiempo.

Como se explicaba al principio, la clasificación que se eligió inicialmente no resultó dar buenos resultados. El nivel de entrenamiento que necesitaría el modelo para poder distinguir también los recuerdos que no encajan en ninguna de las etapas anteriores es mucho mayor. Además, resultaba muy ambigua la etapa indeterminada porque los recuerdos que se meten en esa clasificación no tienen relación entre ellos para el modelo, no puede aprender de similitudes porque cada recuerdo es muy distinto y muy general. Algo parecido pasaba con la etapa de la adolescencia, no se aprecia bien en los ejemplos la diferencia entre la etapa de la adolescencia y la de la juventud porque no existen ejemplos que hagan una buena distinción entre la una y la otra ya que son muy parecidos los sucesos que podrían ocurrir en una y en la otra. Es por eso que se ha decidido quitar ambas etapas y ponérselo mucho más fácil al modelo para que pueda reconocer bien los recuerdos y no confundirse tan fácilmente. En la sección de pruebas se ve claramente la diferencia entre ambos casos. Finalmente, se ha elegido una clasificación muy clara según los hechos que suelen pasar en la vida de las personas en las distintas etapas y para los que cada etapa tiene ejemplos precisos y nada ambiguos. Se hace la siguiente división:

- Infancia de 0 a 12 años
- Juventud de 13 a 26 años
- Etapa adulta de 27 a 59 años
- Vejez de 60 años o más

3.1.2.1. Pruebas

Se comprueba la precisión y el correcto funcionamiento del módulo de clasificación en etapas de vida, se hicieron bastantes pruebas que indicaron que la clasificación funcionaba bastante bien pero que podría mejorarse metiendo muchos más casos de entrenamiento para afinar el modelo. La clasificación definitiva como se explicaba anteriormente era la que distinguía las etapas de infancia, juventud, etapa adulta y vejez. A continuación se muestran algunos ejemplos de frases que se han analizado con el categorizador de textos y las conclusiones que se han sacado. Entre llaves se muestra la probabilidad sobre 1 de que la frase se corresponda con la fase de la infancia, la probabilidad sobre 1 de que se corresponda con la fase de la juventud, la probabilidad sobre 1 de que se corresponda con la fase de la etapa adulta y la probabilidad sobre 1 de que se corresponda con la fase de la vejez. Entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad de todas:

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'INFANCIA': 0.003423456335440278, 'JUVENTUD': 0.004028527531772852,  
'ETAPA ADULTA': 0.979805052280426, 'VEJEZ': 0.012742995284497738}  
['etapa adulta']
```
- “Me dolió muchísimo cuando me rompí una pierna a los 22 años”

```
{'INFANCIA': 0.07795873284339905, 'JUVENTUD': 0.7799875736236572,  
'ETAPA ADULTA': 0.07531660795211792, 'VEJEZ': 0.0667370930314064}  
['juventud']
```
- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”

```
{'INFANCIA': 0.9991468191146851, 'JUVENTUD': 0.00011972746870014817,  
'ETAPA ADULTA': 0.00013617362128570676, 'VEJEZ': 0.0005973773077130318}  
['infancia']
```
- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'INFANCIA': 0.9864945411682129, 'JUVENTUD': 0.0020768800750374794,  
'ETAPA ADULTA': 0.005027524195611477, 'VEJEZ': 0.0064010825008153915}  
['infancia']
```
- “Mi pareja sufrió depresión después del parto”

```
{'INFANCIA': 0.10760761052370071, 'JUVENTUD': 0.003418843261897564,  
'ETAPA ADULTA': 0.8073434829711914, 'VEJEZ': 0.08163014054298401}  
['etapa adulta']
```
- “Mi tía siempre se dedicó a la pintura”

```
{'INFANCIA': 0.15463659167289734, 'JUVENTUD': 0.2145996242761612,  
'ETAPA ADULTA': 0.5623230338096619, 'VEJEZ': 0.0684407576918602}  
['etapa adulta']
```
- “Cuando estudiaba en el instituto teníamos tres gatos”

```
{'INFANCIA': 0.07519558817148209, 'JUVENTUD': 0.6477565169334412,  
'ETAPA ADULTA': 0.23660382628440857, 'VEJEZ': 0.040444087237119675}  
['juventud']
```

También se probó cómo funcionaba el modelo en un principio cuando en la clasificación estaban las etapas de adolescencia y la indeterminada. Se puede comparar así ambas formas de clasificación y comprobar que, efectivamente, funciona mejor la que tiene menos categorías y mucho más diferenciadas. De nuevo, entre llaves se muestra la probabilidad sobre 1 de cada una de las fases y, entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad de todas:

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'INFANCIA': 0.11781484633684158, 'ADOLESCENCIA': 0.29379358887672424,  
'JUVENTUD': 0.22842659056186676, 'ETAPA ADULTA': 0.20501598715782166,  
'VEJEZ': 0.10608396679162979, 'INDETERMINADO': 0.04886503145098686}  
['adolescencia']
```

- “Me dolió muchísimo cuando me rompí una pierna a los 22 años”

```
{'INFANCIA': 0.0013075786409899592, 'ADOLESCENCIA': 0.0039981659501791,  
'JUVENTUD': 0.9936805963516235, 'ETAPA ADULTA': 0.0006605012458749115,  
'VEJEZ': 0.00015935904229991138, 'INDETERMINADO': 0.00019375460396986455}  
['juventud']
```

- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”

```
{'INFANCIA': 0.9913138747215271, 'ADOLESCENCIA': 0.0026805144734680653,  
'JUVENTUD': 0.0005102856084704399, 'ETAPA ADULTA': 0.0003391568607185036,  
'VEJEZ': 0.00324622611515224, 'INDETERMINADO': 0.0019098836928606033}  
['infancia']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'INFANCIA': 0.9902841448783875, 'ADOLESCENCIA': 0.0011972723295912147,  
'JUVENTUD': 0.007258791010826826, 'ETAPA ADULTA': 0.00017597108671907336,  
'VEJEZ': 0.0006049419171176851, 'INDETERMINADO': 0.00047876848839223385}  
['infancia']
```

- “Mi pareja sufrió depresión después del parto”

```
{'INFANCIA': 0.9323758482933044, 'ADOLESCENCIA': 0.011906568892300129,  
'JUVENTUD': 0.018807733431458473, 'ETAPA ADULTA': 0.008572818711400032,  
'VEJEZ': 0.025070885196328163, 'INDETERMINADO': 0.0032660840079188347}  
['infancia']
```

- “Mi tía siempre se dedicó a la pintura”

```
{'INFANCIA': 7.487166294595227e-05, 'ADOLESCENCIA': 0.00675414502620697,
'JUVENTUD': 0.0021927112247794867, 'ETAPA ADULTA': 0.9876710176467896,
'VEJEZ': 0.0032006383407860994, 'INDETERMINADO': 0.00010662782733561471}
['etapa adulta']
```

- “Cuando estudiaba en el instituto teníamos tres gatos”

```
{'INFANCIA': 0.9774642586708069, 'ADOLESCENCIA': 0.005647959187626839,
'JUVENTUD': 0.012261644005775452, 'ETAPA ADULTA': 0.0004069636052008718,
'VEJEZ': 0.0033992205280810595, 'INDETERMINADO': 0.0008198729483410716}
['infancia']
```

3.2. Procesamiento del texto para encadenar preguntas y respuestas

Esta sección consiste en conseguir que las preguntas que se hagan al interrogado tengan sentido con el resto de la conversación previa. Conseguir que sean lo más inteligentes y adecuadas posible.

Lo primero que se ha hecho para encontrar la siguiente pregunta a hacer es coger la lista de todas las posibles preguntas y eliminar aquellas que ya han sido contestadas anteriormente para no repetirlas.

Lo siguiente que se hará es pasar tanto la respuesta más reciente que ha dado el interrogado como todas las posibles preguntas por un proceso de síntesis. Para ello vamos a utilizar el analizador de texto Spacy utilizando el código del TFG de Laura Castilla Castellano (Generación de historias a partir de una base de conocimiento), en concreto el código que aparece en el archivo “analysis.py” de su código, que contiene funciones para el procesamiento y síntesis de los textos. Los pasos que Laura sigue para el análisis del texto aparecen en el apartado 5.1 (Analizar textos para obtener sugerencias) de su memoria, páginas 30 y 31. Las funciones que se han reutilizado en este trabajo, con algunas modificaciones, son las siguientes:

- Read: Coge el texto a analizar y elimina signos de puntuación
- Synthesis: Elimina las palabras vacías como las preposiciones del texto
- Lemmatize: Transforma cada palabra del texto sus lemas correspondientes
- Categorize: Separa las palabras en sustantivos, verbos y adjetivos. Además, coge los 30 más comunes de cada tipo
- Get_most_common_words: Coge las palabras más comunes de todos los tipos del texto

Estas funciones se han utilizado para conseguir una representación del texto más precisa y reducida. Solo los lemas de las palabras importantes del texto serán usados para este módulo de encadenamiento de preguntas y respuestas.

Por último, dada la respuesta y todas las posibles preguntas ya reducidas a sus lemas, para encontrar la pregunta más adecuada, se compara una a una los lemas de las posibles preguntas con los lemas de la respuesta. Dentro de la lista de posibles preguntas, la que más lemas comunes tenga con la respuesta, gana como siguiente pregunta a formular.

3.2.1. Pruebas

Veamos algunos ejemplos de cómo funciona el primer acercamiento a un chatbot inteligente que pregunta con algo de sentido. A continuación se muestra la primera pregunta y la respuesta del interrogado:

IA: ¿Quiénes son las personas más importantes de tu vida?

Tú: Mi familia y concretamente mis padres y mi hermano

En el siguiente texto de salida por consola se muestra el análisis que hace el módulo de cálculo de la mejor siguiente pregunta:

Maxima coincidencia de lemas hasta ahora: 0

Pregunta elegida hasta ahora: ¿Cuál es tu sexo?

Possible pregunta: ¿Hay algún momento que te gustaría volver a vivir?
Lemas de la posible pregunta: {'volver', 'vivir', 'momento', 'gustar'}
Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
Lemas que coinciden: 0

Maxima coincidencia de lemas hasta ahora: 0

Pregunta elegida hasta ahora: ¿Cuál es tu sexo?

Possible pregunta: ¿Dónde viviste cuando eras pequeño?
Lemas de la posible pregunta: {'pequeño', 'vivistar'}
Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
Lemas que coinciden: 0

Maxima coincidencia de lemas hasta ahora: 0

Pregunta elegida hasta ahora: ¿Cuál es tu sexo?

Possible pregunta: ¿Has vivido en algún lugar diferente cuando eras pequeño?
Lemas de la posible pregunta: {'lugar', 'pequeño', 'vivir'}

Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
 Lemas que coinciden: 0

 Maxima coincidencia de lemas hasta ahora: 0
 Pregunta elegida hasta ahora: ¿Cuál es tu sexo?

Possible pregunta: ¿Cómo se llaman tus padres?
 Lemas de la posible pregunta: {'padre', 'llamar'}
 Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
 Lemas que coinciden: 1

 Maxima coincidencia de lemas hasta ahora: 1
 Pregunta elegida hasta ahora: ¿Cómo se llaman tus padres?

Possible pregunta: ¿Si tienes hermanos, cómo se llaman?
 Lemas de la posible pregunta: {'llamar', 'tener', 'hermano'}
 Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
 Lemas que coinciden: 1

 Maxima coincidencia de lemas hasta ahora: 1
 Pregunta elegida hasta ahora: ¿Cómo se llaman tus padres?

Possible pregunta: ¿Cómo era la casa dónde viviste de pequeño?
 Lemas de la posible pregunta: {'pequeño', 'casa', 'vivistar'}
 Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
 Lemas que coinciden: 0

En el texto podemos observar el análisis de varias posibles preguntas. La separación entre preguntas se marca con una fila de guiones. En cada apartado nos encontramos con el número de lemas que han coincidido en preguntas anteriores, después con la pregunta que hasta el momento va ganando como candidata a ser preguntada, con la posible pregunta, con los lemas más relevantes de la posible pregunta entre llaves y con los lemas de la respuesta entre llaves. Las tres primeras preguntas no serán elegidas como candidatas a ser preguntadas porque no coincide ningún lema de la respuesta con los lemas de la posible pregunta, es decir, no mejora el número de coincidencias. Cuando llega a la cuarta pregunta, “¿Cómo se llaman tus padres?” va a encontrar una coincidencia de tal forma:

Lemas de la respuesta:	{hermano, familia, padre}
Lemas de la posible pregunta:	{llamar, padre}

Como podemos observar el lema “padre” coincide en ambas, es decir, en esta posible pregunta encontramos una coincidencia. Como el número de

coincidencias es mejor que 0, la pregunta candidata “¿Cuál ha sido el momento más feliz de tu vida?” se sustituye por la pregunta “¿Cómo se llaman tus padres?”. Es por esto que cuando analizamos el resto de posibles preguntas, no encontramos ninguna que sea mejor porque no superan el número de coincidencias en lemas. En conclusión, esta será la siguiente pregunta que se le plantee al usuario, la cual tiene relación con la que se había hecho anteriormente.

3.3. Base de datos MongoDB

Aplicación Web

La aplicación web se ha creado con el objetivo de proporcionar funcionalidades tanto para los posibles terapeutas como para sus posibles pacientes. Por un lado, el chatbot se ha pensado para que lo usen los pacientes y es el foco principal de este TFG y, donde se han aplicado todas las técnicas que se han tratado en las secciones anteriores. La aplicación web permite probar el chatbot de forma mucho más manejable y visible que desde un terminal del entorno de desarrollo. Por otro lado, se ha querido añadir las funcionalidades para los supuestos terapeutas que, podrían acceder a los recuerdos de sus pacientes de forma sencilla y visual.

4.1. Prototipo inicial

Al inicio de este trabajo, había que ver qué dirección se iba a seguir, en esos momentos yo planteaba a mis tutores diferentes ideas que se me iban ocurriendo y ellos me guiaban por la propuesta más adecuada. Una de las ideas que se propusieron era la de este prototipo de aplicación web inicial. Realicé un esbozo sobre lo que creía que iba a necesitar el chatbot como aplicación para estar completo. Este prototipo a mano alzada está dividido en dos partes. Por un lado, están las funcionalidades pensadas para el uso del terapeuta, que podrá programar las sesiones que quiere hacer personalmente con el paciente (ver Figuras 4.2, 4.3, 4.4, 4.5, 4.6). Por otro lado, están las funcionalidades pensadas para el paciente con demencia (ver Figura 4.1) y que hará uso del chatbot en sí. Es importante aclarar que el terapeuta tiene sus sesiones con el paciente como se hace habitualmente y las funcionalidades que tiene disponibles en la aplicación son para ayudarle a programar esas sesiones. El chatbot es solo accesible desde la cuenta de un paciente porque es una herramienta de apoyo para el terapeuta para facilitar la recolección de recuerdos del paciente y, conseguir la información de otra forma para así ganar tiempo.

Para el usuario con demencia el prototipo es muy sencillo, tiene 2 funcionalidades básicas propias (ver Figura 4.1):

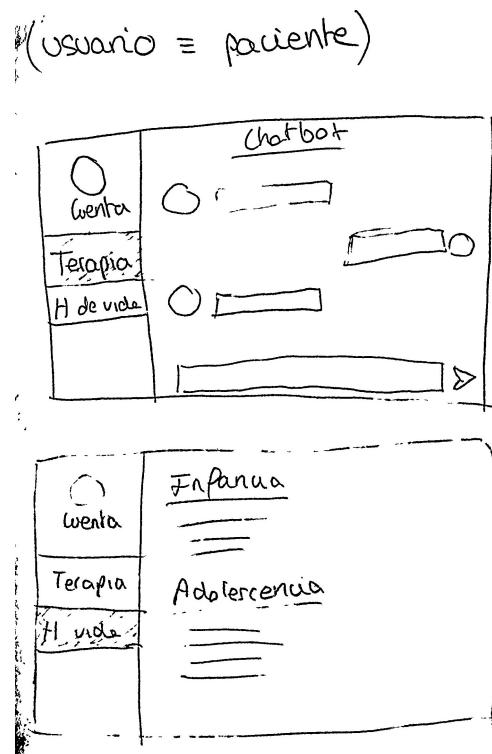


Figura 4.1: Prototipo inicial de la aplicación para el usuario con demencia

- Chatbot: La pestaña de terapia lleva al usuario al chat donde se le hará preguntas sobre su vida para recabar el máximo número de recuerdos. El chatbot es un complemento aparte que ayuda al terapeuta a recoger la información del usuario pero que no sustituye las sesiones habituales paciente-terapeuta.
- Historia de vida: Los recuerdos recabados se guardan en la aplicación y el usuario puede consultarlos divididos por etapas de la vida (infancia, adolescencia, etapa adulta, etc.). Esto sería parte de la terapia ocupacional basada en reminiscencia y un paso previo para construir el libro de vida del usuario. El usuario puede revisitar sus recuerdos a través de la aplicación y esto puede ayudar en el retraso del deterioro cognitivo.

En cuanto a las vistas del terapeuta, tendrá muchas más funcionalidades disponibles:

Ver Figura 4.2 →

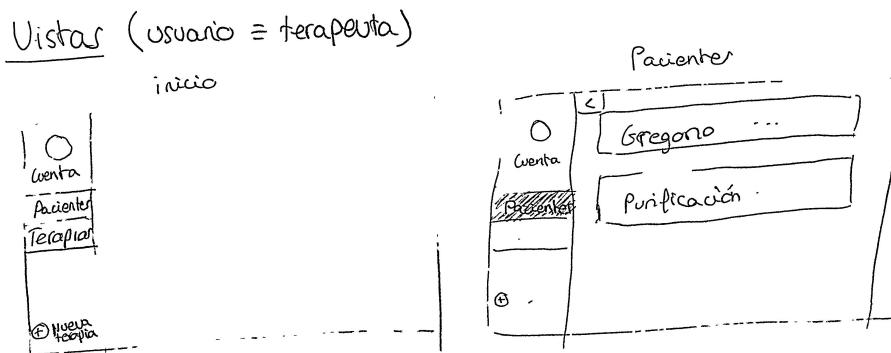


Figura 4.2: Prototipo inicial de la aplicación para el terapeuta: Consultar los pacientes registrados

- Pacientes: Se muestra una lista de los usuarios (pacientes) que tiene el terapeuta asignados y que están registrados en la aplicación.

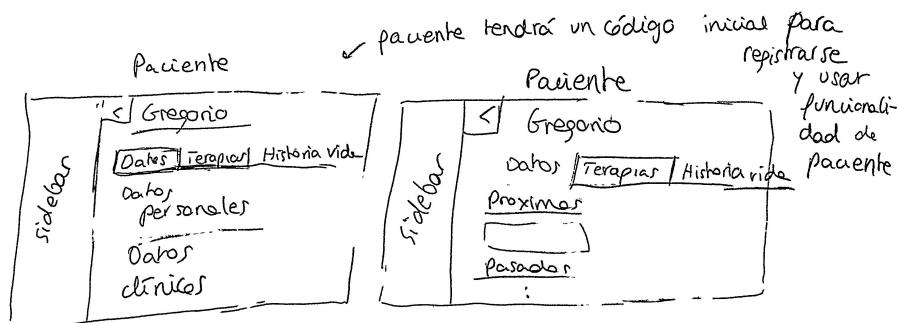


Figura 4.3: Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 1

Ver Figura 4.3 →

- Datos personales del paciente: Los datos personales y clínicos del paciente estarán disponibles para que el terapeuta los consulte cuando quiera.
- Terapias del paciente: Se muestran las terapias que se han creado para ese paciente, tanto las que se programaron para fechas pasadas como las que ocurrirán próximamente. El terapeuta es el que crea estas terapias para sus futuras sesiones con el paciente, las terapias pasadas son las sesiones que se han finalizado y las próximas son las que están pendientes por hacer.

Ver Figura 4.4 →

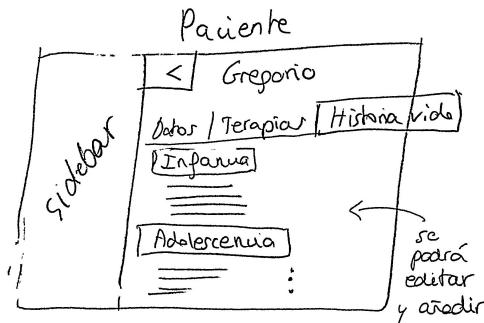


Figura 4.4: Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 2

- Historia de vida del paciente: Los recuerdos recabados en las sesiones terapéuticas se guardan en la aplicación y el terapeuta puede consultarlos divididos por etapas de la vida (infancia, adolescencia, etapa adulta, etc.) igual que podía hacer el usuario. El objetivo de que el terapeuta pueda consultarlos es para revisar que se han grabado los recuerdos correctamente y para comprobar que son precisos y coherentes. En cualquier caso, el terapeuta podrá editar o borrar los recuerdos que crea incompletos o incorrectos y podrá también añadir recuerdos que haya conseguido de alguna otra forma. En caso de que algún recuerdo esté mal clasificado en la etapa de la vida en que ocurrió, también se podría mover a su sitio correcto.

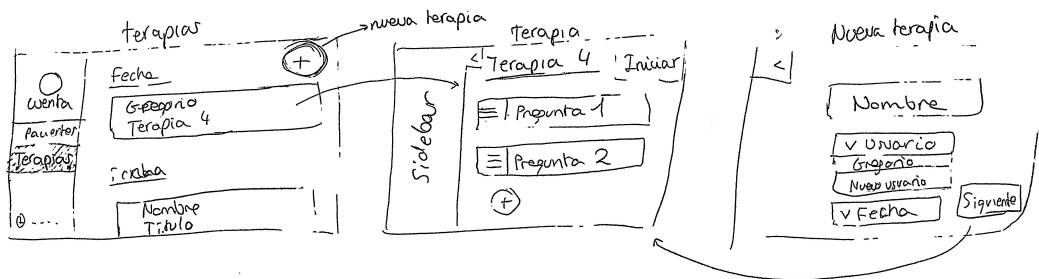


Figura 4.5: Prototipo inicial de la aplicación para el terapeuta: Crear y consultar las terapias creadas

Ver Figura 4.5 →

- Terapias: Se muestra una lista ordenada de las terapias que ha ido creando el terapeuta para los distintos pacientes.

- Consultar terapia: El terapeuta puede revisar una terapia que haya sido ya planificada. También puede editarla o iniciar la terapia cuando esté con el paciente.
- Nueva terapia: Para crear/programar una nueva terapia es necesario asignarle un nombre, el paciente al que va dirigida y una fecha en la que se llevará a cabo. El paciente se puede elegir de entre los usuarios que tiene asignados el terapeuta pero, también puede crear un nuevo usuario al que irá dirigida que no es necesario que esté registrado en la aplicación. Después, se crean por cada bloque las preguntas o temas que se quieren sacar en la sesión en el orden que quieran ser utilizados.

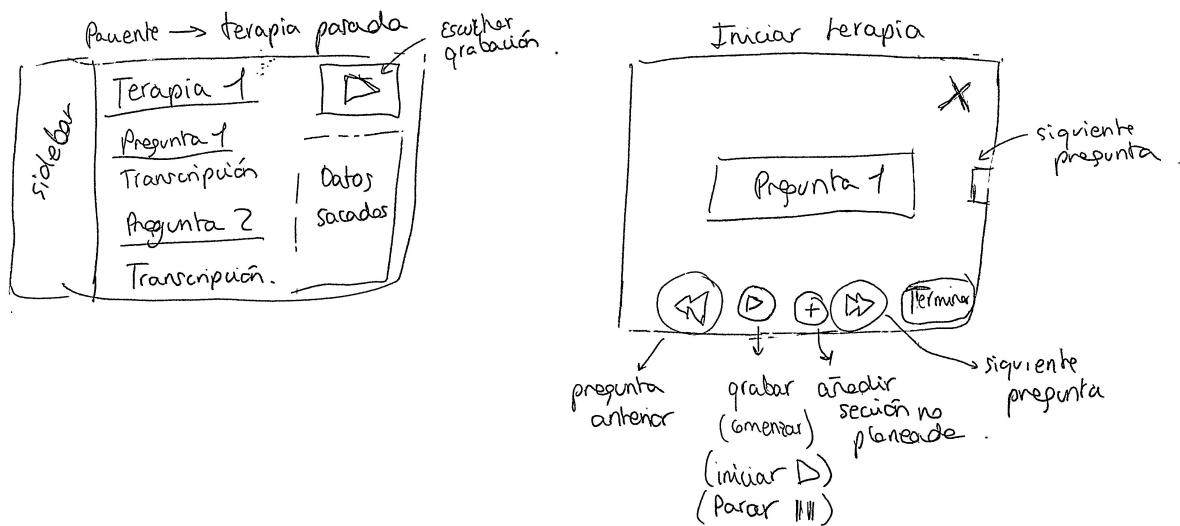


Figura 4.6: Prototipo inicial de la aplicación para el terapeuta: Consultar y reproducir una terapia concreta

Ver Figura 4.6 →

- Terapia finalizada: Se puede consultar una terapia que ya se ha realizado. Se podrá ver toda la información que se ha sacado, apuntes del terapeuta, la grabación de la sesión, la transcripción de la grabación, etc.
- Iniciar terapia: Cuando el terapeuta se reúne con el paciente para la sesión, pulsa a iniciar la terapia y obtendrá en orden los temas y preguntas que le quiere ir sacando al paciente. Podrá ir pasando de pregunta en pregunta cuando lo crea necesario. Al comenzar, puede poner a grabar la sesión para que luego se haga una transcripción automática de todo lo que ambos dicen. Esa grabación se puede pausar o continuar

cuento se quiera. También, se puede añadir una sección nueva que no se contemplase en la planificación pero que se quiera incluir en la terapia. Y finalmente, se puede terminar dar por finalizada la terapia pulsando en terminar.

Este prototipo se descartó porque se centraba mucho en hacer una aplicación web y no tanto en el desarrollo de un chatbot lo suficientemente inteligente para dirigir una sesión de terapia sin necesidad del terapeuta. Con esto, no se desestima la labor del terapeuta que siempre tendría que comprobar que la información recabada por el chatbot es veraz y utilizar esa información para ayudar al usuario con una terapia cara a cara con el usuario. El chatbot es solo una herramienta de apoyo para facilitar el trabajo del terapeuta en conseguir los recuerdos de la vida del usuario.

Por otro lado, en junio de 2022 se presentó el TFG “Recuérdame: Aplicación de apoyo para el tratamiento de personas con problemas de memoria mediante terapias basadas en reminiscencia”, que precisamente se encarga del desarrollo de una aplicación web completa para facilitar la tarea del terapeuta que tiene que ayudar a usuarios con demencia a través de la terapia ocupacional basada en reminiscencia. Es decir, no tenía sentido que yo desarrollase una aplicación web para usuario y terapeuta porque ya se estaba desarrollando una cosa muy parecida en otro trabajo de fin de grado. Es por eso que la recomendación de mis tutores de centrarme en el desarrollo del chatbot fue la acertada. Aun así, sí que se ha desarrollado una pequeña aplicación web para mostrar de forma más visual la funcionalidad del chatbot. El diseño y la implementación de la misma se explicarán en las siguientes secciones.

4.2. Bases de datos

Se han usado dos tipos de bases de datos, una relacional en MySQL encargada de almacenar datos de inicio de sesión y terapias para la página web del chatbot. La otra base de datos es no relacional levantada en MongoDB que almacena los datos relacionados con los recuerdos de la persona con demencia y de la interacción con el chatbot.

4.2.1. MySQL

Se trata de una base de datos ideal para conjuntos de datos que tienen una estructura fija, como en este caso, los datos de inicio de sesión de los usuarios y los datos de las terapias que asocian a los terapeutas con sus pacientes. Es por esto que se ha construido una base de datos con dos tablas fijas y relacionadas, una que identifica a los usuarios y otra que guarda la información sobre las terapias. En la diagrama de entidad-relación de la figura 4.7 se puede ver la relación entre ambas tablas. Se trata de una

relación recursiva ya que la tabla de usuarios se relaciona consigo misma de tal forma que la tabla desempeña un rol de terapeuta en uno de los lados de la relación y un rol de paciente en el otro lado. La tabla de terapias solo cumple la función de intermediaria y en ella se tratan dos tipos de usuarios, terapeutas y pacientes.

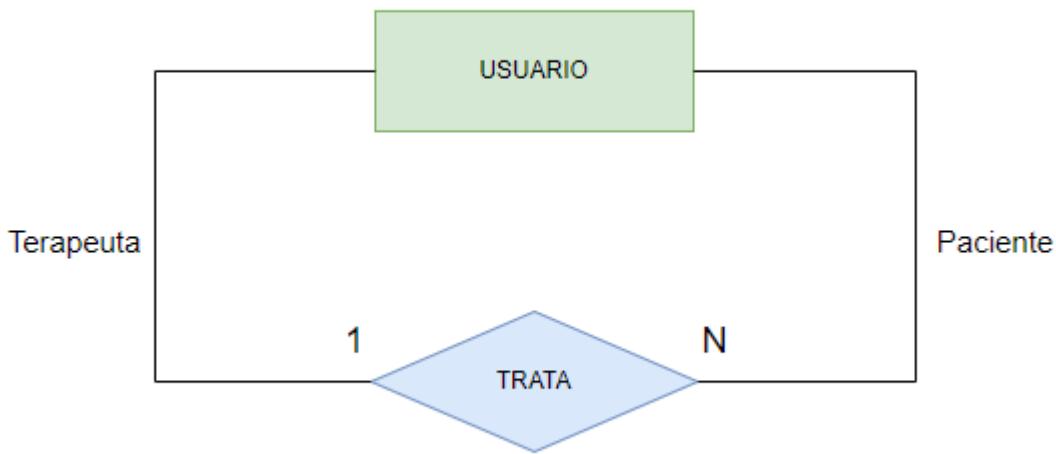


Figura 4.7: Diagrama de entidad-relación MySQL

4.2.1.1. Tabla de usuarios

Es la tabla que alberga toda la información de los usuarios, tanto terapeutas como usuarios que se distinguirán mediante el campo “type”. Son los dos tipos de usuarios que podrán iniciar sesión en la aplicación. Por cada registro tendremos los campos que se muestran en la figura 4.8 y que se explican a continuación:

- *id*: El entero que representa el identificador de la fila, único y clave primaria para poder diferenciar cada registro inequívocamente
- *name*: El nombre de usuario, una cadena de caracteres, que se utilizará para referirse al navegante en la aplicación web
- *email*: El correo electrónico, una cadena de caracteres, único también para que no puedan existir dos usuarios con el mismo email, lo que supondría un error

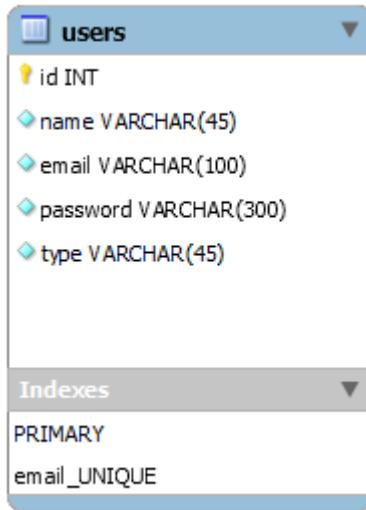


Figura 4.8: Diagrama de la tabla de usuarios

- *password*: La contraseña, cadena de caracteres que se guardará cifrada. Cuando se intente iniciar sesión, se comprobará si los datos son correctos y las contraseñas coinciden.
- *type*: Cadena de caracteres que indica si el tipo de usuario es terapeuta o paciente. Es necesario distinguirles porque tendrán acceso a funcionalidades distintas en la aplicación web.

4.2.1.2. Tabla de terapias

Tabla en la que se guardan las relaciones entre terapeutas y sus pacientes. Por cada registro tenemos los campos que se muestran en la figura 4.9. Las diferentes columnas de la tabla se explican a continuación:

- *id*: El entero que representa el identificador de la fila, único y clave primaria, se utiliza igual que en el de la tabla de usuarios.
- *therapist*: El identificador del terapeuta al que le corresponde la terapia encargado de tratar al paciente que toque.
- *patient*: El identificador del paciente señalado para la terapia
- *code*: El entero que representa un código, único, para que el paciente pueda registrarse como tal en la aplicación y con su terapeuta asignado. Son números elegidos aleatoriamente del 100 al 999. Una vez se registra el paciente, no se vuelve a necesitar este código para nada.

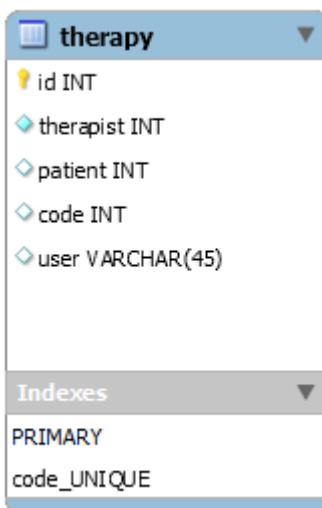


Figura 4.9: Diagrama de la tabla de terapias

- *user*: Cadena de caracteres utilizada para identificar por nombre a un paciente que aún no se ha registrado pero sí tiene código asignado.

4.3. Implementación final

— A nivel de css no me he querido meter en detalles técnicos, porque que se vea bonito no era el objetivo principal de la aplicación sino algo secundario

Para la aplicación del chatbot, con la que el usuario con demencia interactuaría, se ha decidido usar el *framework* de creación de aplicaciones “Flask”. Se trata de un framework para la creación rápida de aplicaciones desarrolladas con Python. Se estuvo planteando usar el framework “Django” pero finalmente se optó por Flask porque Django es para aplicaciones mucho más grandes y complejas de lo que se proponía crear para este trabajo. Flask es mucho más intuitivo y fácil de usar, además, con solo un par de líneas se puede levantar una aplicación web sin necesidad de librerías ni herramientas accesorias.

Para empezar con Flask, se utilizó la guía ¹ proporcionada por la página oficial de este framework y un tutorial ² específico para la creación de aplicaciones como esta. Para empezar con Flask lo principal es tener instalado Python e instalar las dependencias necesarias del framework. Esto se puede hacer desde el propio entorno de desarrollo elegido, en mi caso “Visual Studio Code”, usando el instalador de paquetes “pip” de Python. En resumen, desde

¹<https://flask.palletsprojects.com/en/2.2.x/quickstart/>

²<https://j2logo.com/leccion-1-la-primer-a-aplicacion-flask/>

el terminal se instalaron los siguientes paquetes, necesarios para el correcto funcionamiento de la aplicación:

```
> pip install Flask-WTF  
> pip install email-validator  
> pip install flask-login  
> pip install flask-sqlalchemy  
> pip install cryptography
```

El paquete principal para que la aplicación funcione es el de Flask, en este caso Flask-WTF para que también incluya la validación de formularios, como sería el de inicio de sesión en el que se profundizará más adelante. El resto son paquetes que se han ido necesitando a lo largo del desarrollo de la aplicación para incluir ciertas funcionalidades.

Una vez instalados los paquetes, el despliegue de la aplicación es muy sencillo y, tan solo se necesitan unas pocas líneas de código para que funcione. Lo primero sería crear un fichero python, en nuestro caso llamado “run.py”, en el que irán los métodos asociados a las URLs que formarán la aplicación, el primer método en nuestro caso, sería el que nos lleva a la página principal, el “index”, que es básico para que funcione la aplicación. El fichero run.py más básico para que se despliegue la aplicación quedaría de esta forma:

```
from flask import Flask  
app = Flask(__name__)  
  
@app.route('/')  
def index():  
    return render_template("index.html")
```

Para que tenga sentido este fichero sería necesario crear aparte un archivo HTML llamado “index.html” con la lógica de la página principal, que puede ser lo más sencilla que se quiera. Este código funcionará de tal forma que cuando se acceda a la url de la aplicación (identificada también por terminar con este símbolo “/”) se va a cargar la página del index.html. Pero, antes, para arrancar la aplicación se necesita ejecutar varios comandos en el terminal:

```
> $env:FLASK_APP = "run"  
> python -m flask run
```

Flask incluye un servidor interno propio al que se podrá acceder desde *localhost*. Para que este servidor sepa qué aplicación debe lanzar, se usa el primer comando que apunta al fichero run del directorio en el que se encuentra la aplicación. El segundo comando es el que definitivamente lanza la aplicación. Podemos comprobar que la aplicación funciona entrando a un

navegador y accediendo a la URL “`http://127.0.0.1:5000/`” que cargará el fichero principal `index.html`.

Una vez tenemos funcionando la aplicación ya solo queda ampliar sus funcionalidades que se explicarán a continuación.

4.3.1. Funcionalidad inicio de sesión y registro

Se añade esta nueva funcionalidad para que cada recuerdo se puedan asociar a un usuario y así dar la posibilidad de guardar incontables historias de vida. De la mano del inicio de sesión, se encuentra la funcionalidad del registro de usuarios que se contará también en esta sección de la memoria. De nuevo, para ambas funcionalidades, me he guiado por el tutorial de Flask³ de la página “J2logo” y hay algunas líneas de código que he cogido directamente. En este caso, esta página web nos guía en el uso de formularios en Flask, que se requieren tanto para el inicio de sesión como para el registro de usuarios.

Los formularios que se necesiten para una aplicación de Flask es conveniente tenerlos separados en un archivo Python aparte, en este caso en “`forms.py`”. En este archivo se definirán las clases `SignUpForm` y `LoginForm` que heredan de la clase `FlaskForm`. Dentro de estas clases se definirán los atributos, variables que se corresponden con los campos a validar del formulario. La validación de los campos se hará automáticamente teniendo en cuenta el tipo de dato que se defina para cada atributo y también los parámetros opcionales que se definan para ese atributo. Por ejemplo, para el campo del email que se usa en ambos formularios se define el siguiente atributo:

```
email = StringField('Email', validators=[DataRequired(), Email()])
```

Este atributo `email` se ha definido como una cadena de caracteres y como parámetrosopcionales tiene “`DataRequired()`, `Email()`”. Es decir, cuando se haga la validación del formulario, la aplicación comprobará que el campo no está vacío y que en efecto, se trata de un correo electrónico basándose en su formato.

A parte de haber definido el formulario, se necesita una url asociada a cada página, tanto la de inicio de sesión como la de registro. Como había comentado antes, las URLs que formen parte de la aplicación deben estar definidas en el fichero “`run.py`”. Serán respectivamente las URLs “`/login`” y “`/signup`”. Hay dos funciones, una para gestionar el inicio de sesión y otra para el registro. Ambas funciones usan los métodos `GET` y `POST` porque necesitan enviar datos a la página web y recibirlos también. En estas funciones, además, será donde se invoque a las clases que se crearon anteriormente de los formularios, creando una instancia de las mismas.

— Aquí se ha dejado de contar con tanto detalle puesto que no se si se debería hablar tan en profundidad, incluso sobre el código

³<https://j2logo.com/tutorial-flask-leccion-3-formularios-wtforms/>

El inicio de sesión de usuario consiste en introducir los campos email y contraseña que se hayan usado en el registro. Si coinciden y la contraseña es correcta se dará acceso al resto de funcionalidades y se redirigirá al usuario de nuevo a la página principal pero ya identificado. En el caso de que el email no exista o la contraseña sea incorrecta se mostrará el siguiente error: “Error al iniciar sesión”, y se pedirán los datos de nuevo.

La página de registro es parecida a la de inicio de sesión pero añadiendo un campo más, el nombre, que se utilizará para que la aplicación se dirija al usuario. El email y la contraseña se guardarán en la base de datos de MySQL y se utilizarán para comprobar si el inicio de sesión es correcto. La contraseña se guarda encriptada mediante un hash gracias a la librería “werkzeug.security” de Python, que proporciona funciones para generar el hash y para comprobar si una cadena de caracteres dada coincide con la contraseña encriptada.

4.3.2. Funcionalidad chatbot

El chatbot es la funcionalidad principal de la aplicación y es en lo que a nivel de backend se ha invertido más tiempo. Sin la parte visual del frontend, el chatbot también funciona desde la línea de comandos de Python. Sin embargo, se ha decidido hacer la aplicación web para facilitar la interacción con el usuario. Esta funcionalidad consiste en un chat convencional en el que el usuario interactúa con la máquina que hay detrás. No obstante, esta funcionalidad se distingue de otras inteligencias artificiales de tipo chatbot porque es el bot quién dirige la conversación y el que pregunta o habla con el usuario y no al revés.

Lista: - Otras tecnologías para bases de datos, para el código - Fotos del funcionamiento de la aplicación - Meter funcionalidad de sacar la información recabada tanto para que la vea el usuario como para que la vea el terapeuta + apartado para el terapeuta y que pueda añadir preguntas específicas para cada usuario. - Contar por qué se ha usado mongo y hacer diagramas de base de datos - Flujo de la aplicación (de que pasa en cada caso)

4.3.3. Funcionalidad usuario terapeuta vs usuario paciente

Dentro de la aplicación existen dos roles para los usuarios, el terapeuta podrá acceder al listado de sus pacientes y a los recuerdos ya recogidos por el chatbot. Por otro lado, el paciente podrá acceder al chatbot con el que interactuaría para sacar el mayor número de recuerdos y también consultar toda la información guardada sobre él y sus respuestas.



Figura 4.10: Funcionalidad consultar pacientes registrados



Figura 4.11: Funcionalidad consultar pacientes registrados

- 4.3.4. Funcionalidad consulta de pacientes registrados por parte del terapeuta
- 4.3.5. Funcionalidad crear nuevo paciente
- 4.3.6. Funcionalidad historia de vida de un paciente

NOMBRE	EMAIL
Carmen Lopez	carmen.lopez@gmail.com
Ines Castaños	inescastanos_91@gmail.com
Pepe Gonzalez	pepe@gmail.com

Figura 4.12: Funcionalidad consultar pacientes registrados

USUARIO	CÓDIGO
Mario	577

Figura 4.13: Funcionalidad crear nuevo paciente

Ricorda

LUCIA | PACIENTES | CONVERSAR | LOGOUT

INFANCIA

- ¿Cuál ha sido el momento más feliz de tu vida?
Cuando nació mi hermano pequeño, fue un momento inolvidable

ADOLESCENCIA

- ¿En qué fecha naciste?
1999

JUVENTUD

- ¿Cuántos años tienes?
23
- ¿Cuál es tu sexo?
hombre
- ¿Recuerdas un momento en que te sentiste orgulloso?
Cuando me saqué el examen C2 de inglés

ETAPA ADULTA

Figura 4.14: Funcionalidad crear nuevo paciente

Capítulo 5

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Mejorar el análisis de sentimiento, mejorar la clasificación en etapas de vida que funciona fatal y añadirle más inteligencia al chatbot con mayor precisión y fluidez de conversación entre preguntas y respuestas. Que no solo responda con una pregunta sino que reaccione a los mensajes del paciente añadiendo algún módulo de mensajería. Mejorar la aplicación web para que el terapeuta tenga más funcionalidades como la de poder añadir las temáticas que quiera tratar con el paciente o preguntas que uiera proponer para el chatbot.

Chapter 5

Conclusions and Future Work

Conclusions and future lines of work.

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.*

Miguel de Cervantes Saavedra

Extracción de información personal a partir de redes sociales para la creación de un libro de vida. Versión electrónica, Disponible en <https://eprints.ucm.es/id/eprint/68328/>.

Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning. Versión electrónica, Disponible en <https://eprints.ucm.es/id/eprint/66857/>.

Generación de resúmenes de video-enbibtex.exe redes neuronales. Versión electrónica, Disponible en <https://eprints.ucm.es/id/eprint/68333/>.

Herramienta de ayuda guiada para la reminiscencia. Versión electrónica, Disponible en <https://eprints.ucm.es/id/eprint/68332/>.

Sistema de asistencia para cuidados de enfermos del Alzheimer. Versión electrónica, Disponible en <https://eprints.ucm.es/id/eprint/67069/>.