
Implementación de soluciones en la nube para el análisis de datos públicos a través de modelos de inteligencia artificial

Implementation of cloud solutions for
public data analysis through artificial
intelligence models



Trabajo de Fin de Master
Curso 2024–2025

Autor
Cristian Molina Muñoz

Director
Jose Luis Vazquez-Poletti
Rubén Fuentes-Fernández

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Implementación de soluciones en la nube para el análisis de datos públicos a través de modelos de inteligencia artificial

Implementation of cloud solutions
for public data analysis through
artificial intelligence models

Trabajo de Fin de máster en Ingeniería Informática

Autor
Cristian Molina Muñoz

Director
Jose Luis Vazquez-Poletti
Rubén Fuentes-Fernández

Convocatoria: *Septiembre 2025*
Calificación:

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

28 de junio de 2025

Autorización de difusión

los abajo firmantes, matriculados en el Master en Ingeniería en Informática de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores el presente Trabajo Fin de Master: “Implementación de soluciones en la nube para el análisis de datos públicos a través de modelos de inteligencia artificial”, realizado durante el curso académico 2024-2025 bajo la dirección de Jose Luis Vazquez-Poletti y Rubén Fuentes-Fernández, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Cristian Molina Muñoz

28 de junio de 2025

Dedicatoria

A mis padres, por hacer posible todo esto. Por su esfuerzo

Agradecimientos

Muchas gracias a todos los profesores y compañeros que nos han acompañado en este viaje y de los que tanto hemos aprendido

Resumen

[TODO 250 palabras sobre datos, cloud y IA] [Se redacta en pasado y no debe incluir abreviaturas, referencias a figuras o tablas ni citas bibliográficas. Tampoco se debe incluir información que no aparezca en el proyecto.]

Los ficheros de GitHub se encuentran en el siguiente repositorio:

<https://github.com/crismo04/TFM-cloud-soliutions-to-public-data/>

Palabras clave

Tratamiento de datos, Cloud, Big data, inteligencia Artificial, [TODO mas sobre clouds, se mencionan en orden alfabético]

Abstract

[TODO three paragraphs on data, cloud and AI].

The GitHub files can be found in the following repository:

<https://github.com/crismo04/TFM-cloud-solutions-to-public-data/>

Keywords

Data processing, Cloud, Big data, Artificial intelligence, TODO something else about clouds.

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Plan de trabajo	1
1. Introduction	3
1.1. Motivation	3
1.2. Work plan	3
2. Estado de la Cuestión	5
2.1. Datos	5
2.2. Nubes	5
2.3. Inteligencia Artificial	5
2.4. Trabajos anteriores	6
2.4.1. Trabajos dirigidos a	6
3. Materiales y métodos	7
3.1. Investigación	7
3.1.1. Fuentes de datos	7
3.1.2. Datos de Facebook	9
3.1.3. Datos de Google	11
3.2. Análisis de requisitos	15
3.2.1. Lenguajes	15
3.2.2. Herramientas	19
3.2.3. Tecnologías	23
4. Implementación de la aplicación	31
4.1. Descarga de datos de Facebook	34
4.1.1. Identificación de usuario	34
4.1.2. Descarga de datos	36
4.1.3. Preparación de carpetas	36

4.2.	Creación de las tablas, parseo de JSON e inserción en las tablas	37
4.2.1.	Creación de tablas SQL	37
4.2.2.	Parseo de JSON e inserción en las tablas	40
4.3.	Tratamiento de los datos	43
4.3.1.	Tratamiento	43
4.3.2.	Envío de datos a la interfaz	45
4.4.	Interfaz de usuario versión 1.	47
4.5.	Descarga de datos de Google	49
4.5.1.	Tutorial de descarga y preparación	50
4.5.2.	Tratamiento y revalorización de los datos	51
4.6.	Interfaz de usuario versión 2	53
5.	Conclusiones y Trabajo Futuro	57
5.1.	Trabajo futuro	58
5.	Conclusions and Future Work	61
5.1.	Future work	62
6.	Contribución individual	65
6.1.	Cristian Molina Muñoz	66
6.1.1.	Análisis e investigación	66
6.1.2.	Descarga de datos	66
6.1.3.	Creación de las tablas, parseo de JSON e inserción	67
6.1.4.	Tratamiento de los datos	67
6.1.5.	Interfaz de usuario	67
6.1.6.	Memoria	68
6.2.	Pablo Aguilera Heredero	69
6.2.1.	Análisis e investigación	69
6.2.2.	Descarga de datos	69
6.2.3.	Creación de las tablas, parseo de JSON e inserción	69
6.2.4.	Tratamiento de los datos	70
6.2.5.	Interfaz de usuario	70
6.2.6.	Memoria	71
A.	Tablas de SQL	73
B.	Manual de usuario y casos de uso	79
B.1.	Guía para la instalación	79
B.2.	Manual de usuario	81
B.2.1.	Bienvenida y login	81
B.2.2.	Descarga de Datos	83
B.2.3.	Creación de las tablas	86
B.2.4.	Interfaz gráfica para visualizar datos	91
Bibliografía		99

Índice de figuras

4.1. Diseño de las fases de la app	32
4.2. Diagrama de componentes del back-end	32
4.3. Diagrama de clases del back-end de la aplicación	33
4.4. Diagrama de actividad para la descarga de datos	35
4.5. Diagrama del parseo de datos	38
4.6. Diagrama de la Base de Datos	39
4.7. Diagrama del tratamiento de datos	44
4.8. Tabla de la antigua interfaz	49
4.9. Diagrama con los componentes del front	53
4.10. Interfaz de usuario, apartado de Página principal	54
4.11. Interfaz de usuario, apartado de fotos	55
B.1. Inicio de sesión	82
B.2. Login para descarga	83
B.3. Login Facebook con el correo	84
B.4. página de descarga de datos	85
B.5. Tutorial de Google	86
B.6. Tutorial de Google	87
B.7. Primero desmarcamos todo	88
B.8. Después marcamos los datos correctos	89
B.9. Por ultimo nos descargamos los datos	90
B.10. Interfaz de usuario, apartado de la Página principal	91
B.11. Interfaz de usuario, apartado de amigos	92
B.12. Interfaz de usuario, apartado de lugares, mapa y tabla	92
B.13. Interfaz de usuario, apartado de lugares, personas y fotos	93
B.14. Interfaz de usuario, apartado de fechas en forma de tabla	94
B.15. Interfaz de usuario, apartado de fechas en forma de timeline	95
B.16. Interfaz de usuario, apartado de fotos	96
B.17. Interfaz de usuario, apartado de fotos etiquetadas con amigos	96
B.18. Interfaz de usuario, apartado de páginas favoritas	97

B.19.Interfaz de usuario, apartado de logros académicos y laborales	97
B.20.Interfaz de usuario, apartado de información personal	98

Introducción

“We can only see a short distance ahead, but we can see plenty there that needs to be done.”
— Alan Turing

1.1. Motivación

Empezaremos por el principio, definiendo que son los tres principales elementos de este proyecto [TODO]

El alcance de este proyecto es, por un lado [TODO]

1.2. Plan de trabajo

Una vez definido el alcance, me gustaría destacar las cinco fases en las que se ha dividido el proyecto, que se han ido iterando para la creación de varios prototipos funcionales:

1. **Fase de investigación:** Búsqueda de información a cerca de diferentes fuentes públicas de datos, tecnologías en la nube y modelos o herramientas de IA que nos ayuden a tratar, filtrar y entender todos los datos públicos recopilados.
2. **Fase de análisis de requisitos:** [TODO]
3. **Fase de implementación:** [TODO]
4. **Fase de pruebas:** [TODO]

5. **Memoria:** Elaboración de este documento, plasmando las fases anteriores en texto y especificando el desarrollo del proyecto y los resultados del mismo.

Chapter 1

Introduction

“We can only see a short distance ahead, but we can see plenty there that needs to be done.”
— Alan Turing

1.1. Motivation

We will start at the beginning by defining what the three main elements of this project are, [TODO]

The scope of this project is, on the one hand [TODO]

1.2. Work plan

Having defined the scope, I would like to highlight the five phases in which the project has been partitioned, which have been iterated for the creation of several functional prototypes:

1. **Research phase.** Search for information about different public data sources, cloud technologies and AI models or tools that help us to treat, filter and understand all the public data collected.
2. **Requirements analysis phase:** [TODO].
3. **Implementation phase:** [TODO]
4. **Testing phase:** [TODO]

5. **Memory phase:** [TODO] The elaboration of this document, translating the previous phases into text and specifying the development of the project and its results.

Capítulo 2

Estado de la Cuestión

En este apartado expondremos el estado actual de los puntos principales de nuestro proyecto según las investigaciones realizadas, así como los trabajos o artículos relacionados con los temas a tratar. Estos son, entre otros, trabajos relacionados con los principales proveedores Cloud y su comparación, trabajos que traten con grandes volúmenes de datos públicos, o trabajos que utilicen diferentes IAs para el tratamiento de datos y la obtención de conclusiones a partir de estos

2.1. Datos

[TODO]

2.2. Nubes

[TODO]

2.3. Inteligencia Artificial

[TODO]

2.4. Trabajos anteriores

[TODO]

2.4.1. Trabajos dirigidos a ...

[TODO]

Materiales y métodos

En este capítulo vamos a describir el proceso previo a la implementación de la aplicación, es decir, el estudio de las distintas tecnologías, lenguajes de programación y herramientas que nos permitan llevarla a cabo, así como las que hemos descartado. Todo esto estará centrado en las tecnologías útiles para crear un libro de vida, del cual ya hemos hablado en el capítulo anterior.

3.1. Investigación

Aquí vamos a desarrollar todas las opciones que hemos barajado en cuanto a qué redes sociales usar y por qué, así como qué datos de los que hemos extraído son útiles y nos van a servir para la creación del libro de vida. Como ya hemos mencionado en el estado de la cuestión, el libro de vida tiene numerosos beneficios, tanto para la persona protagonista, como para todo su círculo de familiares y cuidadores, por este motivo, es importante recoger una cantidad de datos útiles suficiente, que puedan aportar valor a la realización del mismo y al tratamiento del paciente. Estos datos son los que vamos a detallar en los siguientes apartados.

3.1.1. Fuentes de datos

En la actualidad la red está plagada de numerosas webs y aplicaciones que permiten a las personas conectar con sus amigos o seres queridos, así como facilitarles la vida con numerosas tareas diarias, como desplazarse por una ciudad sin perderse, por ejemplo. En este apartado hemos tenido que filtrar las redes más útiles para el propósito de nuestro proyecto, teniendo en cuenta las características de nuestro público objetivo. Las listaremos a continuación:

- **Twitter:** Es un servicio de comunicación bidireccional con el que puedes compartir información de diverso tipo de una forma rápida, sencilla y gratuita. Principalmente es un servicio de microblogueo. Hemos decidido descartar esta red social porque actualmente la gente la utiliza para estar a la última de todas las noticias de actualidad en vez de compartir su vida, además tiene fama de ser una comunidad bastante negativa y consideramos que el libro de vida tiene que tocar los aspectos más alegres de la vida de las personas.
- **Tiktok:** Es una aplicación que permite grabar, editar y compartir videos cortos (de 15 a 60 segundos) en bucle y con posibilidad de añadir fondos musicales, efectos de sonido, filtros o efectos visuales. Aunque esta red social está actualmente muy de moda la hemos descartado en el proyecto porque la volatilidad y la importancia de sus vídeos en la vida de las personas es más reducida que en el resto de redes.
- **Instagram:** Es una red social que permite a sus usuarios subir imágenes y vídeos con múltiples efectos fotográficos como filtros, marcos, etc. Hemos decidido no utilizarla porque al estar muy ligada a Facebook, podemos abarcar el contenido de esta red desde la otra. Además, esta aplicación ya está en cierta medida estructurada como un libro de vida, por lo que no nos parecía que tuviera tantas cosas que aportar.
- **Facebook:** Es una red social creada para poder mantener en contacto a personas, y que éstos puedan compartir información, noticias y contenidos audiovisuales con sus propios amigos y familiares. Es uno de los canales digitales más conocidos por todos los usuarios que navegan hoy en día por internet, así como uno de los pioneros en este ámbito. Nos hemos decantado por incluir los datos de esta red porque prácticamente todo el mundo tiene aquí a sus familiares y amigos, y es la más completa para generar un libro de vida. Otro de los motivos por los que hemos elegido Facebook es que las pérdidas de memoria empiezan a aparecer después de los cuarenta años y esta es la red social más utilizada por gente de esta edad (Family, 2018). Además, es la red social en la que más podíamos aportar, ya que debido a la dificultad para extraer datos, hace que no existan muchos trabajos parecidos al nuestro.
- **Goolge:** Es una de las mayores empresas tecnológicas dedicadas a la recogida y tratamiento de datos, por lo que, a pesar de no ser una red social (ya que Google+ dejó de funcionar en 2019), decidimos estudiarla y ver si guardaba algún dato interesante para la creación de un libro de vida. Al tener tantas aplicaciones, Google guarda una inmensa cantidad de datos sin que el usuario se dé siquiera cuenta, algunos de los cuales hemos visto que tienen utilidad para nuestro proyecto, como por ejemplo un apartado en el que se almacenan las reseñas de los lugares

que el usuario ha visitado, o los enlaces y páginas que el usuario ha guardado en favoritos. Como la mayoría de personas usan Google para su día a día, es muy posible que todos los pacientes tengan al menos uno de estos datos, por lo que vimos de una gran utilidad añadir alguno de estos a nuestro proyecto.

Después de analizar todas estas redes sociales con sus pros y sus contras mencionados (y algunas menos importantes que no nos ha parecido útil detallar, como Reddit o cuentas en dispositivos personales), nos decantamos por utilizar Facebook y Google, en resumen, por ser dos de las mayores empresas en el área de recopilación de datos y por tener una base de usuarios que encajan con nuestro público objetivo. Estas dos Empresas guardan una cantidad masiva de datos para cada usuario que se registra en ellas, y de todos estos datos solo algunos nos van a ser útiles para nuestro objetivo, pues hay algunos demasiado privados, como cuentas e inicios de sesión, otros que no tienen ningún interés o no aportan datos e incluso datos que se repiten en varios apartados. Por esta razón, vamos a listar a continuación los datos que estas empresas permiten descargar (a fecha de la creación de este proyecto), así como un análisis de los mismos y si tienen utilidad para nosotros.

3.1.2. Datos de Facebook

Estos son los apartados que existen en Facebook a fecha de la publicación de esta memoria, es posible que en el futuro estos datos cambien, ya que durante el desarrollo, han cambiado varias veces. A continuación vamos a definir cuáles nos son útiles para la generación del libro de vida (en negrita) y cuáles no hemos considerado oportuno tratar.

- **Publicaciones:** Esto es lo que definimos como los posts que es el núcleo de nuestra aplicación pues es donde el usuario tiene el grueso de sus datos, ya que cualquier tipo de interacción en la red social es guardada como un post dentro del perfil de usuario. Muchas de las siguientes carpetas nombradas en este apartado son redundantes respecto a esta, por eso la hemos elegido como la principal.
- **Fotos:** Están dentro de post, ya que cuando publicas una foto se crea un post automáticamente, pero hay algunos datos que se guardan en esta carpeta que nos han parecido útiles y que post no almacena, por lo que los hemos extraído de este apartado.
- **Tus lugares:** Los lugares y sitios visitados, pueden ser muy importantes para el paciente, ya que suelen referenciar a viajes y momentos únicos. Además, se pueden enlazar con qué amigos ha compartido la experiencia en ese mismo lugar, e incluso las fotos publicadas en el mismo.

- **Comentarios:** Los hemos considerado muy útiles, ya que, al ser texto, se pueden estudiar a través de análisis sentimentos y relacionar las personas mencionadas con sentimientos negativos o positivos, aportando un gran valor a la relación con la o las personas involucradas.
- **Me gusta y reacciones:** Nos han resultado útiles, ya que nos han permitido afinar la puntuación que le damos a los amigos y páginas del paciente en función del tipo de reacción. Hemos visto que eran siete tipos de reacciones y que podía ser útil asignarle una puntuación a cada una.
- **Amigos:** Claramente, nos han resultado útiles para descubrir cuáles son los principales lazos de unión para el paciente, por lo que son una parte indispensable y otro de los grandes pilares de nuestra aplicación, ya que los podemos cruzar con el resto de datos para asignarles un valor dependiendo de sus interacciones con el paciente.
- **Cuentas que sigues:** Útil para determinar los gustos del paciente, ya que hay cuentas empresariales o de grupos que aportan valor a lo que el usuario considera interesante.
- **Información de perfil:** Nos ha sido de gran utilidad para obtener la información laboral, académica y residencial del paciente, que aunque es una información básica, aporta información adicional para el tratamiento.
- **Páginas:** Son muy útiles para obtener información a cerca de los gustos del paciente, ya que contienen todas las páginas de eventos y marcas con las que el paciente ha interactuado, las cuales podemos analizar para ver sus reacciones positivas y negativas y sacar así un ranking de las más valoradas por el usuario, al igual que hacemos por ejemplo, con los comentarios de amigos.
- **Vídeos:** Los vídeos de Facebook pueden llegar a ocupar mucho espacio, y son más difíciles de tratar, por lo que este volumen e información nos ha parecido excesivo para el nivel de aportación que generan respecto a las fotos.
- **Historias:** Al ser efímeras las hemos considerado menos relevantes que las fotos, que es donde el usuario plasma los momentos más importantes.
- **Mensajes:** Aunque creíamos que tiene gran utilidad, por motivos como su gran tamaño, el cual hacía que la aplicación se demorase demasiado al tratarlos, o la privacidad hemos decidido descartarlos.
- **Grupos y eventos:** Si el usuario ha reaccionado a estos grupos, esta información ya se encuentran dentro de los post.

- Marketplace e historial de pagos: Por temas de privacidad principalmente ha sido descartado.
- Elementos guardados y colecciones: Información sobre gustos ya recogida con más precisión en páginas y likes.
- Facebook Gaming: Poco relevante para generar una historia de vida, ya que se trata de minijuegos de Facebook.
- Other Activity: Recoge los toques que has dado o recibido por lo que no nos servía mucho para el libro.
- Interacciones: Similar a likes, reacciones y comentarios. Aunque los engloba no es tan preciso como analizar los anteriores.
- Papelera: Ningún dato de interés para un libro de vida, ya que son cosas que el usuario ha descartado o que quería eliminar de su cuenta.
- Vídeos cortos: Son GIFs y vídeos poco importantes, que el usuario manda en los chats, por lo que a la hora de construir un libro de vida no aporta y decidimos descartarlos.
- Recompensas: Recompensas en juegos de Facebook sin ningún valor para el libro.
- Historial de búsqueda: Son búsquedas en Facebook, que no se asocian a personas ni páginas concretas y no aportan gran información, por lo que decidimos descartarlos.
- Ubicación: Similar a lugares, pero con menos datos sobre lo que ha realizado el usuario en estos, por lo que los descartamos en pos de los primeros.
- Seguridad e información de login: Se la pedimos al usuario para registrarse, por lo que no es necesaria, además de tener también problemas de privacidad.
- Grabaciones y transcripciones de voz: Podría ser útil, pero es difícil gestionar y tratar ese volumen de datos en formato audio, para la información que podría aportar.

3.1.3. Datos de Google

Estos son los apartados que existen en el “Takeout” de Google (la web para descargar tus datos personales que la empresa almacena sobre ti) a fecha de la publicación de esta memoria. Es posible que Google en el futuro también decida modificarlos. A continuación vamos a definir cuáles nos son útiles para

la generación del libro de vida (en negrita) y cuáles no hemos considerado oportuno tratar.

- **Calendar:** Contiene los datos de tu calendario en formato iCalendar (.ics), esto es, eventos a los que has asistido, los cumpleaños que te has guardado, recordatorios, etc. Hemos decidido coger estos datos, porque pueden marcar fechas importantes en la vida de la persona o cumpleaños de amigos cercanos entre otros datos de interés.
- **Google Fotos:** Almacena las fotos y vídeos de Google Fotos y otros servicios de Google, como Google+, Blogger y Hangouts, es decir, recopila todas las fotos que Google tiene de ti. Aunque hay que tener en cuenta que seguramente sea el archivo más pesado, sí que nos puede ser útil para la creación del libro porque en los metadatos de estas fotos nos dan mucha información de cuando y donde se ha tomado esta, así como si el usuario la ha guardado en favoritos.
- **Maps (Tus sitios):** Son los registros de tus sitios destacados y tus reseñas de sitios y establecimientos. Este archivo contiene información de los lugares que te gustan o en los que has escrito una reseña. Están de una manera mucho más comprimida y fácil de analizar que en otros apartados con lugares, como por ejemplo “ubicación” y “maps”, además de que aquí está la valoración del sitio, por lo que hemos decidido usarlos para añadirlos a lugares.
- **Perfil:** Configuración e imágenes del perfil de Google del usuario. Estas imágenes de perfil de Google podrían resultar útiles para el libro de vida, ya que suelen ser las fotos a las que el paciente más importancia le da.
- **Save:** Guarda colecciones de varias cosas que el usuario ha decidido salvar (imágenes, sitios, páginas web, etc.) de Maps o de las búsquedas en Google. Es decir, cosas que al usuario le han parecido interesantes y que por tanto, creemos pueden tener utilidad para gustos, para imágenes o para lugares importantes en la vida del paciente.
- **Arts y Culture:** Guarda las galerías que has creado o visitado en Google Arts y Culture. Podría contener información sobre gustos, pero son datos muy específicos para un libro de vida.
- **Chrome:** Recopila los marcadores, el historial y otros ajustes de Chrome, que aunque podrían ser útiles para gustos, la delicadeza de los datos del historial y la dificultad para filtrar nos ha hecho decantarnos por no utilizarlos.
- **Contactos:** Son los contactos y las fotos de los contactos que has añadido, así como los que se han guardado durante las interacciones en productos de Google como Gmail. Lo descartamos porque no se puede

hacer coincidir con las listas de amigos de Facebook de una manera eficaz, ya que no se corresponde el nombre del contacto con la cuenta de Facebook.

- Correo: Almacena mensajes y archivos adjuntos de la cuenta de Gmail del usuario en formato MBOX y configuración de usuario de tu cuenta de Gmail en formato JSON. No los hemos recogido porque no nos ha parecido útil para un libro de vida, ya que se suele usar para temas formales, más que para comunicarte con amigos.
- Cuenta de Google: Datos sobre el registro y la actividad básica de la cuenta, nada que no hayamos recogido ya en otros apartados.
- Drive: Guarda los archivos de tu propiedad que tienes almacenados en Mi unidad y en Ordenadores. Hay de todo, es difícil filtrar y la mayoría van a ser trabajos o documentos, ya que las fotos se almacenan en “Fotos”, por lo que lo hemos descartado.
- Fit: Son los datos de actividad de Google Fit, lo cual es sensible por ser temas de salud y poco útil, solo serviría para saber la actividad física del paciente, lo cual no es relevante para un libro de vida.
- Google My Business: Todos los datos relacionados con la empresa del usuario, si es que tiene. Es poco útil para recuerdos que añadir a un libro de vida.
- Google Play Libros: Los títulos y los autores de los libros que hayas comprado en Google Play Libros o subido a esta plataforma, así como las notas y marcadores. En este, y el resto de apartados de Google Play, hemos descartado los datos porque aunque son cosas que pueden dar información sobre gustos, no son datos demasiado valiosos para un libro de vida, ya que estos elementos, si son muy queridos, se suelen adquirir en formato físico.
- Google Play Música: Una lista con las canciones, las playlists, las emisoras de radio, las subidas y las compras de la colección de Google Play Música, así como el historial de búsqueda y de reproducciones.
- Google Play Películas: Las preferencias, servicios, lista y valoraciones de Google Play Películas.
- Google Play Store: Datos sobre las descargas de aplicaciones y juegos, las puntuaciones y los pedidos
- Google Shopping: Recopila el historial de pedidos, datos de programas de fidelización, direcciones y reseñas de Google Shopping. Al ser información privada de pedidos y compras, no nos hemos querido meter a analizarlo.

- Hangouts: Almacena el historial de conversaciones y archivos adjuntos de Hangouts. Como la utilización de esta aplicación es mínima, no los hemos considerado.
- Historial de ubicaciones: Los datos del historial de ubicaciones recogidos mientras tenías el historial de ubicaciones activado. Guarda información sobre viajes y lugares que has visitado, pero es tan extenso (ya que Google recopila información de cada movimiento), que no hemos visto nada eficiente coger estos datos, porque, además tenemos los más importantes en "Tus sitios".
- Keep: Registra las notas y los archivos multimedia adjuntos almacenados en Google Keep. Aunque se pueden guardar fotos y notas personales, estas tienen, por lo general, poca relevancia en la vida del usuario.
- Maps: Guarda las preferencias y sitios personales de Maps, es decir, lugares que te gustan, pero, al igual que en la ubicación, aportan poca información para los Gigabytes de datos que suele ocupar, por lo que nos quedaremos con el apartado de "Tus sitios" únicamente
- Mi actividad: Registros de los datos de actividad, junto con archivos adjuntos a esta, los cuales no aportan a la información ya recogida.
- My Maps: Mapas, capas, funciones y elementos multimedia almacenados en My Maps sin utilidad para el proyecto.
- Publicaciones en Google: Los datos del historial de Publicaciones en Google, como las colecciones de cuentas, las publicaciones, los cameos, los datos de métricas, y todas las imágenes y los vídeos que has subido aquí. Al no ser datos personales *per se*, sino más bien aportaciones, no los hemos considerado.
- Recordatorios: Recordatorios que has creado con Google, como cumpleaños de amigos, cosa que ya recogemos en Calendar de forma más amplia, por lo que este apartado no lo hemos considerado.
- Search Contributions: Las valoraciones, las reseñas, los comentarios y otras contribuciones que has hecho en la Búsqueda de Google, que no consideramos muy importantes al tener ya los lugares guardados, que sí que pueden aportar más que las reseñas a las búsquedas.
- Servicios de Juegos de Google Play: Los datos, incluidos los logros y las puntuaciones, de los juegos de móvil a los que juegas, sin mucha información aparte de algún interés por algún juego, pero que no hemos considerado importante.
- Street View: Las imágenes y los vídeos que has subido a Google Street View, es decir, más información de mapas que no aporta a lo que ya

tenemos.

- Tareas: Los datos de las tareas pendientes y completadas del usuario, que no aporta información a los datos que ya recogemos en calendar.
- YouTube y YouTube Music: El historial de reproducciones y búsqueda, los vídeos, los comentarios y otro contenido que hayas creado en YouTube y YouTube Music. Ya que YouTube es la mayor plataforma de contenido de vídeo en el mundo y que solo podría tener información sobre los gustos del paciente, lo hemos considerado fuera del alcance de este proyecto.

3.2. Análisis de requisitos

En un proyecto como este, basado en el código que programemos, uno de los puntos más importantes es la elección de los lenguajes de programación, librerías y herramientas a utilizar. Basándonos en nuestros conocimientos adquiridos tanto en la carrera, como fuera de ella, como investigando para este proyecto y en la utilidad de las tecnologías, listaremos a continuación los lenguajes, herramientas y tecnologías tanto utilizadas como consideradas para llevar a cabo este proyecto.

3.2.1. Lenguajes

Estos son los lenguajes que hemos utilizado, por considerarlos los más oportunos, así como los que hemos analizado, pero finalmente se han descartado o se han dejado de utilizar durante el desarrollo del proyecto por diferentes inconvenientes.

3.2.1.1. Principales

PYTHON

Python es un lenguaje de programación interpretado y centrado en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Nosotros nos hemos decantado por una programación orientada a objetos combinada con imperativa, por parecernos la más organizada y funcional.

Utilizamos Python en la mayor parte de la aplicación, principalmente en el back-end y el tratamiento de datos, ya que nos permite enlazar con facilidad

las distintas tecnologías, además de ser uno de los lenguajes más útiles para el tratamiento de datos, con miles de librerías sustentadas por una gran comunidad de usuarios. Esto nos ha permitido ahorrar esfuerzo y tiempo en cosas como implementar la interfaz para obtener los datos y tratarlos. Pese a no ser un lenguaje que hemos estudiado en la carrera, las ventajas anteriormente citadas y su popularidad creciente en el mercado laboral, nos motivaron para aprenderlo y utilizarlo en este proyecto.

SQL

SQL es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Es un sistema que facilita el tratamiento de datos, así como la separación de estos datos del programa principal, permitiendo tener más modularidad. Utilizamos SQL para almacenar información en las tablas relacionales, así como para extraer esta misma información, tratarla y almacenarla ya tratada en la base de datos.

Este es uno de los apartados en los que más nos ha costado decidir, ya que había opciones como Mongo, o un dataframe en Python, como los que permite crear Pandas, que tenían también mucha potencia a la hora de tratar datos, al final nos hemos decantado por una base de datos SQL, porque una base de datos no relacional como Mongo, es más difícil de normalizar y de tratar los datos, esta no escala bien cuando tienes una gran cantidad de datos con grafos (personas, comentarios, etc.). Esto era una facilidad que nos daba SQL, ya que queríamos modularizar el proceso y tratar diferentes tablas de gran volumen para facilitar el trabajo futuro.

Por otro lado, los dataframes tenían un tiempo de respuesta más lento de lo que deseábamos para el volumen de datos que puede llegar a tener Facebook, pudiendo sobrepasar la potencia de la memoria para tratarlos, además, no nos daban tampoco la modularidad deseada, ya que se quedaba todo dentro del mismo código. Otro motivo es que la base de datos relacional nos permitía hilar los múltiples contenidos de media, tags y places que puede contener un único post dentro de Facebook, o poder combinar estos datos con los de Google, facilitándonos la extracción de los datos que pensábamos más relevantes.

React JS

React es una biblioteca JavaScript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software

libre, con más de mil desarrolladores libres contribuyendo. Hemos elegido React por la cantidad de componentes gráficos que tiene para el desarrollo Web, los cuales nos han permitido desarrollar una interfaz bonita y amigable para el terapeuta y paciente que van a darle uso.

También hemos elegido React porque nuestros tutores nos lo recomendaron y nos ha parecido más actualizado a lo que aprendimos cuando cursamos la asignatura Aplicaciones Web, en la que se creaban webs con PHP y HTML, lo cual nos parece una forma un poco obsoleta. Esto nos ha permitido adquirir nuevos conocimientos más modernos sobre el desarrollo de aplicaciones web y crear una interfaz mucho más eficazmente de lo que podríamos haber logrado con otras tecnologías, y con una apariencia más amigable. Además, React permite que las aplicaciones que implementes, se puedan visualizar también correctamente en dispositivos móviles o Tablets, cosa que nos ha parecido útil, ya que en algunos casos es posible que se utilice desde una tableta, o cualquier otro dispositivo.

Para la implementación de la interfaz hemos investigado varios componentes de código libre que pudieran ayudar a la visualización de todo lo que nuestra aplicación necesitaba, como pueden ser tablas, galerías de imágenes o líneas de tiempo (Brillout, 2019). React proporciona una forma sencilla de unir todos estos componentes que la comunidad crea, modificarlos a nuestro gusto, y gestionar nuestros datos, los cuales vienen del Backend creado en Python con la ayuda de Flask, y que aquí tratamos y adaptamos para poder utilizar en los ya citados componentes.

L^AT_EX

L^AT_EX es un sistema de composición tipográfica de alta calidad que incluye funcionalidades diseñadas para la producción de documentación técnica y científica. Es el estándar de facto para la comunicación y publicación de documentos científicos, el cual nos ha permitido desarrollar una memoria profesional y facilitar el diseño sin tener que preocuparnos por la forma cada vez que añadíamos cambios. Hemos usado L^AT_EX para desarrollar este documento en la aplicación de TeXstudio y el compilador MikteX.

3.2.1.2. Secundarios

JavaScript native

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Utilizamos JavaScript para la

creación del tutorial de descarga de datos de Facebook. También porque Selenium usa JavaScript para sus operaciones, además, es la base de React, en el cual está implementada nuestra interfaz y que ya explicamos anteriormente.

HTML

HTML (hypertext markup language) se utiliza para el desarrollo y creación de páginas web. Se compone de una serie de etiquetas que el navegador interpreta. Entre las etiquetas que se incluyen dentro del lenguaje HTML se encuentran: hipervínculos, etiquetas para imágenes, saltos de página, entre otras. Hemos utilizado HTML para la creación del grid en el tutorial. Lo intentamos utilizar para mostrar las tablas resultado en la interfaz de Tkinter, pero las librerías que combinan estas dos tecnologías están bastante limitadas, ya que no nos proporcionaban el elemento TableRow (<tr>), por lo que al final decidimos usar la propia librería de Tkinter, que tiene un elemento treeview (que al final también descartamos). Finalmente, lo usamos en la nueva interfaz web con React, ya que tiene elementos los cuales es necesario definirlos con HTML.

3.2.1.3. Descartados

Java y C++

Java / C++ son otras alternativas a Python, más dirigidas a la programación orientada a objetos, pero pese a conocerlas más por haberlas estudiado en la carrera y ser lenguajes robustos, no nos daban tantas herramientas como las que ofrece Python para el tratamiento de datos.

Mongo, GraphQL y otros lenguajes de BBDD

Una de las decisiones más difíciles ha sido elegir entre una base de datos relacional, una base de datos no relacional (como Mongo DB o GraphQL) o usar simplemente un dataframe como pandas.

Una base de datos de grafos (Como Mongo o GraphQL) es una base de datos que utiliza estructuras de grafos para consultas semánticas con nodos y propiedades para representar y almacenar datos. Basándose en aristas y relaciones. El grafo relaciona los elementos de datos del almacén con una colección de nodos y aristas, y las aristas representan las relaciones entre los nodos. Las relaciones permiten vincular directamente los datos del almacén y, en muchos casos, recuperarlos con una sola operación. Mientras que otras

bases de datos calculan las relaciones en el momento de la consulta mediante costosas operaciones JOIN, una base de datos gráfica almacena las conexiones junto con los datos del modelo. Estas bases de datos destacan en la gestión de datos altamente conectados y consultas complejas. Por lo que, aunque podrían ser útiles para datos como los de Facebook, la mala escalabilidad nos ha parecido un inconveniente suficientemente grande para no usarlas.

El DataFrame de Pandas por su parte, es una estructura de datos tabular bidimensional, potencialmente heterogénea con ejes. Un DataFrame es una estructura de datos bidimensional, es decir, los datos se alinean de forma tabular en filas y columnas. Consta de tres componentes principales, los datos, las filas y las columnas. Esta tecnología tiene prometedoras funciones para tratar datos en Python, pero nos parecía que los inconvenientes, como su dificultad para tratar grandes volúmenes de datos, tenían más peso en la elección.

En resumen, por el volumen de datos y la modularidad, acabamos descartando estas tecnologías y escogiendo SQL.

Php

PHP es un lenguaje de programación adaptado al desarrollo web, que puede servir para la creación de webs o aplicaciones web dinámicas o para conectar la parte del servidor y la interfaz de usuario. Hemos descartado este lenguaje por parecernos obsoleto, prefiriendo crear el front-end de nuestra aplicación en React, y porque la parte del back-end de nuestra aplicación estaba escrita en Python, el cual cuenta con la librería Flask para la creación del mismo y la conexión con la interfaz, la cual nos parece una forma más elegante de implementarlo.

3.2.2. Herramientas

A continuación listamos las herramientas que hemos creído más adecuadas para el proyecto y también las que hemos estudiado pero sin llegar a utilizar.

3.2.2.1. Principales

XAMPP

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. Utilizamos XAMPP para conectar la base de datos con nuestra aplicación y visualizar los datos

de las tablas en su página de administración (phpMyAdmin), lo cual nos ha parecido una gran manera de visualizar estos datos y probar código SQL aislado para luego añadirlo a nuestro programa.

Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y MacOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Utilizamos Visual Studio Code como entorno de desarrollo software, ya que comparándolo con otras alternativas que nos brindan en la carrera, lo creemos bastante más útil, sobre todo para la programación en Python o React.

Lo que nos ha hecho decantarnos por él por encima del resto, es la gran comunidad que tiene detrás, la cual cuenta con un gran número de tutoriales y extensiones que nos facilitan mucho la programación y la integración con otras aplicaciones como Github. También destacar su intérprete, para probar pequeños fragmentos de código, lo cual nos ha ahorrado tiempo en depuración de errores.

Github

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.

Utilizamos GitHub como sistema de control de versiones y repositorio de código por su tremenda utilidad para comunicarnos y trabajar en paralelo, lo cual ha sido una necesidad en los tiempos de pandemia en los que este proyecto se ha realizado. Esto nos ha asegurado no perder nada de progreso y llevar un control del avance del proyecto en todo momento. Además nuestros tutores nos facilitaron un repositorio privado en el que nos asegurábamos la seguridad del código, por lo que su uso era casi una obligación frente a otras alternativas.

Discord y aplicaciones de trabajo remoto

Discord es un servicio de mensajería instantánea freeware de chat de voz VoIP, video y chat por texto. Funciona a través de servidores y está separado en canales de texto o de voz. Es una de las apps más populares y con mayores utilidades.

Con la situación extraordinaria en la que nos ha tocado realizar este proyecto, la utilización de estas aplicaciones de trabajo remoto se nos ha hecho indispensable. Hemos utilizado Discord como aplicación principal para la comunicación entre los integrantes del proyecto, pudiendo ver con esta los avances que íbamos realizando en las diferentes partes del proyecto a tiempo real y brindarnos ayuda entre nosotros, la cual ha sido esencial en la realización de todo el mismo.

TeXstudio y MiKTeX

TeXstudio es un editor de L^AT_EX de código abierto y Multiplataforma con una interfaz similar a Texmaker. TeXstudio es un IDE de L^AT_EX que proporciona un soporte moderno de escritura, como la corrección ortográfica interactiva, plegado de código y resaltado de sintaxis, por lo que lo hemos considerado ideal para la elaboración de este documento. Mientras que MiKTeX es el gestor de paquetes integrado, que instala los paquetes que hacen falta para el correcto funcionamiento de TeXstudio y para la creación de este documento.

Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Nos pareció el entorno más potente para la creación de nuestra aplicación y lo usamos en tiempo de ejecución para visualizar la interfaz y para instalarnos todos los componentes y estilos que hemos decidido utilizar en la misma.

Draw.io

Draw.io es una herramienta de diagramación gratuita con la que se puede dibujar cualquier tipo de mapas mentales, mapas conceptuales, esquemas o diferentes representaciones gráficas, como diagrama de jerarquía o conjuntos. Utilizamos esta herramienta durante la primera fase de diseño para poder compenetrarnos y tener una idea clara de la aplicación que íbamos a realizar. También la hemos usado para la creación de mockups en la parte del diseño web para tener una idea más clara de como queríamos que fuera.

PlantUML

Es un proyecto open source, a la vez que un lenguaje de programación, que permite crear diagramas de diferentes tipos en un formato UML, como pueden ser diagramas de casos de uso, de Componentes o de Actividad, etc. Estos diagramas se programan en el lenguaje PlantUML y se pueden exportar en diversos formatos. Lo hemos utilizado para la creación de los diagramas de diseño finales de la aplicación, realizados para tener una idea clara y concisa de como iba a ser esta.

3.2.2.2. Descartadas

Microsoft Visual Studio y Eclipse

Eclipse, Visual Studio Pro u otros entornos que también valoramos fueron finalmente descartados, porque, a pesar de conocerlos por diversas asignaturas, no proporcionan la comunidad de VScode y esto nos complicaba el hecho de programar y enlazar con otras aplicaciones.

Gitlab y Google Drive

Gitlab, Google Drive u otro control de versiones son (aun con la barrera de la complejidad de aprender a usar un control de versiones) la mejora en cuanto a la programación en paralelo, al ser dos integrantes en este TFG, es sustancial, por lo que, aun habiendo usado Drive para almacenar ciertos ficheros, al final utilizamos principalmente las funcionalidades que nos proporcionaba Github. Gitlab fue descartado por la posibilidad de tener un directorio privado en Github.

TexWorks y Overleaf

Algunos de los motivos por los que descartamos el editor en línea Overleaf es por la seguridad del documento, ya que con editores locales como TexStudio nos asegurábamos totalmente de ello. Otro motivo era que no podíamos permitirnos la posibilidad de que la página se cayese o cerrase el servicio y nos dejase en la estacada. Y por último que no tiene el control del proceso de compilación que tenemos usando un editor instalado. Respecto a TexWorks sabemos que tiene un gran parecido a TexStudio, pero ya que debíamos decantarnos por uno de los dos elegimos TexStudio.

Heidi DB

HeidiSQL, inicialmente conocido como MySQL-Front, es un software libre y de código abierto que permite conectarse a servidores MySQL, así como Microsoft SQL Server y PostgreSQL. Es la alternativa a usar XAMPP y Maria DB, tiene potencia para gestionar las BBDD sin necesidad de tener un gran conocimiento en SQL. Nos decidimos por XAMPP porque nos parecía que nuestro conocimiento en SQL nos iba a facilitar el uso de estas herramientas sin necesidad de la ayuda que proporciona Heidi.

3.2.3. Tecnologías

En este apartado vamos a explicar todas las tecnologías, librerías de Python y componentes de React (entre otras cosas) que hemos encontrado, así como los que finalmente nos decantamos por utilizar, explicando que son y para qué las utilizamos.

3.2.3.1. Principales

Tkinter

Tkinter es un binding de la biblioteca gráfica Tcl/Tk para el lenguaje de programación Python. Se considera un estándar para la interfaz gráfica de usuario para Python y es el que viene por defecto con la instalación para Microsoft Windows (PythonSoftwareFoundation, 2021d).

Utilizamos Tkinter por su facilidad y simpleza para la creación de la interfaz gráfica de usuario en el apartado de descarga de datos. Es una librería que recuerda bastante al lenguaje de programación CSS que ya conocemos por asignaturas como Aplicaciones Web y que nos proporciona una interfaz simple y eficaz para que le resulte más fácil de manejar a los usuarios.

Sqlite3

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C (PythonSoftwareFoundation, 2021b).

Utilizamos esta tecnología porque es una base de datos muy compacta, que nos sirve para almacenar en local el estado de la descarga de datos por parte del usuario, así como la última fecha de descarga, lo que nos sirve para comprobar si deben volver a pasar por el proceso de descarga, el cual puede

ser tedioso. Es una de las bases de datos que menos espacio ocupa y de las más rápidas, por lo que era perfecta para esta necesidad.

Selenium

Selenium es un entorno de pruebas de software para aplicaciones basadas en la web. Selenium provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas (CODERASHA, 2019).

Hemos utilizado Selenium para la automatización de la descarga de datos. Facebook rehúsa mucho en cuanto al scraping de sus datos, complicando bastante esta operación, por lo que tras muchos intentos y cambios por parte de la aplicación que dejaron nuestro código obsoleto, creímos que lo mejor era usarlo para crear un tutorial que facilitara las cosas al terapeuta. Esto nos ha facilitado el camino, pues nos hemos desentendido del manejo de datos personales, al no tener que almacenar la contraseña de los usuarios, además de hacerlo más reutilizable ante los numerosos cambios de formato de Facebook.

Dentro de Selenium hemos utilizado WebDriverWait para esperas, webdriver, que es la que gestiona la mayor parte del proceso y controla Google Chrome y expected-conditions, que junto a WebDriverWait, sirve para gestionar las condiciones de espera. (SoftwareFreedomConservancy, 2021).

Os y Sys

Os y Sys son módulos de Python que permiten realizar operaciones relacionadas del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc. (PythonSoftwareFoundation, 2021f), (PythonSoftwareFoundation, 2021h).

Estas dos librerías de Python las hemos utilizado para el manejo de ficheros y rutas dentro de la aplicación, haciendo que sea exportable a otros equipos sin necesidad de especificar rutas o configurar demasiados parámetros, entre otras cosas como el manejo de carpetas y datos dentro del sistema operativo.

GoogleTrans y Google_trans_new

estas son dos bibliotecas de Python que permite traducir el idioma de un texto a través de la tecnología de Google (PythonSoftwareFoundation, 2020a).

Hemos utilizado estas librerías para traducir el texto de los comentarios publicados por el paciente, así como las reseñas de Google u otros textos,

para después poder analizar el sentimiento de estos, pues las alternativas para analizar texto en español eran de un nivel inferior, y haciendo pruebas hemos comprobado que los resultados eran mejores con el método de traducción. Hemos utilizado las dos durante el desarrollo porque diversas actualizaciones dieron problemas y tuvimos que variar entre ellas un par de veces.

Text2emotion

Text2emotion es un paquete de Python que permite extraer las emociones de un texto (Band, 2020b).

Hemos utilizado Text2emotion para analizar los sentimientos de los comentarios publicados por el paciente, y dar así una puntuación a los textos que hemos extraído de los datos de Facebook y Google. Esta aplicación separa los textos que analiza en 5 categorías: Angry, Fear, Happy, Sad y Surprise. Hemos dado una puntuación a cada uno de estos sentimientos para completar con ello una puntuación total con la que darle una nueva capa de significado a los datos.

ZipFile

ZipFile es un módulo de Python que proporciona herramientas para crear, leer, escribir, agregar y listar un archivo ZIP.

Utilizamos ZipFile para descomprimir los datos personales del paciente, ya que al descargarlos de Facebook y Google, estos vienen en formato ZIP e incluso varias carpetas, y queríamos hacer el proceso lo más automático posible, evitando que el terapeuta tuviera que intervenir para cosas como descomprimir archivos.

Inspect

Inspect proporciona varias funciones útiles para ayudar a obtener información sobre objetos vivos como módulos, clases, métodos, funciones, tracebacks, objetos de marco y objetos de código (PythonSoftwareFoundation, 2021c).

La hemos utilizado para sacar la carpeta en la que el proyecto está alojado dentro del ordenador del usuario y que así la aplicación funcione correctamente en cualquier equipo, sin necesidad de especificar carpetas.

Time

Este módulo proporciona varias funciones relacionadas con el tiempo (Python-SoftwareFoundation, 2021i).

Lo usamos para proporcionar un ID de usuario único y dependiente de epoch y para gestionar las fechas que vienen en los datos de Facebook y Google, ya que estas tienen diversos formatos.

Shutil

Librería que ofrece varias operaciones de alto nivel en archivos y colecciones de archivos (PythonSoftwareFoundation, 2021g).

La utilizamos para la limpieza de datos en caso de que el usuario quiera volver a descargarlos, para evitar duplicados y para mover las fotos para que React pueda tener acceso a ellas.

Stat

Librería para dar permisos de escritura en diversos sistemas operativos (Python-SoftwareFoundation, 2021e).

La utilizamos para tratar la carpeta con datos y al igual que con shutil, borrar los datos que ya no sirven.

mysql.connector

Es una librería que permite que los programas de Python accedan a las bases de datos MySQL, utilizando una API que cumple con la Especificación de la API de la base de datos de Python v2 (OracleCorporation, 2021).

La utilizamos para establecer la conexión con la base de datos SQL y ejecutar consultas y operaciones.

Flask

Flask es un framework de Python para crear aplicaciones web de forma rápida y sencilla, lo cual nos ha permitido conectar nuestra aplicación web con los datos de Python en muy pocas líneas y de una manera eficiente, pudiendo manejar y pasar datos, así como tener una web de desarrollo donde hacer pruebas.

Librería JSON

Librería para trabajar con datos JSON PythonSoftwareFoundation (2021a).

La utilizamos a la hora de parsear los datos de Google y Facebook pues la gran mayoría viene en este formato. También para pasar datos al Back-end de Flask, pues esta librería maneja mucho mejor este tipo de datos. Además de por una función de decodificar y codificar los datos string a UTF-8 de manera sencilla.

Webbrowser

Proporciona una interfaz de alto nivel que permite mostrar documentos basados en la web a los usuarios (PythonSoftwareFoundation, 2020c).

Nosotros lo usamos para redirigir al usuario a la página de descarga de datos de Google y que así no tenga que buscarla en su navegador.

Re

Proporciona operaciones de coincidencia de expresiones regulares similares a las que se encuentran en Perl (PythonSoftwareFoundation, 2020b).

Hemos utilizado esta librería para limpiar de signos de puntuación, números y links los comentarios del usuario antes de puntuarlos.

String

Proporciona operaciones de cadenas de caracteres (PythonSoftwareFoundation, 2019).

La utilizamos en el archivo LogicTables.py para extraer la información útil de las publicaciones, comentarios y reacciones de Facebook entre otras cosas menores.

Jicson

Librería que permite transformar del formato ICS (ICalendar) a JSON (JS-piner, 2017).

Hemos utilizado esta librería para poder tratar el calendario de Google, ya que es más sencillo trabajar con datos JSON.

CSV

El módulo CSV implementa clases para leer y escribir datos tabulares en formato CSV (PythonSoftwareFoundation, 2018).

Utilizamos la librería CSV para leer los sitios favoritos marcados en Google por el usuario, ya que al contrario que el resto, vienen en este formato y nos parecía más fácil tratarlos en CSV que traducirlos a JSON.

React-router

Es una colección de componentes de navegación que se añaden de forma declarativa a su aplicación. Ya sea desde tener una URL que se pueda marcar como favorita o una forma de navegar entre las páginas de React en todo momento. La utilizamos para redirigir al usuario a una página nueva en el navegador.

React-router-dom

Es una librería de React que nos provee funcionalidades para crear una barra de navegación con BrowserRouter y NavLink, lo que le permite al usuario cambiar de página fácilmente y navegar entre las diferentes páginas que hemos creado.

React-photo-gallery

Es una librería de React que nos proporciona un componente para mostrar galerías de imágenes. La utilizamos para crear galerías amigables donde mostrar las fotos más destacadas del usuario así como ampliar y visualizarlas dentro de un carrusel de imágenes.

React-images

Es un componente para la creación de un carrusel altamente personalizable y compatible con dispositivos móviles para mostrar imágenes en ReactJS. Lo usamos dentro de las galerías de fotos para mostrarlas en formato carrusel cuando el usuario pincha en las mismas, y así pueda visualizarlas en un tamaño mayor.

Simple-react-google-maps

Componente de React de Google Maps que proporciona todas las características de esta aplicación, el cual hemos usado para la creación de un mapa con los lugares más importantes para el usuario. Este funciona con una ApiKey que nos hemos intentado descargar, pero al ser de pago, hemos terminado usando la versión de desarrollo, la cual nos ha permitido añadir todos los lugares en marcadores visibles en un mapa global.

React-circle-modal

Es una librería de React que nos facilita componentes para la creación de ventanas emergentes o pop-ups. La hemos utilizado con el fin de generar ventanas emergentes para lidiar con páginas en las que había que mostrar varios carruseles de fotos.

React-vertical-timeline-component

Es un componente de React que permite generar líneas de tiempo. La hemos utilizado para mostrar la información de fechas del usuario en orden cronológico.

Rsuite-table

Es una librería de React que nos proporciona un componente tabla muy editable y parametrizado. La hemos usado para crear tablas que muestren los datos ya tratados en un formato entendible para el terapeuta.

React-awesome-button

Es una librería de React que nos facilita componentes para crear un botón. La hemos utilizado para crear botones que te redirijan a otras páginas, haciendo la interfaz más accesible.

3.2.3.2. Descartadas**BeautifulSoup y Scrapy**

Ambas son alternativas a Selenium con capacidades parecidas. Beutifull soup lo descartamos porque aun siendo amigable para aprender y utilizar, las

dependencias que tiene nos parecían un punto en contra, ya que dificultan su uso. Scrapy lo descartamos por estar menos enfocado a Javascript, lo que nos resultaba menos útil. Además, buscando tutoriales para descargar datos de las redes sociales que utilizamos, la mayoría utilizaban Selenium, por lo que nos acabamos familiarizando con él.

Text Analytics y Sentiment Analysis API

Text Analytics y Sentiment Analysis API es una API de Microsoft para el análisis de sentimientos, la cual es bastante potente y dispone de muchas características, pero nos pareció demasiado pesada para nuestro proyecto, además de ser propiedad de una empresa privada, lo que nos traía problemas de licencia. En cambio Text2emotion es una librería bastante sencilla y que nos daba la funcionalidad suficiente para nuestras necesidades, por lo que nos decantamos por esta última.

Django o Back-end con SQL

Para la parte del back-end, aunque utilizamos Flask, estudiamos tecnologías como Django, un framework de Python para implementar un modelo de vista controlador creado para el desarrollo de webs complejas, o la utilización de las tablas de SQL directamente desde el Front-end, pero esta última opción dejaba todos los datos al descubierto y nos limitaba en la fase de tratamiento de datos. Finalmente, entre Django y Flask, acabamos usando Flask, por ser la más sencilla ante este tipo de aplicaciones.

Capítulo 4

Implementación de la aplicación

El objetivo que perseguimos al implementar este proyecto es conseguir desarrollar una aplicación web que tenga el formato de un libro de vida y pueda ser útil para los terapeutas. En este, el usuario podrá navegar por las distintas páginas que se corresponderán con distintos tipos de datos que el paciente tiene en sus redes sociales, como pueden ser fechas, fotos, amigos, etc. Para conseguir estos datos necesitamos implementar también una forma de extraerlos y tratarlos, por lo que en este capítulo vamos a explicar las distintas etapas que hemos seguido hasta lograrlo, teniendo en cuenta todo el diseño y fases por las que hemos pasado, así como las ampliaciones que hemos hecho en la misma. Hemos basado el diseño de la aplicación en toda la investigación vista en el capítulo anterior, intentando que sea lo más modular posible y que al usuario le resulte fácil obtener los datos y visualizarlos.

En la **figura 4.1** se muestra el flujo que diseñamos como idea inicial para la creación de la aplicación y por el que el usuario debería de pasar para la obtención y visualización de los datos. Después, en las **figura 4.2 y 4.3** se muestran los diagramas de clases y de Componentes de lo que finalmente ha sido nuestra aplicación por la parte de Python. Posteriormente, cuando hablemos de la interfaz, la acompañaremos con los suyos correspondientes.

Para llegar a nuestro objetivo de diseño, hemos dividido la implementación de la aplicación en diferentes módulos, los cuales hemos ido creando siguiendo el diseño inicial. Estos son:

- **Descarga de datos de Facebook.**
- **Creación de las tablas, parseo de JSON e inserción.**
- **Tratamiento de los datos.**
- **Interfaz de usuario versión 1.**

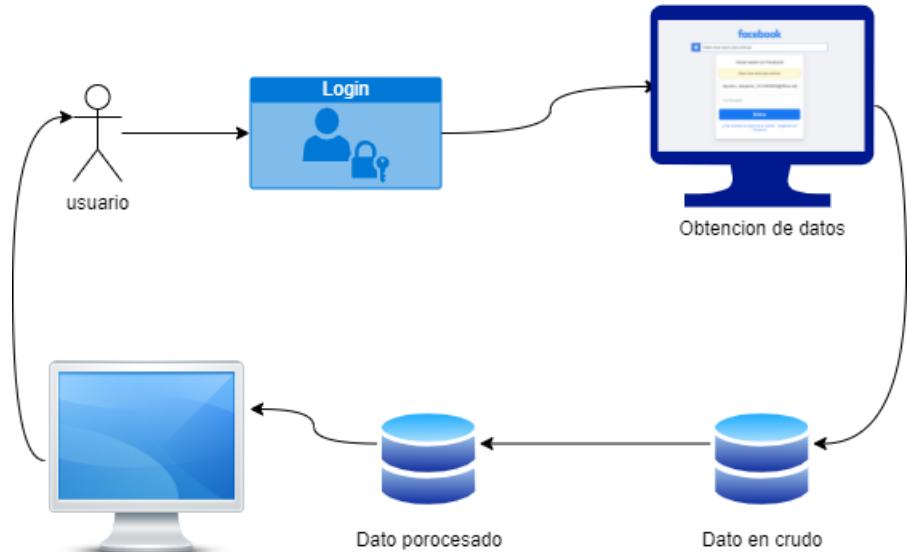


Figura 4.1: Diseño de las fases de la app

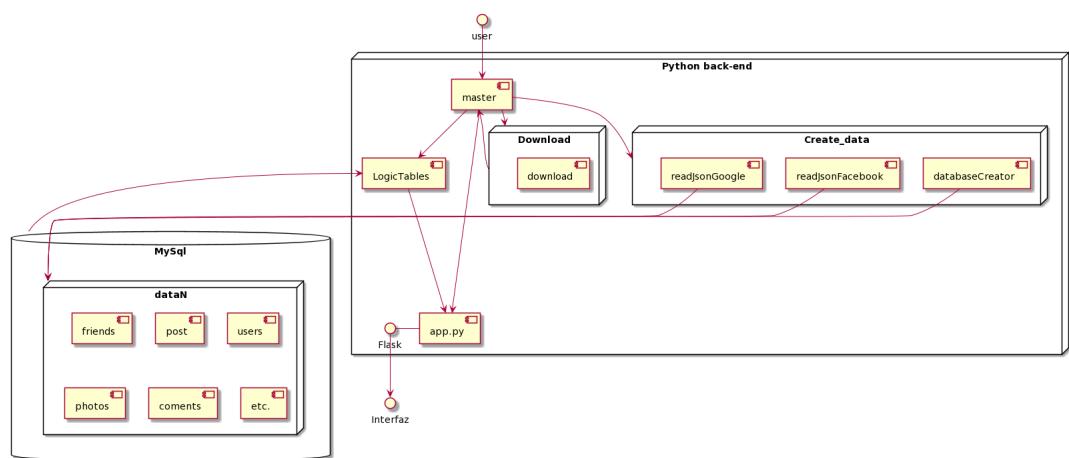


Figura 4.2: Diagrama de componentes del back-end

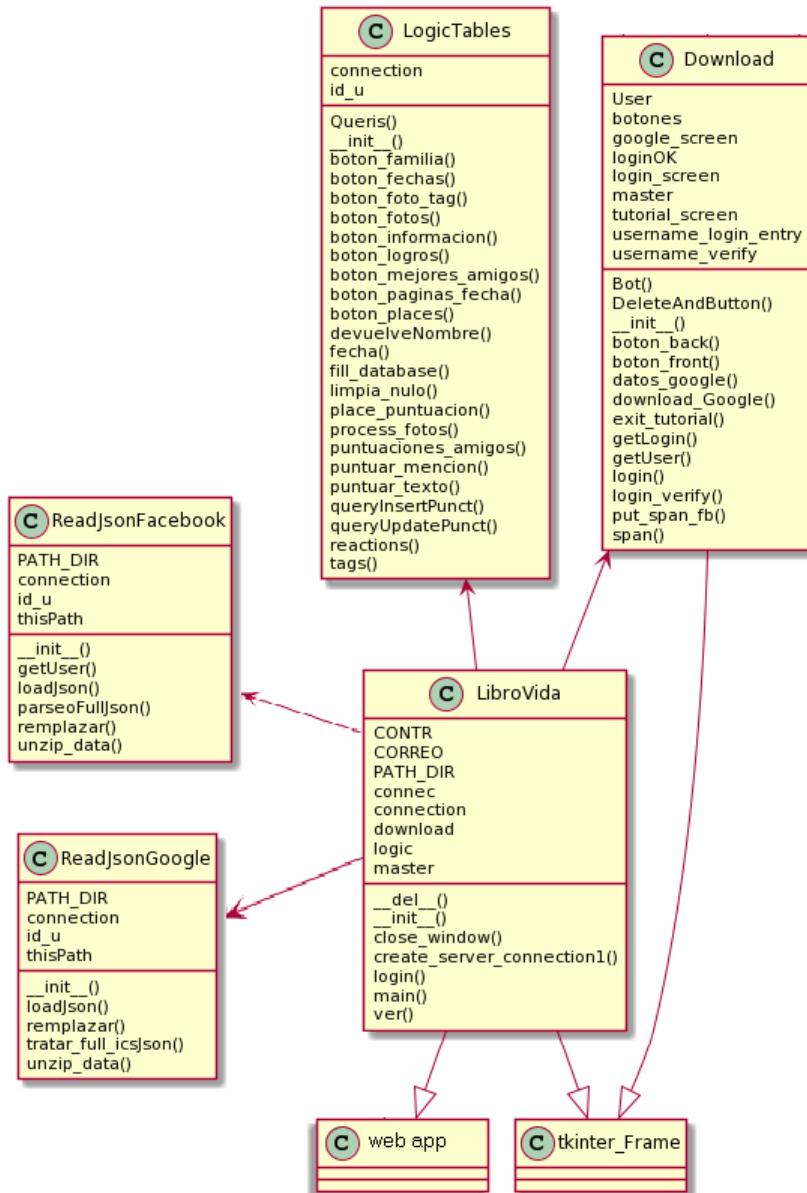


Figura 4.3: Diagrama de clases del back-end de la aplicación

- **Descarga y tratamiento de datos de Google.**
- **Interfaz de usuario versión 2.**

En las siguientes secciones vamos a explicar en más detalle en que constan estas etapas. Para ver el código en detalle, consulta el GitHub del proyecto: <https://github.com/NILGroup/TFG-2021-RedesSociales>

4.1. Descarga de datos de Facebook

Esta primera parte de la aplicación es la que más ha variado en todo el proceso, aunque empezó siendo un scraper que sacaba todos los datos automáticamente sin interferencia del usuario, las dificultades que pone Facebook a este tipo de técnicas nos obligó a optar por una opción mucho menos potente, pero igual de eficaz. Dividiéndose en las etapas de: Identificación de usuario, Tutorial de descarga y Preparación de carpetas. Todas estas etapas se ven reflejadas en la **Figura 4.4** donde se muestra el diagrama de Actividad a seguir para descargar los datos. En las siguientes subsecciones definimos también las etapas de este proceso:

4.1.1. Identificación de usuario

Los datos de Facebook en formato JSON tienen un tamaño considerable, por lo que el tiempo que tarda en proporcionártelos puede variar desde varios minutos a incluso rondar la hora si el usuario es muy activo. Esto hace necesaria una manera de que el usuario decida si volver a descargarlos o no, para esto usamos SQLite3, la base de datos ya explicada en capítulos anteriores y que nos permite almacenar una confirmación de las descargas para cada usuario que haya usado la aplicación.

La función que domina esta parte de la aplicación es **login_verify**, la cual se conecta a la base de datos y comprueba que el usuario no está ya registrado en ella, así como la última vez que ha descargado los datos, para preguntarle si considera necesario volverlos a descargar a través de un span o ventana emergente.

Los spans se gestionan con funciones auxiliares, que son la función **span**, la cual genera un pop-up con botones para mostrar información y confirmar pasos. Esta recibe como argumento una lista de los botones necesarios, el título del span, el texto a mostrar y el tamaño de la ventana, devolviendo el botón que se ha pulsado en el span para actuar en consecuencia. Esto lo hace con ayuda de la función **DeleteAndButton**, que borra el span y avisa del botón pulsado.

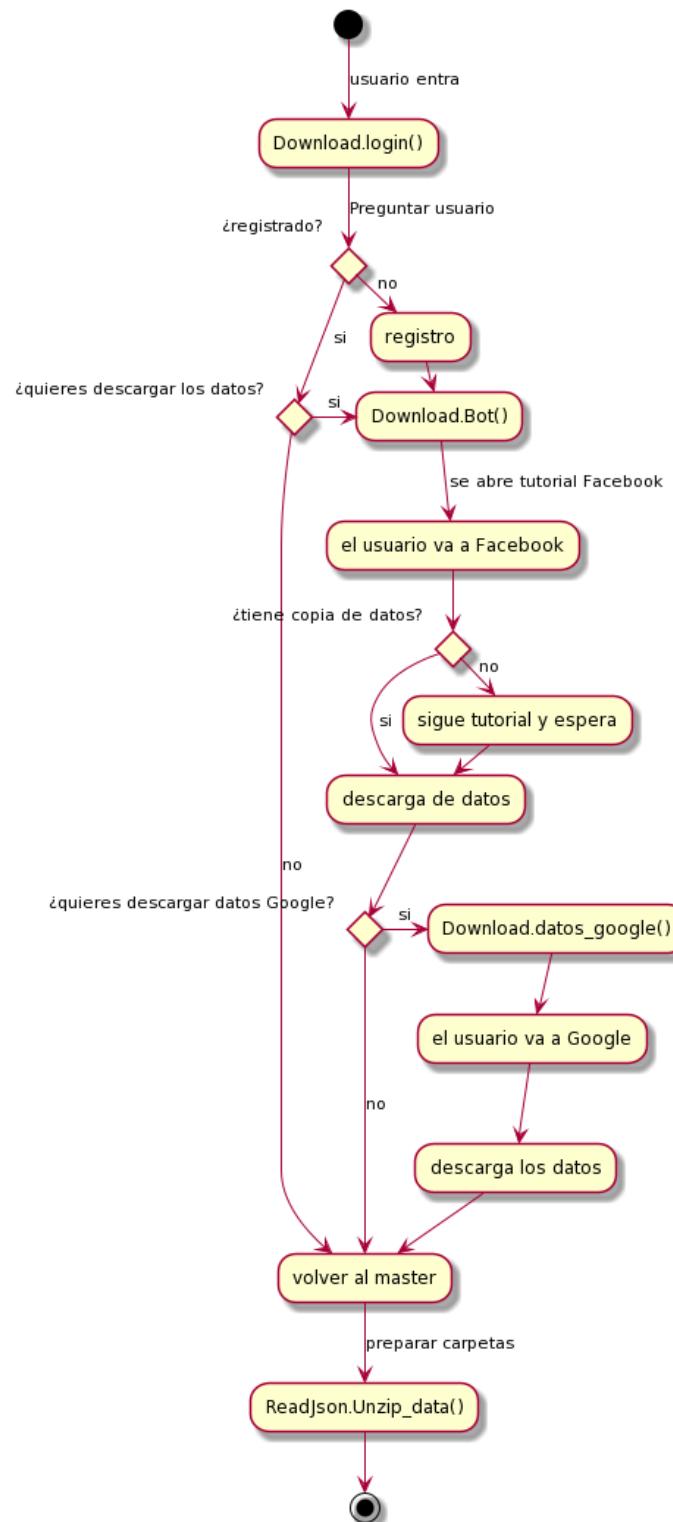


Figura 4.4: Diagrama de actividad para la descarga de datos

Todo esto está gestionado por la función **login**, que es la principal de la clase, simplemente crea la ventana padre en Tkinter y pregunta el correo del usuario que se utiliza como id. Si el usuario accede a descargar datos, se llama a la última función **Bot** dentro del tutorial de descarga. En la versión 2, gestionamos los datos de Google a partir de este punto.

4.1.2. Descarga de datos

Manejada por **Bot** controla toda la función web con Selenium. Empieza borrando los datos existentes en la carpeta de descarga, para evitar datos duplicados o errores si el usuario ha introducido datos externos en ella. A continuación, configura las opciones de Google Chrome para que, entre otras cosas, la descarga se haga automáticamente a la carpeta necesaria y evitar problemas. Después, se manda al usuario a la pantalla de login de Facebook, donde el bot añade directamente el usuario añadido y pide al usuario que añada la contraseña (que en un inicio incluía también el Bot, pero los niveles de seguridad eran menores y decidimos descartarlo). Esta acción redirige al usuario a la carpeta de descarga, donde con la función **put_span_fb**, implementada en JavaScript, se muestra un tutorial en la propia web de Facebook para ayudar al usuario a descargar los datos. Después de todo esto, la función Bot espera a que la carpeta de descarga contenga algún dato y espera un tiempo prudencial para que la descarga se complete, pasando a continuación el control a la fase de preparación de carpetas.

4.1.3. Preparación de carpetas

Una vez con los datos en formato zip en la carpeta adecuada y el navegador cerrado, cosa de la que se ha encargado Selenium, la función **unzip_data** del fichero ReadJsonFacebook.py se encarga de tratar esta carpeta y descargar todos sus datos para su utilización en el siguiente módulo de la aplicación, donde se almacenarán estos datos para su tratamiento.

4.2. Creación de las tablas, parseo de JSON e inserción en las tablas

Una vez realizada la descarga de los datos en formato zip y extraídos los archivos, lo primero que hubo que hacer fue analizar el contenido de estos para ver que datos eran útiles, la carpeta está organizada en distintas subcarpetas (amigos, comentarios, likes, etc.). Estas carpetas contienen los mismos datos ya explicados en el capítulo 3, y cada subcarpeta contiene uno o varios archivos JSON que contiene la información del usuario en ese apartado. Cada una de las subcarpetas que hemos analizado, la hemos trasladado a una o varias tablas en la base de datos. En función de los datos que consideramos útiles dentro de estas carpetas (también explicado en el apartado de datos de Facebook y Google del capítulo 3), íbamos añadiendo los atributos y relaciones de estas tablas. Las etapas de esta sección se pueden ver reflejadas en el diagrama de la figura 4.5.

4.2.1. Creación de tablas SQL

Para este apartado, hemos creado una base de datos relacional, la cual queda ilustrada en el diagrama de la **figura 4.6**, así como en el **Apéndice A**, donde detallamos todas las tablas con su función, un listado de los diferentes campos y una pequeña descripción de lo que guarda cada uno.

El código perteneciente a esta parte del proyecto se encuentra en el fichero `dataBaseCreator.py` que se llama cuando el usuario pulsa el botón de ver libro de vida y antes de inicializar el Back-end para poder visualizarlo.

Como vamos a tener que realizar muchas consultas a la base de datos hemos creado dos funciones que nos ahorran líneas de código:

- **`create_db_connection(host_name, user_name, user_password, db_name)`**: Función que sirve para establecer directamente conexión con la base de datos creada (`db_name`). Es llamada antes de la creación de las tablas y también antes de hacer inserciones.
- **`execute_query(connection, query)`**: Función que ejecuta la consulta pasada por parámetro en forma de string. Es llamada cada vez que queremos crear una tabla, insertar datos o extraerlos para manipularlos.

Las tablas de SQL están detalladas en el Apéndice A.

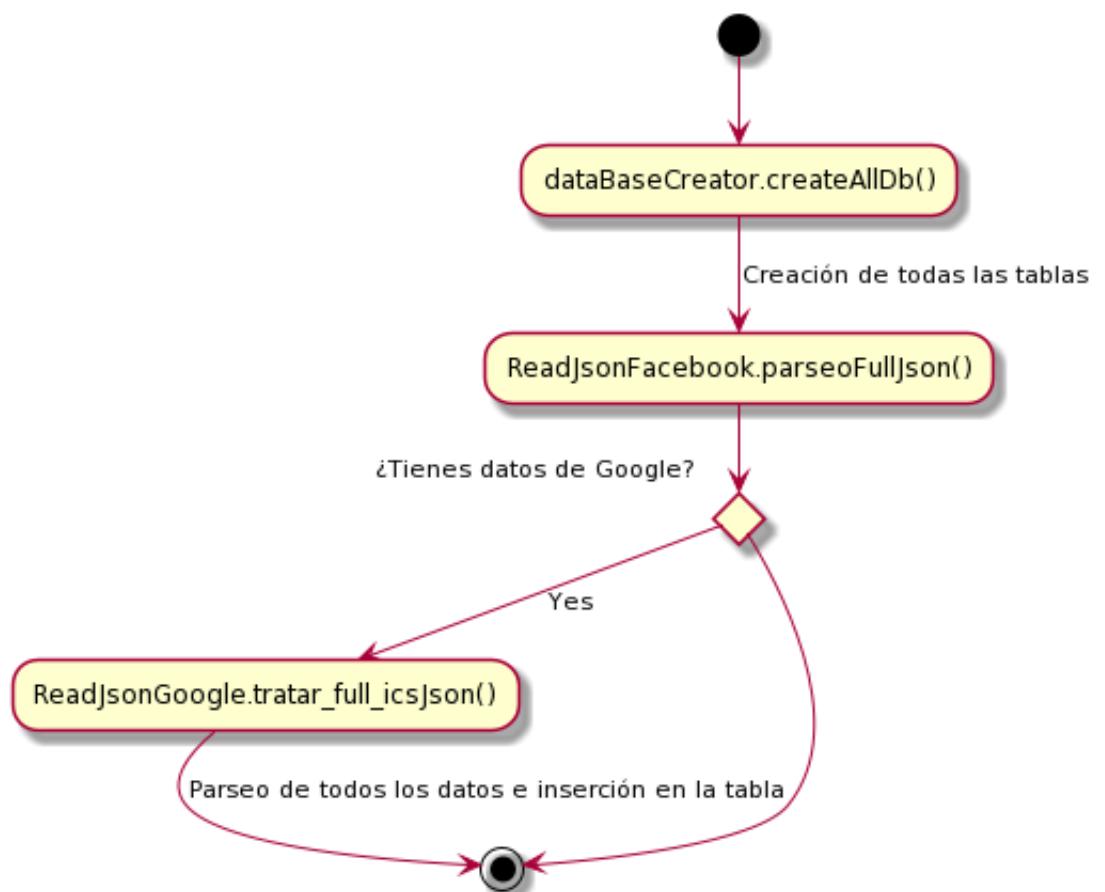


Figura 4.5: Diagrama del parseo de datos

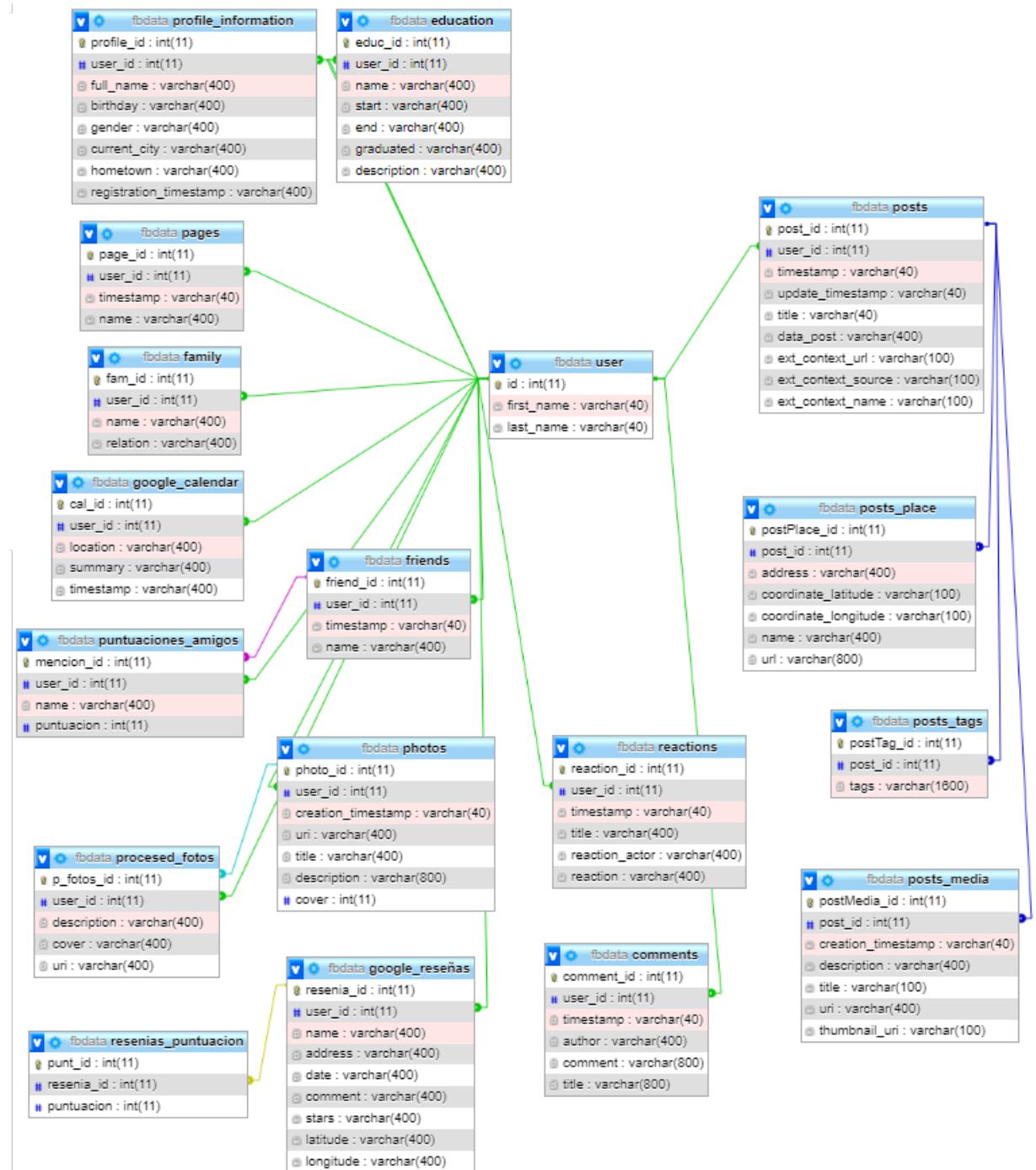


Figura 4.6: Diagrama de la Base de Datos

4.2.2. Parseo de JSON e inserción en las tablas

Para tener los datos del paciente en una base de datos con la que podamos trabajar, primero tenemos que parsear la información del sujeto en cuestión, que viene en formato JSON, ICS, CSV. Esto es, clarificarla y dejarla en un formato que pueda ser leído por las tablas SQL.

JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. ICS es la extensión de los archivos del estándar iCalendar para el intercambio de información de calendarios. Ya que el calendario de Google viene en este formato vamos a transformar a formato JSON para poder tratarla más fácilmente. Todo el código perteneciente a esta parte del proyecto se encuentra en los ficheros ReadJsonFacebook.py y ReadJsonGoogle.py.

En esta parte hemos creado también algunas funciones que nos ahorran líneas de código, como por ejemplo:

- `loadJson(self, path)`: esta función recibe por parámetro la ruta de un fichero JSON el cual decodifica, y devuelve una variable con la que se puede empezar a trabajar en Python.
- `reemplazar(self, s)`: esta función recibe por parámetro un String y lo escapa para que a la hora de insertarlo en la consulta SQL no de problemas.

Para entender mejor esta parte, vamos a describir los distintos ficheros JSON que hemos tratado dentro de la carpeta de Facebook y como se corresponden esos datos con las tablas de la base de datos.

- `friends.json`: Este fichero está formado por un array en el que cada elemento hace referencia a un amigo del usuario en Facebook, cada elemento está formado por una estructura en la que se define el timestamp (la fecha en formato Facebook) de cuando se hicieron amigos, así como el nombre y apellidos del amigo. Estos datos los recogemos y los guardamos en la tabla `fbdata.friends`.
- `comments.json`: Este fichero también está formado por un array en el que cada elemento es un comentario. Cada elemento está compuesto por una estructura formada por el timestamp, el título del comentario y otra estructura en la que se definen el autor y el contenido del comentario. Estos datos son almacenados en la tabla `fbdata.comments`.
- `posts_and_comments.json`: En este fichero se guardan los likes y reacciones del usuario, es también un array en el que en cada elemento se describe el like o reacción del usuario a través de una estructura compuesta por el timestamp, el título (en el que se describe el destinatario

al que se le está dando el like) y una estructura formada por el tipo de reacción y el actor de la misma. Guardamos estos datos en la tabla fbdatabase.reactions.

- profile_information.json: Este fichero es algo más complejo, de aquí se obtiene información que va a distintas tablas. El fichero define una estructura con los campos:

- name: es una estructura formada por el nombre y los apellidos del usuario.
- birthday: es una estructura compuesta por el día, el mes y el año de nacimiento del usuario.
- gender: define el género del usuario.
- current_city: es una estructura que define el lugar donde vive ahora mismo el usuario y un timestamp que define desde cuando.
- hometown: otra estructura más formada por el lugar de nacimiento del paciente y un timestamp que define cuanto tiempo vivió allí.
- registration_time: Es un timestamp con la fecha de registro del usuario en Facebook.

Todos los datos mencionados de profile_information hasta aquí, se guardan en la tabla fbdatabase.profile_information.

- family_members: Es un array en el que cada elemento define una estructura para representar a un familiar del usuario, esta estructura se compone del nombre y apellidos del familiar, la relación parental que hay entre ellos y el timestamp que representa desde cuando son amigos en Facebook. Estos datos se guardan en la tabla fbdatabase.family.
- education_experiences: Es un array donde los elementos que lo componen representan las distintas formaciones educativas del usuario. Cada elemento está constituido por una estructura con campos para definir el nombre del lugar de estudios, la fecha de inicio, la fecha de fin y si está graduado o no. Estos datos se almacenan en fbdatabase.education.
- work_experience: También es un array en el que cada elemento representa un lugar de trabajo. Este lugar de trabajo se define a través de una estructura con el nombre del trabajo, la descripción de lo que se hacía en su trabajo y las fechas de inicio y fin. Al ser muy parecido a education_experience, hemos añadido estos datos en la tabla fbdatabase.education.

- pages.json: Es un array en el que cada elemento describe una página a la que el usuario ha dado like. Cada elemento se compone de una estructura con el nombre de la página y la fecha en la que se empezó a seguir ese contenido. El contenido de este fichero se almacena en la tabla fbdata.pages.
- Fotos de Facebook: Al tener Facebook la posibilidad de guardar las fotos en álbumes, tenemos una carpeta llamada álbum en la que se almacena un archivo JSON para cada uno de los distintos álbumes del usuario, por lo que recorremos todos los JSON que haya en la carpeta para ir recopilando en la tabla fbdata.photos todas las fotos que haya subido el usuario a Facebook. Cada fichero JSON está compuesto por una estructura con el nombre del álbum y un array con las fotos de ese álbum. Este array de fotos está formado por elementos en los que en cada uno hay una estructura para detallar toda la información de la foto, estos campos son la ruta donde está guardada la foto, el título de la foto, la descripción, la fecha de subida y si es portada o no.
- your_post.json: Este fichero está también formado por un array en el que los elementos contienen la información de cada post a través de una estructura. Los campos de esta estructura son la fecha de publicación del post, el título, la descripción, la url externa que se halla compartido en ese post, el nombre del contenido externo unido a este post, un array con los archivos adjuntos (fotos y ubicaciones), y un array con las personas etiquetadas en la publicación. Excepto los archivos adjuntos y las etiquetas se almacena todo fbdata.posts.

Los archivos adjuntos están formados por una estructura con un campo para guardar las fotos y otro para guardar las ubicaciones. El campo para guardar las fotos está formado por un array con todas las fotos adjuntadas al post, cada una se compone de una estructura con los campos título de la foto, descripción, ruta de la foto y la fecha de subida de la foto. Las distintas fotos se guardan en la tabla fbdata.post_media con una foreign key que hace referencia al post al que pertenecen.

Las ubicaciones son similares, pero guardan los campos dirección, latitud, longitud, nombre de la ubicación y una url a la página de Facebook para esa ubicación. Estas se guardan en una tabla llamada fbdata.posts_place con una foreign key que hace referencia al post al que pertenecen.

Las etiquetas son un array en el que cada elemento es el nombre y apellidos del amigo de Facebook que está etiquetado. Se guardan en la tabla fbdata.posts_tags con una foreign key al post al que pertenecen.

El proceso de parseo es similar para los distintos archivos JSON. Primero se carga con la función loadJson() el contenido del fichero JSON en una variable

que nos permita trabajar los datos, después se recorre esta variable con uno o varios bucles, según las profundidades que tenga y se guardan los datos en variables tras ser scapeadas con la función remplazar, para su posterior inserción en la base de datos. Para la inserción se guardan en variables de tipo String las consultas que serán ejecutadas con la función execute_query junto a los datos escapados mencionados anteriormente.

Con esto, los datos del usuario ya estarían a salvo en la base de datos y listos para tratar, de lo cual se encargara la siguiente etapa.

4.3. Tratamiento de los datos

La implementación de esta sección del proyecto se encuentra en el fichero LogicTables.py, y se encarga de recolectar todos los datos almacenados en la BBDD y tratarlos para conseguir un formato entendible y mostrable al usuario. Se divide en esas dos partes: “extracción y tratamiento de datos” y “Envío de datos a la interfaz gráfica de usuario”. Estas últimas tienen la peculiaridad de ser funciones botón, porque se llaman así en la interfaz (boton_accion). Se puede ver las etapas de esta sección en el diagrama de la **figura 4.7**.

A continuación detallaremos las dos partes.

4.3.1. Tratamiento

Además de buscar un formato entendible y mostrable, en esta parte queremos darle un sentido extra a los datos, para ello hemos clasificado los comentarios, las reacciones y likes, y los amigos etiquetados del usuario, para darles una puntuación en *puntuaciones_amigos*, clasificando así los más importantes para el usuario. También hemos cruzado datos de diferentes partes, como tags o menciones, para añadir más valor a las fotos, o, en la segunda versión, datos de Google con datos Facebook.

- Para clasificar los comentarios recorreremos la tabla con un bucle e iremos obteniendo la persona o página a la que va dirigida el comentario por parte del usuario. Para ello nos apoyaremos de la función devuelveNombre() que hemos creado, la cual recibe por parámetro el comentario en tipo string y calcula los caracteres que ocupa el nombre de la persona que publica el comentario, más los caracteres que ocupa el tipo de comentario (ha comentado la publicación de..., ha respondido al comentario de..., ha comentado la foto de..., ha comentado el vídeo de..., ha comentado el enlace de..., etc.) y finalmente recortar el string para devolver el nombre de la persona o página a la que va

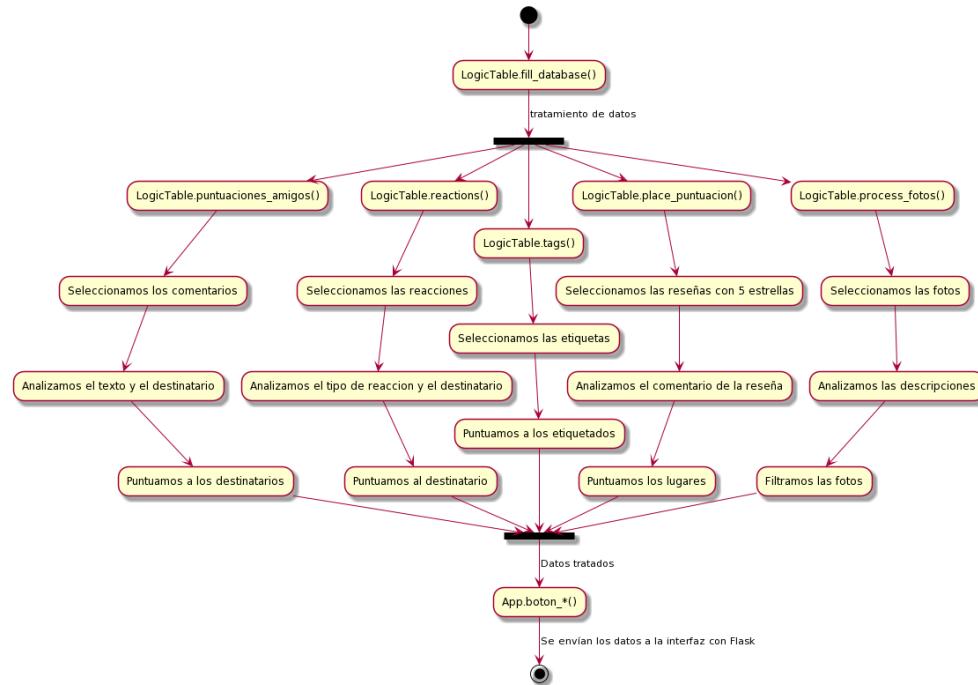


Figura 4.7: Diagrama del tratamiento de datos

dirigido. Una vez obtenido el nombre, si no existe un registro de esta persona en la tabla de *puntuaciones_amigos* lo insertamos con la puntuación a uno, y si ya existe le sumaremos uno a la puntuación que ya tenga calculada.

- Para clasificar los likes y las reacciones seguiremos un procedimiento parecido al descrito en el apartado anterior. Recorremos la tabla *reactions* y recortaremos el string del campo *title* para obtener el usuario al que va dirigido la reacción, también seleccionaremos el campo *reaction* en el que se guarda el tipo de reacción y en función a esta, daremos más o menos puntuación en la tabla *puntuaciones_amigos* a los amigos o páginas involucradas. A continuación se muestra el valor que se le otorga al tipo de reacción:

- LIKE: 1 punto.
- LOVE: 3 puntos.
- JAJA: 2 puntos.
- WOW: 1 punto.
- SAD: -2 puntos.
- ANGRY: -2 puntos.

- Hemos considerado que las personas que salen etiquetadas en las fotos del paciente, por norma general tienen un vínculo importante con este. Por lo que hemos recorrido la tabla *posts_tags* con un bucle, y con ayuda de la función *eval()* del estándar de Python, convertimos en un array el campo *tags*, donde Facebook almacena a las personas etiquetadas en formato de string. Después, recorremos este array de amigos para darles una puntuación en la tabla *puntuaciones_amigos*.
- Otra parte importante del tratamiento es la utilización de un sistema de procesamiento del lenguaje(NLP) para la valoración de las emociones en los textos que hay dentro de toda la aplicación, hemos creado una función *puntuar_texto*, la cual limpia el texto, eliminando las posibles interferencias, quitando mayúsculas, números, o algunos símbolos que no se pueden analizar, mientras que mantenemos otros símbolos o emoticonos. Después traduce el texto al inglés, puesto que en español (como ya hemos mencionado en el capítulo 2) no hay, de momento, herramientas de NPL tan avanzadas como nos gustaría. Por último, lo pasamos por el analizador de sentimientos, que devuelve una puntuación en función de los sentimientos detectados en el texto.

Estos textos analizados nos han sido de gran utilidad en varias partes de la aplicación, ya que nos han permitido evaluar que lugares han sido importantes, que comentarios son los positivos y por lo tanto quienes son los amigos con los que te relacionas más positivamente y otras muchas cosas.

4.3.2. Envió de datos a la interfaz

Esta parte se centra en la organización de los datos ya tratados y almacenados en SQL, los selecciona con diferentes querys para entrelazar los que están relacionados entre ellos y sacar el máximo partido a los mismos. Hemos separado el código en las funciones que posteriormente se llamarán en la GUI. Estas funciones son:

- **boton_amigos_fecha:** Analiza tus 10 mejores puntuaciones entre los amigos que tiene el usuario y devuelve la fecha en la que conoció a esos amigos en la red social en formato JSON: ('fecha': YYYY-MM-DD HH:MM, 'amigo': nombre, 'puntuación': entero). Esta función lee de las tablas friends y puntuaciones_amigos.
- **boton_paginas_fecha:** Analiza tus 10 mejores páginas entre las que el usuario es miembro y devuelve la fecha en la que conociste a tus amigos en formato JSON: ('fecha': YYYY-MM-DD HH:MM, 'página': nombre de página, 'puntuación': entero). Esta función lee de las tablas friends y puntuaciones_amigos.

- **boton_places:** Analiza todos los lugares visitados por el usuario en Facebook y Google devolviendo un JSON con el formato: ('fecha': YYYY-MM-DD HH:MM, 'contenido': contenido del post, 'url': url al contenido del post, 'donde': dirección del lugar, 'personas_list': array de personas con las que apareces, 'fotos_list': array de fotos del lugar, 'coordenadas': coordenadas para buscar en Google y 'puntuación': puntuación que le hemos dado al lugar según la interacción de usuario). Esta función lee de la tabla post_places, resenias_puntuacion y google_reseñas.
- **boton_foto_tag:** Analiza la lista de mejores amigos (puntuación) y manda la URI de los archivos en local que apunta a esas fotos, en las que el usuario aparece con sus mejores amigos, devolviendo: ('amigo': nombre, 'fotos_list': array de fotos, 'fecha': YYYY-MM-DD HH:MM). Esta función lee de las tablas friends y puntuaciones_amigos, posts_tags y post_media.
- **boton_fotos:** Analiza las fotos de portada, que consideramos importantes o con descripción positiva (a través de NLP) y devuelve la URI de los archivos en local que apunta a esas fotos y si son o no portada, devolviendo ('cover': si es cover o no, 'URI': URI a la foto). Esta función lee de la tabla profile_information.
- **boton_informacion:** Devuelve la información básica del usuario: ('nombre': nombre, 'cumple': YYYY/MM/DD, 'género': género, 'vivienda actual': vivienda actual, 'lugar de nacimiento': lugar de nacimiento, 'inicio de los datos': inicio de los datos). Esta función lee de la tabla family.
- **boton_familia:** Devuelve la información de tu familia: ('nombre': nombre del familiar, 'relación': relación de parentesco)
- **boton_logros:** Devuelve tu información académica y laboral: ('nombre': nombre, 'inicio': fecha de inicio del curso, 'fin': fecha de fin del curso, 'graduado': si el usuario está graduado). Esta función lee de la tabla education.
- **boton_fechas:** Devuelve las fechas importantes del programa con todas las actividades que has realizado en ellas, cogiendo datos de varias tablas:
 - Facebook Friends: Coge el timestamp de cuando te has hecho amigo de los 50 mejores amigos que tienes según nuestra puntuación calculada.
 - Google calendar: Coge el timestamp de los eventos importantes que has guardado en Google.
 - Google reseñas: Coge el timestamp de cuando has visitado los

lugares que has añadido a favoritos.

- Facebook Post media: Coge el timestamp de cuando has subido, o has sido etiquetado en fotos importantes.

Esta parte de la aplicación tiene el punto fuerte de unir una gran cantidad de datos, como los de la función que trata los post o los amigos, resumiéndolos en unos pocos datos importantes mucho más fáciles de manejar, o mezclando datos de diferentes fuentes como los datos de todas las fechas de Google y Facebook. Estos datos en el formato manejable son los que se pasan a la interfaz de usuario para que se muestren, lo cual detallamos en el siguiente apartado

4.4. Interfaz de usuario versión 1.

Esta parte de la aplicación estaba comandada por la clase VistaLibroVida() y master() principalmente, en el archivo showLifeBook.py y master.py, se trataba de la parte que implementaba una vista de usuario en tkinter, pero que finalmente fue descartada por diversas razones, entre ellas la dificultad de crear tablas, imágenes y libros de vida en general, con las librerías que proporcionaba tkinter. Utilizábamos principalmente las siguientes:

- ttk: que proporciona acceso a un conjunto de widgets para mejorar el estilo de tkinter y poder definir objetos como treeview, que es el elemento que utilizábamos para la visualización de tablas, pero que está pensado más como un sistema de archivos.
- tkFont: que sirve para manejar fuentes en tkinter y lo utilizábamos para cambiar el tamaño o fuente de las letras en partes como el título o la sección en la que se encuentra el usuario.
- PIL: Es una librería gratuita que permite la edición de imágenes directamente desde Python. Soporta una variedad de formatos, incluidos los más utilizados como GIF, JPEG y PNG. La utilizábamos para mostrar las fotos del paciente, pero tenía problemas al mostrar fotos, enlaces y estilos, por lo que decidimos descartarlo en pos de la aplicación web mucho más amigable que tenemos en la versión actual.

Empezamos por máster, que es la clase que controla el resto de la aplicación, siendo la que posteriormente se puede convertir en un ejecutable de Windows con “pyinstaller”. Esta empieza creando la ventana raíz de la que heredan el resto de ventanas y preguntando al usuario qué acción quiere realizar, estando disponibles las acciones de “Login”, “Crear datos” y “Ver libro de vida”, llamando estas a las respectivas clases que realizan la funcionalidad que describe el botón que se pulsa.

También comprueba que los datos estén disponibles antes de pasar a la siguiente función y se encarga de destruir la aplicación si el usuario cierra la ventana. Esta parte se ha mantenido en la segunda versión de la interfaz, así como la parte de descarga de datos, que ya fue explicada con más detalle en las librerías del capítulo 3. Se han mantenido porque necesitábamos una interfaz Python para tratar y descargar los datos, ya que las librerías y funciones principales estaban escritas en Python, por lo que decidimos no migrarla a React.

A partir de aquí, se descartó todo por la dificultad de crear un libro de vida con Tkinter, es decir, la parte de visualización de datos. Esta parte implementaba una vista de usuario para mostrar los datos ya tratados y formateados. Decidimos centrarnos en la utilización de botones Tkinter para que el usuario seleccione la vista que le interesa, y si esa vista tiene fotos o enlaces externos, el usuario puede seleccionar cuál visualizar por medio de otro botón.

Mientras que el resto de la GUI está gestionada por el formato `pack()` de tkinter, esta parte seguía el formato `grid()`, el cual era más conveniente para la creación de tablas y colocación de objetos en diferentes partes de la ventana. El formato está gestionado por filas y columnas, en las que habíamos colocado los 8 botones para las diferentes posibilidades que tenía el usuario, una tabla que muestra los datos seleccionados y, por último, unos botones de selección para enlaces externos que puedan aparecer en los datos. Para cada sección que puede elegir el usuario, se había definido una función para coger el evento de botón y realizar la acción correspondiente, así como una variable que guardaba los datos para evitar cargas necesarias. Después, se enviaban estos datos a la función principal de Python `create_table()`, que gestionaba toda la parte de borrar datos anteriores para que no se pisaran y separaba el diccionario para mostrarlo como tabla, mostrando desplegables si los datos contienen una lista como puede ser una lista de fotos dentro de un mismo post o una lista de amigos etiquetados en la misma foto.

Todo esto fue descartado por la dificultad de mostrar fotos y por la apariencia más bien primitiva de la app, la cual no nos convenció del todo y creímos que tenía mucho margen de mejora. En la siguiente figura (4.8) se puede ver como era la interfaz antes del cambio de tecnología.

The screenshot shows a window titled 'Este es tu libro de vida' (This is your life book). At the top, there are tabs for 'Info', 'Amigos', 'Paginas', 'Lugares', 'Tag', 'Fotos', 'Familia', and 'Legatos'. The 'Fotos' tab is selected. Below the tabs, there is a table with the following data:

		amigo	fotos	fecha
0	Pablo Cotillas Mena	foto	lista de fotos 1	2015-10-17 01:08
1	Pablo Cotillas Mena	foto	lista de fotos 2	2015-09-21 16:34
2		foto	3	
3		foto	4	
4		foto	5	
5		foto	6	
6		foto	7	
7		foto	8	
8		foto	9	
9	Sergi Ter Cas	foto	lista de fotos 10	2015-10-17 01:08
10	Javier Martinez de Lizarrondo	foto	lista de fotos 11	2015-09-21 16:34
11		foto	12	
12		foto	13	
13		foto	14	
14		foto	15	
15		foto	16	
16		foto	17	
17		foto	18	
18	Javier Martinez de Lizarrondo	foto	lista de fotos 19	2015-09-19 19:33

Figura 4.8: Tabla de la antigua interfaz

4.5. Descarga de datos de Google

Una vez tuvimos una aplicación funcional con los datos de Facebook, creímos oportuno ampliarlos con los datos que almacena otra gran empresa, como lo es Google. Esto lo hicimos porque, aunque Facebook es la red social más utilizada, es posible que no todo el mundo tenga una cuenta, sin embargo es más probable que un paciente haya utilizado alguna vez Google Maps, o haya guardado sus fotos en aplicaciones como Google Fotos, que en este momento viene preinstalada en los móviles Android, ya que los móviles sí que son un elemento prácticamente extendido a toda la población.

Estos datos también nos acarrearon problemas, pues Google protege aún más sus datos frente al scraping, ni siquiera es posible registrarse en Google si estás entrando con un bot que simule a un humano o un navegador automatizado, por lo que hemos delegado completamente la descarga de los datos de Google al usuario de la aplicación, eso si, con un tutorial de ayuda escrito en Tkinter e integrado en nuestra propia aplicación para que esto sea lo más sencillo posible.

Esta parte, de manera similar a los datos de Facebook, se caracteriza por una descarga basada en un tutorial y la preparación de la carpeta, saltándose el paso de login pues el usuario en este punto ya debería estar registrado en la aplicación, pero al obtener nuevos datos, también consta de una parte en la que estos datos se tratan y analizan.

4.5.1. Tutorial de descarga y preparación

Al no poder controlar todas las funciones web con Selenium, no podemos configurar las opciones de Chrome para que la descarga se haga automáticamente a la carpeta necesaria, pero Tkinter si deja abrir navegadores web, por lo que lo primero que se hace es mandar al usuario a la página de Google Takeout, donde se puede descargar sus datos una vez se registre con su cuenta principal de Google. Después la función `datos_google` se encarga de mostrar el tutorial de Tkinter con todos los pasos a seguir por el usuario para descargarse sus datos. Hacemos hincapié en que se guarden en la carpeta descargas, pues no somos capaces de adivinar donde se ha descargado la carpeta de manera simple.

Una vez con los datos en formato ZIP en la carpeta de descargas, la función `unzip_data` del fichero `ReadJsonGoogle.py` se encarga de tratar esta carpeta y descargar todos sus datos para su utilización en el resto de la aplicación, donde pasará por las mismas etapas que los datos de Facebook, se almacenarán para su tratamiento, se tratarán y posteriormente se mostrarán por la nueva GUI. Al haber realizado una aplicación escalable, la cantidad de cambios a realizar con estos nuevos datos no nos ha supuesto una modificación en ninguna parte fundamental del código, solamente añadir las funciones que tratan y analizan los nuevos datos, juntándolos con los de Facebook, así como la inserción de los nuevos JSON en la base de datos

Esta carpeta está compuesta de los datos mencionados en el capítulo 3 donde ya hemos explicado cuáles consideramos útiles, pero vamos a detallar un poco la estructura de los ficheros JSON, ICS y CSV y la correspondencia con las tablas de la base de datos.

- `user_name.ics`: Este fichero lo convertimos en JSON gracias a la librería `Jicson`, que nos genera un array en el que cada elemento corresponde con un evento en el calendario formado por una estructura con la fecha de comienzo del evento, la fecha de fin, la localización, y la descripción. Estos datos son almacenados en la tabla `fbdata.google_calendar`.
- `reseñas.json`: Este fichero se define a través de un array en el que cada elemento hace alusión a una reseña que haya publicado el usuario sobre algún establecimiento. Cada elemento se compone de una estructura en la que el campo “geometría” representa un lugar, que se almacena como la latitud y longitud de este, el campo de la localización es también otra estructura que almacena la URL a `Google_maps`, con la dirección del establecimiento y el nombre del mismo, y por último los campos en los que se almacenan el comentario que se escribió y el número de estrellas con las que se puntuó al sitio. Estos datos se almacenan en la tabla `fbdata.google_reseñas`.

- Para cada foto de Google, hay un archivo JSON en el que se describen la URL, el título, la descripción y el timestamp de la foto, hemos implementado una función que recorre todo el directorio donde están contenidas las fotos y los JSON para recopilar los datos y almacenarlos en la tabla fbdata.photos.
- sitios_favoritos.csv: Cada fila de este fichero referencia a un lugar o establecimiento marcado como favorito por el usuario y cada fila está formada por tres columnas que representan el nombre del lugar, el comentario que se dejó del mismo y la URL. Estos datos son también almacenados en la tabla fbdata.google_reseñas.

4.5.2. Tratamiento y revalorización de los datos

Aunque ya hemos definido los datos que hemos elegido en el capítulo 3, aquí vamos a hacer un análisis más detallado de lo que hemos actualizado en la implementación del proyecto para tratar y analizar los datos, así como el valor que estos datos han aportado a los que ya teníamos en la versión anterior. Como ya hemos dicho, la modularidad de la aplicación nos ha permitido modificar los datos sin cambios sustanciales, todo lo que hemos realizado han sido ampliaciones de los datos.

El primero ha sido la incorporación de las fotos que el usuario tenía en Google, para esto hemos analizado las que el usuario ha guardado como favoritas únicamente, pues la cantidad de fotos que un usuario puede tener en sus archivos es enorme, y también hemos cogido las que el usuario ha guardado como fotos de perfil. Estas fotos las hemos añadido a las que ya tenían de Facebook para mostrar.

El segundo ha sido la creación de una organización por fechas de la que aún no disponíamos, pues los datos del calendario de Google han dado más importancia a este apartado. De Google hemos analizado todos los datos del calendario, así como las fechas en las que según Google el usuario estaba visitando lugares que le resultaron importantes. Estos datos los hemos mezclado con diversos datos de Facebook que ya tenían fechas para crear una visualización ordenada como si fuera un timeline, lo cual es un gran aporte para un libro de vida.

El último ha sido la ampliación de los lugares que ya teníamos de Facebook, puesto que Google Maps guarda una gran cantidad de lugares que el usuario ha visitado y reseñas de los mismos que este ha publicado, hay una gran cantidad de datos que se pueden extraer. Hemos cogido las reseñas de los sitios que ha visitado con la mayor puntuación y pasando el texto de la reseña por nuestro analizador de emociones para guardar en una nueva tabla resenias_puntuacion las reseñas más importantes, y por lo tanto, los lugares

que más le han gustado. También los sitios que el usuario ha guardado en favoritos, pues hemos considerado que son los importantes para él. Todo esto se une con los mismos campos que los que tienen los lugares en Facebook, creando así una cantidad de datos de sitios de usuario muy considerable.

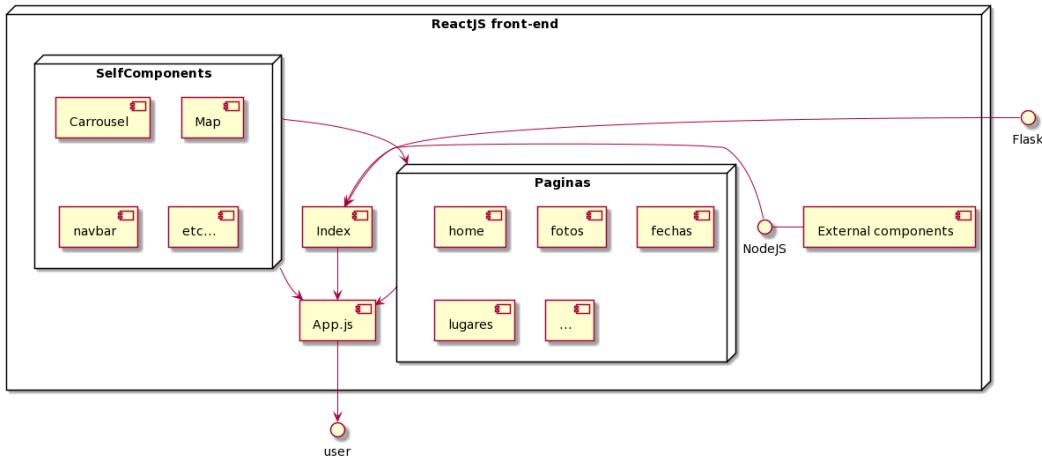


Figura 4.9: Diagrama con los componentes del front

4.6. Interfaz de usuario versión 2

En este apartado vamos a explicar las cuestiones técnicas de la implementación de la segunda versión de la interfaz, la cual es una aplicación web, y de la cual se pueden ver ejemplos en las figuras 4.10 y 4.11, además de ser explicada con todo lujo de detalle en el **apéndice B**, donde hay una guía paso a paso con todas las pantallas que tiene nuestra aplicación.

Teniendo en cuenta la cantidad de problemas que tuvimos con la versión 1 de la interfaz y la utilización de Tkinter para la creación de libros de vida, decidimos investigar más a fondo las posibles tecnologías que nos permitían mostrar los datos de la forma más eficiente y fácil de visualizar para el usuario. Entre las diferentes tecnologías que encontramos, como aplicaciones de escritorio en otros lenguajes, o aplicaciones web, nos acabamos decantando por utilizar una interfaz web con dos partes, un back-end en Python, que potenciado con la librería de Flask tratará los datos y los enviará, con un front-end en React que mostrará todo el contenido de una manera fácil y amigable. Esta parte del front-end queda diseñada en la **Figura 4.9**.

Para la parte del back-end programada en Flask, que funciona creando un archivo app.py que define las rutas de la aplicación de las que luego va a leer React, es decir, la parte del Front. Cada ruta llama a una de las “funciones botón” definidas posteriormente en LogicTables y que ya envían los datos en un formato fácilmente transmitible, por lo que lo único que tuvimos que hacer fue crear el enlace entre las dos partes y configurar los puertos. Con todo configurado, la aplicación de React ya tendrá los datos de las páginas si la parte del Back-end está levantada.

Para la parte del front-end, React ofrece una gran variedad de componentes,



Figura 4.10: Interfaz de usuario, apartado de Página principal

un componente es un elemento software visual, independiente y reutilizable, que tiene su propio estado, recibe unas propiedades e implementa su propia lógica de renderizado. El esqueleto principal de nuestra interfaz consiste en un componente principal Navbar, que es una barra de navegación en formato menú que permite al usuario navegar entre las distintas páginas de las que está compuesta la aplicación, las cuales también se renderizan en sus propios componentes, y luego cada una llama a los elementos visuales más simples, es decir, otros componentes (como mapas, carruseles de galerías...) Cada una de estas páginas finales, se corresponde con los datos proporcionados por cada una de las funciones botón, y también hay una página de Home para recibir al usuario y contarle un poco como funciona la web.



Figura 4.11: Interfaz de usuario, apartado de fotos

Capítulo 5

Conclusiones y Trabajo Futuro

El objetivo principal de este proyecto era, como ya hemos dicho, desarrollar una aplicación que recopile datos, dándoles un sentido extra, para después mostrarlos de una forma que resulte útil para la creación de un libro de vida. Después de implementar toda la aplicación, podemos afirmar que hemos conseguido cumplir en gran medida los objetivos con nuestra solución, ya que finalmente hemos analizado qué redes sociales, y que datos de estas, son los más convenientes para la creación de un libro de vida, descargárnoslos y tratarlos, dándoles aún más sentido y valor, mostrándolos después en una interfaz con el formato de un libro de vida. Esperamos haber contribuido a la sociedad en algo positivo, ayudando a personas que de verdad lo necesitan y facilitando la vida de profesionales como lo son los terapeutas ocupaciones.

En la parte técnica, creemos que los datos que hemos obtenido tienen un valor añadido sobre los trabajos ya existentes en este campo y que cosas como la puntuación de los usuarios o páginas puede ser de mucha utilidad. Como contraparte, sabemos que es prácticamente inevitable perder ciertas funcionalidades pensadas en la idea inicial, como el tratamiento de más datos demasiado grandes para el alcance de este proyecto, aunque hemos intentado mitigar este impacto de la mejor forma posible.

Por otro lado, podemos afirmar sin ninguna duda que este trabajo nos ha servido para crecer como desarrolladores, ya que hemos trabajado partiendo de cero, sin ningún tipo de guión para la realización del proyecto y teniendo que buscar la información y documentación necesaria por nuestra cuenta. Mencionar por supuesto lo orgullosos que estamos de todo lo aprendido con las distintas tecnologías, lenguajes y herramientas empleadas, las cuales estamos convencidos de que volveremos a utilizar en el futuro en nuestros proyectos personales y laborales. También los conocimientos aprendidos en ámbitos como el webScraping, los libros de vida y los problemas de perdida

de memoria, adquiriendo una sensibilización especial con estos problemas tan actuales.

5.1. Trabajo futuro

Durante el trabajo que hemos realizado, hemos dejado funcionalidades o mejoras en el camino, que se podrían realizar en el futuro para mejorar el desempeño de la aplicación, así como cosas a las que es posible que haya que realizarles un mantenimiento periódico:

Descarga de datos

En este apartado, respecto a Facebook, hay poco que creamos que se pueda hacer. Quizás un maestro del Scraping web consiga automatizar todo el trabajo que la empresa ya hace recopilando los datos de sus usuarios. Puede que la mejor forma de ampliarlo respecto a esta red social sea convencer a los familiares y amigos más cercanos para que sumen sus datos a los que ya disponemos y crear así una base de datos mucho más potente.

También se podría considerar la extracción de datos de otras fuentes de datos, como puede ser APIs o trabajos ya existentes en Twitter o los datos que otras aplicaciones diferentes a Facebook y Google recopilan del usuario, que también pueden tener algún valor para la creación de libros de vida, así como los datos de estas que no hemos considerado oportuno tratar, pero pueden aportar valor.

Creación de las tablas, parseo de JSON e inserción

Esta sección se ampliaría a costa de las anteriores, es decir dependiendo del tamaño de las nuevas fuentes de datos que se sumen a la aplicación.

Tratamiento de los datos

Con este apartado estamos contentos en el resultado, aun así, se podría mejorar la técnica de análisis de sentimiento, a costa de tiempo de ejecución o el aprovechamiento de investigación futura en este campo para el idioma español. Como ya hemos dicho en el capítulo 2, esto podría ser algo muy interesante que mirar en el futuro, ya que hay trabajos que ya están consiguiendo analizar los sentimientos de textos en español de forma muy eficaz, y que es posible que añadir esa potencia de clasificación de emociones al proyecto que nosotros hemos llevado a cabo, incremente en gran medida la

puntuación que les damos a elementos como los amigos, los lugares, Etc. además de dar valor a cosas que hemos descartado como los mensajes privados. También, si tuviéramos en cuenta la ampliación en cuanto a la descarga de datos, este apartado también debería crecer en tamaño con respecto a esos nuevos datos descargados, para enlazarlos de nuevas maneras y aportar más valor.

Interfaz de usuario

En la primera versión con Tkinter, la interfaz cumplía su funcionalidad, pero no era lo más vistoso para un usuario, una vez pasado a un formato web, se ha visto una mejora sustancial para la visualización de los datos, pero dada la amplitud de la comunidad de React, es seguro que si se pudiera contar en el equipo con un diseñador y con un experto en React, la aplicación podría llegar a unos niveles superiores. También, aunque la aplicación se ve correctamente en dispositivos móviles, la necesidad de Flask y Python para obtener los datos hace que no se puedan mostrar en estos dispositivos, por lo que una mejora sería añadir toda la información a un servidor, para que se leyieran los datos desde ahí y poder visualizar la aplicación en su formato móvil.

Usabilidad de la aplicación

Somos conscientes de que utilizamos varias tecnologías entrelazadas y que es muy posible que un usuario que no esté familiarizado con la tecnología no pueda instalarse todos los componentes necesarios para su utilización, por lo que se podrían realizar varias mejoras cambiando las tecnologías utilizadas con el fin de unificarlas o aunando todas las que ahora utilizamos. Esta última parte podría realizarse subiendo la aplicación a un servicio de hosting y simplemente habría que hablar con el hosting para que nos instalase las tecnologías y librerías citadas anteriormente. También podría Dockerizarse toda la aplicación para que el usuario simplemente tuviera que instalarse Docker y hacer un Docker push y Docker run para poder utilizarla.

Chapter 5

Conclusions and Future Work

The main objective of this project was, as we have already said, to develop an application that collects data, giving them an extra meaning, and then display them in a way that is useful for the creation of a life book. After implementing the whole application, we can say that we have largely achieved our objectives with our solution, as we have finally analysed which social networks, and which data from these networks are the most suitable for the creation of a book of life, downloading and processing them, giving even more meaning and value to these data, displaying them afterwards in an interface with the format of a book of life. We hope to have contributed to society in a positive way, helping people who really need it and making life easier for professionals such as occupational therapists.

On the technical side, we believe that the data we have obtained has an additional value over existing work in this field and that things like user or page ratings can be of great use. On the other hand, we know that it's practically inevitable to lose certain functionalities thought in the initial idea, such as the processing of more data, which is too big for the scope of this project, although we have tried to mitigate this impact in the best possible way.

On the other hand, we can say without any doubt that this work has helped us to grow as developers, since we have worked starting from zero, without any kind of guide for the realization of the work and having to look for the necessary information and documentation on our own. Of course, we must mention how proud we are of everything we have learned with the different technologies, languages and tools used, which we are convinced that we will use again in the future in our personal and work projects. Also the knowledge learned in areas such as webScraping, life books and memory loss problems, acquiring a special awareness of such current issues.

5.1. Future work

During the work we have done, we have left functionalities or enhancements along the way, which could be done in the future to improve the performance of the application, as well as things that may need to be maintained on a regular maintenance basis:

Downloading data

In this section, with respect to Facebook, there is little that we think can be done. Perhaps a web scraping master will be able to automate all the work the company already does collecting data from its users. Perhaps the best way to scale it up with respect to this social network is to convince family and close friends to add their data to what we already have and create a much more powerful database.

It could also be considered the extraction of data from other data sources, such as apis or jobs already existing in Twitter, or data that other applications other than Facebook and Google collect from the user, which may also have some value for the creation of life books, as well as data from these that we have not considered appropriate to treat, but can provide value.

Tables creation, JSON parsing and insertion

This section would be expanded at the expense of the previous ones, that is, depending on the size of the new data sources added to the application.

Data processing

With this section we are happy in the result, even so, the sentiment analysis technique could be improved, at the cost of execution time or the exploitation of future research in this field for the Spanish language. As we have already said in chapter 2, this could be something very interesting to look at in the future, as there are works that are already managing to analyze the sentiments of Spanish texts very effectively, and that it is possible that adding that power of emotion classification to the project that we have carried out, will greatly increase the score we give to elements such as friends, places, Etc. as well as giving value to things that we have discarded like private messages. Also, if we were to take into account the expansion in terms of data download, this section should also grow in size with respect to that new data, to link them in new ways and provide more value.

User interface

In the first version with Tkinter, the interface was functional, but it was not the most attractive for a user, once moved to a web format, we have seen a substantial improvement for the visualization of the data, but given the breadth of the React community, it is certain that if you could count on the team with a designer and a React expert, the application could reach higher levels. In addition, although the application looks good on mobile devices, the necessity of Flask and Python to obtain the data means that it cannot be displayed on these devices, so an improvement would be to add all the information to a server, so that the data can be read from there and the application can be displayed in its mobile format.

Application usability

We are aware that we use several intertwined technologies and that it is quite possible that a user who is not familiar with technology may not be able to install all the necessary components for its use, so various improvements could be made by changing the technologies used in order to unify them or by bringing together all the ones we now use. This last part could be done by uploading the application to a hosting service and we would simply have to talk to the hosting to install the technologies and libraries mentioned above. The whole application could also be Dockerized so that the user would simply have to install Docker and do a Docker push and Docker run to be able to use it.

Capítulo 6

Contribución individual

A lo largo del desarrollo de todo el trabajo, la carga de tareas ha sido equitativa, trabajando de forma paralela cada uno de los integrantes en distintas tareas o en conjunto a través de aplicaciones como Discord, proporcionándonos en todo momento apoyo mutuo o resolviendo las dudas del otro siempre que se daba la ocasión, si es que disponíamos de ese conocimiento. En las próximas páginas se detalla la aportación individual de cada uno de los autores de este trabajo.

6.1. Cristian Molina Muñoz

6.1.1. Análisis e investigación

Para esta primera etapa, teníamos que recopilar la máxima información para la realización del proyecto de la forma más efectiva posible. Yo ya conocía un poco Python de algún trabajo personal y sabía que tenía una gran potencia para trabajar con grandes volúmenes de datos, por lo que partiendo de eso, me puse a investigar como poder tratar datos de redes sociales y cuáles nos podrían dar mejores resultados. Una vez hecho el análisis sobre los lenguajes a utilizar y alcance, decidimos que Facebook era la red social con más potencial para este tipo de tarea, por lo que me puse a investigar sobre trabajos previos que nos pudieran servir de ayuda y los puntos flacos de estos, a los que pudiéramos aportar más valor. Investigando esto me encontré con Selenium y me pareció una herramienta potente para conseguir datos de usuarios y probar el código de manera automática, por lo que decidimos que era una buena opción para incorporar al proyecto. Por último estuve mirando técnicas de PLN que pudieran ayudarnos a tratar datos y darle un punto único a nuestro proyecto, y viendo la gran potencia de algunas de estas técnicas, decidimos incorporar una de las que nos brindaba la utilidad necesaria, es decir text2emotion. Posteriormente, para la versión 2, al haber trabajado ya con Selenium, investigué las posibilidades para descarga de datos de Google, mientras que la parte de analizar que datos eran válidos la llevamos a cabo en conjunto. En la parte de la interfaz, estuve mirando varios componentes que pudiéramos utilizar, como Gatsby u otras librerías, finalmente encontré la compilación de “awesome-react-components”, de donde sacamos gran parte de los componentes, ya que los listados en ella son de código libre.

6.1.2. Descarga de datos

Este es uno de los apartados con los que más tiempo me he peleado, ya que Facebook tiene un equipo detrás dedicado a evitar estas técnicas, mi primera idea era ir al perfil del usuario sin necesidad de registrarse y scrapear la información de este y todos sus amigos, pero Facebook deja obsoletas este tipo de técnicas con cada actualización, además de que hay usuarios que tienen la cuenta privada. La segunda opción fue scrapear la cuenta del usuario una vez registrada, lo cual daba más margen, aunque seguía el problema del cambio de formato por parte de Facebook. Hecho esto y viendo que los datos que se conseguían eran de una calidad inferior a los que el propio Facebook te proporcionaba, decidimos que lo mejor era utilizar la tecnología para automatizar los pasos de descarga y ayudar así al usuario a proporcionarnos estos datos, teniendo que intervenir solamente en un par de Clicks. Al final,

el proceso consta, como ya hemos explicado, de un tutorial automatizado que deja las carpetas de datos listas para su tratamiento. En esta parte tuve que investigar bastante sobre todo el mundo del scraping web, del cual he aprendido en gran medida, aunque al final la porción de automatización utilizada es menor a lo que me hubiera gustado. Posteriormente, para la versión 2, al haber trabajado ya con la descarga de datos en Facebook y Tkinter, implemente el tutorial para descarga de datos de Google, el cual fue la única manera que encontré para poder descargarlos y adapte los archivos locales para que se pudieran añadir a las tablas en el siguiente apartado.

6.1.3. Creación de las tablas, parseo de JSON e inserción

Con los archivos ya preparados, fue mi compañero el que tuvo un papel principal en su tratamiento y conversión a formato SQL, siendo yo solamente partícipe en la creación de alguna función auxiliar y resolución de algunas dudas. Dicho esto, no he sido ajeno a esta parte, pues los datos que almacenábamos también iba a tener que tratarlos yo más adelante, por lo que tuve que estar al tanto de todo el proceso, incluido la ampliación de la versión 2, en las que aporte tratando los datos en formato .csv, entre otros aportes.

6.1.4. Tratamiento de los datos

En este apartado de la aplicación, mi aportación fue, entre otras, la creación de las funciones “botón”, es decir, las que seleccionaban los datos de las tablas SQL finalmente tratadas y transformaban ese formato en un diccionario que la GUI de Tkinter pudiera manejar más fácilmente, y con campos más manejables para el usuario, como el paso del timestamp de Facebook y Google a un formato de fecha entendible. La parte de tratamiento del lenguaje natural y análisis de emociones para la creación de la puntuación fue un apartado que nos repartimos entre ambos y del que colabore en todas las partes. Para la segunda versión, cree nuevas “funciones botón” con, por ejemplo, las fechas y actualice las existentes para que manejaran también los datos de Google, así como cambiar su forma de retornar datos para que se pudiera adaptar a la interfaz

6.1.5. Interfaz de usuario

Este apartado fue otra de mis puntas de lanza en la primera versión del proyecto, descubrí Tkinter como una librería con la que crear GUIs fácilmente y que podían resultar sencillas para la interfaz de nuestra aplicación. La empecé a utilizarla para que el usuario pudiera registrarse en la base de datos SQLite y lo acabe extendiendo a todo el proceso. Pero nos dimos

cuenta de que tiene limitaciones que me dieron bastantes problemas, como la visualización de tablas (la cual es mucho más difícil de lo que parece en un primer momento y ha cambiado durante el desarrollo, pasando por HTML integrado en Tkinter, pasando por la librería Treeview de Tkinter y terminando en la versión 2 con React), ha sido de gran ayuda para comunicar todas las partes de la aplicación, para mostrar mensajes de error al usuario, o para esperar a que se completen acciones. Para la versión 2 de la interfaz, implemente la parte de comunicar el Back-end de Flask con el Front-end de React. En este apartado tuve que aprender React de cero, el cual me ha parecido una de las herramientas más útiles de JavaScript y que seguro que usaré en el futuro para la creación de webs. Utilice estos conocimientos para la adaptación e implementación de algunos componentes, como las tablas o botones, también realice la parte de ajustar todos los datos que venían de Flask, ya en JavaScript, para las especificidades que cada componente necesitaba, así como aportar en la parte de diseño en css y en los propios atributos de los componentes.

6.1.6. Memoria

Sobre esta apartado, por mi lado, tuve que aprender a utilizar L^AT_EX, el cual no conocía y me ha parecido de gran utilidad para la creación de documentos y memorias. Mi aportación a este documento ha sido plasmar en papel las partes en las que más he contribuido en el resto del proyecto, pues cada unos hemos escrito principalmente lo que mejor se nos daba y a lo que más tiempo le hemos dedicado, tanto en la primera versión de la aplicación como en la segunda.

6.2. Pablo Aguilera Heredero

6.2.1. Análisis e investigación

En esta primera etapa comencé buscando información sobre la perdida de memoria. Ya que teníamos que realizar un libro de vida partiendo de datos extraídos de redes sociales para ayudar a gente con estos problemas, lo primero que hice fue buscar sobre que media de edad empiezan a aparecer estos problemas para después elegir que red social es la más utilizada por gente de esta edad y así que nuestra contribución a la sociedad con este proyecto fuese útil. Estuve valorando varias redes como Twitter, Tiktok e Instagram, pero finalmente después de buscar información me decante junto a mi compañero Cristian por la red social Facebook. Me descargué mis propios datos de Facebook para analizar el contenido de los ficheros y así averiguar cuál de los formatos que nos ofrecía Facebook era el más fácil de tratar con las tecnologías con las que barajábamos desarrollar el proyecto. Después de que decidí que el formato más adecuado era el JSON analicé en detalle los ficheros para hacerme una idea de que datos eran los más útiles para la creación de un libro de vida y empecé a buscar información sobre como debía decodificarlos y codificarlos para empezar a tratarlos. En la segunda versión del proyecto, nuestros profesores nos dieron algunas recomendaciones como tratar los datos de Google o realizar la interfaz gráfica con React y Flask, por lo que realicé el mismo proceso de análisis de los ficheros que hice con Facebook, con los ficheros de Google. También al no haber trabajado nunca con React ni con Flask tuve que documentarme y buscar información sobre cómo se utilizan estas tecnologías.

6.2.2. Descarga de datos

Aunque en este tema se ha centrado más mi compañero, enterarme era una necesidad primordial para poder desarrollar mis tareas asignadas. Al yo centrarme más en el siguiente punto mi aportación aquí fue implementar un código que trajese el zip de los datos de Facebook y lo guardase en una carpeta dentro del proyecto, para yo poder empezar a trabajar con los datos sin tener que esperar a que estuviese terminada esta parte.

6.2.3. Creación de las tablas, parseo de JSON e inserción

Esta ha sido una de mis principales contribuciones al proyecto, en esta etapa me encargué primero de implementar las funciones de conexión y ejecución de consultas a la base de datos, y después me encargué de crear todas las tablas en las que íbamos a almacenar los datos de Facebook. Una vez creadas

analicé todos los JSON para hacerme una idea clara de dónde estaban los datos útiles que queríamos almacenar para después ponerme a implementar las funciones que se encargasen de parsearlos. A la vez que iba creando estas funciones de parseo iba elaborando las consultas SQL que se encargaban de insertar los datos en la base de datos. Respecto a este apartado añadir que he ampliado mis conocimientos a la hora de trabajar con datos JSON así como mis conocimientos en bases de datos, también mencionar que he aprendido mucho del lenguaje Python con el cual nunca había trabajado y del que me he forjado una idea muy positiva para futuros proyectos personales. En la segunda versión también me encargué de crear todas las tablas que íbamos a utilizar para guardar estos nuevos datos de Google así como de implementar las funciones que recorriesen los ficheros JSON y almacenasesen los datos.

6.2.4. Tratamiento de los datos

En este apartado mi aportación fue la elaboración de las funciones que se encargan de recorrer las tablas con los datos recogidos para darles un sentido extra. Para ello cree las funciones que obtenían los nombres de los amigos o páginas a los que iban dirigidos los comentarios del usuario, así como los likes y reacciones que publicó. Una vez obtuve estos nombres les fui dando una puntuación para establecer una tabla con los vínculos más fuertes del paciente. También analicé la tabla de las personas etiquetadas en las publicaciones del usuario para darles una puntuación. Para ello también me encargué de crear las tablas necesarias en la base de datos para guardar estos nuevos datos tratados así como de elaborar las consultas de inserción. En la segunda versión me encargué de realizar una funcionalidad parecida, pero con los datos de Google, seleccionando por ejemplo las reseñas con mejor puntuación de los establecimientos a los que había ido el usuario y luego pasando los comentarios por el analizador de texto de Python text2emotion. Una vez ya tenía estos datos tratados me encargué de crear las tablas de la base de datos donde almacenarlos cruzándolos con los de Facebook.

6.2.5. Interfaz de usuario

En este apartado se ha centrado más mi compañero, pero al igual que en el de descarga de datos he tenido la necesidad de aprender cómo se usa la tecnología Tkinter, ya que me parece una forma bastante sencilla de crear interfaces gráficas. Posteriormente en la segunda versión aunque mi compañero se centró más que yo en como enviar los datos a la interfaz con Flask, tuve la obligación de enterarme del total funcionamiento que le dábamos a esta librería para poder coger los datos con los componentes de React. También

me encargué sobre todo de implementar los componentes de React como el timeline, las tablas, las galerías, el mapa o la barra de navegación.

6.2.6. Memoria

Para este apartado como yo ya había trabajado con Latex en el editor de texto Overleaf en la asignatura Desarrollo de Sistemas Interactivos, ya tenía un conocimiento mínimo de los comandos, por lo que me encargue de generar un esqueleto a la plantilla proporcionada por nuestros tutores. Una vez ya estaban definidos los capítulos que iban a componer esta memoria nos los repartimos en función de lo que mejor se nos daba a cada uno y a lo que más tiempo le habíamos dedicado para poder agilizar el trabajo.

Apéndice A

Tablas de SQL

A continuación vamos a listar todas las tablas que hemos creado en SQL para almacenar los datos del paciente y qué es lo que se guarda en cada campo.

- **Tabla POST:** En esta tabla se guardan datos correspondientes a las publicaciones en los tablones de Facebook por parte del paciente. Sus campos son:
 - *post_id*: Es la clave primaria de la tabla para establecer un identificador de post.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece este post.
 - *timestamp*: Es la fecha en la que se publicó el post, guardada en formato de marca temporal. La marca temporal es el tiempo en segundos que ha pasado desde epoch (1-1-1900).
 - *update_timestamp*: Es la fecha de modificación del post en el formato marca temporal.
 - *title*: Título del Post.
 - *data_post*: Es el texto agregado al post.
 - *ext_context_url*: Es la Url a contenido externo de Facebook (publicación de Instagram, artículos de periódicos...).
- **Tabla POST_MEDIA:** En esta tabla se guardan los datos correspondientes a los archivos adjuntos al post. Suelen ser fotos.
 - *post_media_id*: Es la clave primaria de la tabla para establecer un identificador del archivo adjunto.

- *post_id*: Es la clave foránea que referencia a qué post pertenecen este archivo.
 - *creation_timestamp*: Es la fecha en la que se subió el archivo que se adjunta al post guardada en formato de marca temporal.
 - *description*: Texto relacionado con el archivo adjunto.
 - *title*: Título del archivo adjunto.
 - *uri*: Es el enlace a la carpeta donde está el archivo adjunto.
- **POST_PLACE**: En esta tabla se guardan los datos referentes a la ubicación del post.
 - *post_place_id*: Es la clave primaria de la tabla para establecer un identificador de la ubicación el post.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece este post.
 - *address*: Es la dirección de la ubicación.
 - *coordinate_latitude*: Es la latitud de las coordenadas.
 - *coordinate_longitude*: Es la longitud de las coordenadas.
 - *name*: Es el nombre del lugar.
 - *url*: Es la Url a la página que tiene Facebook creada de ese lugar.
 - **POST_TAGS**: Son las personas que están etiquetadas en el post.
 - *post_Tag_id*: Es la clave primaria de la tabla para establecer un identificador de las etiquetas del post.
 - *post_id*: Es la clave foránea que referencia a qué post pertenece la gente etiquetada.
 - *tags*: Son las personas etiquetadas en el post.
 - **FRIENDS**: En esta tabla se guardan todos las personas que el paciente tiene agregadas en Facebook como amigos.
 - *friend_id*: Es la clave primaria de la tabla para establecer un identificador a los amigos del paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece este amigo.
 - *timestamp*: Es la fecha en la que se inició la relación por Facebook entre el amigo y el paciente guardada en formato de marca temporal.

- *name*: Es el nombre del amigo.
- **REACTIONS**: En esta tabla se guardan las reacciones (LIKES, LOVE...) que ha tenido el paciente a comentarios, publicaciones, páginas, enlaces o fotos.
 - *reaction_id*: Es la clave primaria de la tabla para establecer un identificador de la reacción del paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta reacción.
 - *timestamp*: Es la fecha en la que el paciente reaccionó.
 - *title*: En el título se indica a qué o a quien ha reaccionado el paciente.
 - *reaction*: Es el tipo de reacción que ha tenido el paciente (Like, Love, Care, Haha, Wow, Sad and Angry).
- **COMMENTS**: En esta tabla se guardan los comentarios realizados por el paciente en Facebook.
 - *comment_id*: Es la clave primaria de la tabla para establecer un identificador del comentario escrito por el paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece este comentario.
 - *timestamp*: Es la fecha en la que el paciente redactó el comentario.
 - *title*: En el título se indica a qué o a quien ha comentado el paciente.
 - *comment*: Es el texto que escribió el paciente.
- **PHOTOS**: En esta tabla se guardan las fotos subidas por el paciente a la red social.
 - *photo_id*: Es la clave primaria de la tabla para establecer un identificador de la foto subida por el paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta foto.
 - *creation_timestamp*: Es la fecha en la que el paciente subió esta foto.
 - *uri*: Es el enlace a la carpeta donde está almacenada la foto.
 - *cover*: Valor que determina si es la foto de portada del álbum.
 - *title*: En este campo se indica a qué álbum pertenece la foto.

- *description*: Pie de foto.
- **PAGES**: en esta tabla se guardan las páginas de Facebook que sigue el usuario.
 - *page_id*: Es la clave primaria de la tabla para establecer un identificador de la página que sigue el paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta página.
 - *timestamp*: Es la fecha en la que el paciente empezó a seguir esta página.
 - *name*: Es el nombre de la página.
- **PUNTUACIONES_AMIGOS**: En esta tabla se guarda una valoración de las páginas y los amigos con los que el paciente más interactuaba ya fuese con comentarios reacciones o fotos en las que salen juntos.
 - *mencion_id*: Es la clave primaria de la tabla para establecer un identificador del amigo o página que tiene relación con el paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta puntuación.
 - *puntuación*: Es la puntuación con la que se valora la interacción que tenía el amigo o la página con el paciente.
 - *name*: Es el nombre de la página o del amigo del paciente.
- **PROFILE_INFORMATION**: En esta tabla se guardan los datos básicos del usuario que tiene publicados en Facebook.
 - *profile_id*: Es la clave primaria de la tabla para establecer un identificador del paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta información.
 - *full_name*: Es el nombre completo del paciente.
 - *birthday*: Es la fecha de nacimiento del paciente.
 - *gender*: Es el género del paciente.
 - *current_city*: Es la ciudad en la que actualmente reside el paciente según Facebook.
 - *hometown*: Es la ciudad origen del paciente.
 - *registration_timestamp*: Es la fecha de registro del paciente.

- **FAMILY:** En esta tabla se guardan los amigos que tienen un vínculo familiar con el paciente.
 - *fam_id*: Es la clave primaria de la tabla para establecer un identificador del familiar del paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece este familiar.
 - *name*: Es el nombre del familiar.
 - *relation*: Es la relación familiar entre el amigo y el paciente.
- **EDUCATION:** En esta tabla se guarda los lugares en los que el paciente se formó académicamente y laboralmente.
 - *educ_id*: Es la clave primaria de la tabla para establecer un identificador de los datos de formación paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta información.
 - *name*: Es el nombre del lugar donde estudió, se formó o trabajó el paciente.
 - *start*: Es la fecha en la que inicio esta experiencia.
 - *end*: Es la fecha en la que terminó esta experiencia.
 - *graduated*: Valor que determina si se graduó en la formación.
 - *description*: Campo que especifica que era lo que llevaba a cabo el paciente.
- **GOOGLE_CALENDAR:** En esta tabla se guardan los eventos fijados por el usuario en el calendario de Google.
 - *cal_id*: Es la clave primaria de la tabla para establecer un identificador de los datos de eventos del paciente.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta información.
 - *Location*: Es el sitio donde tuvo lugar el evento.
 - *Summary*: Es la descripción del evento.
 - *Timestamp*: Es la fecha en la que se fijo el evento.
- **GOOGLE_RESEÑAS:** En esta tabla se guardan las reseñas publicadas en Google por el usuario.
 - *resenia_id*: Es la clave primaria de la tabla para establecer un identificador de las reseñas publicadas en Google por el paciente.

- *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta información.
 - *Name*: Nombre del establecimiento sobre el que se publicó la reseñas.
 - *address*: Es la dirección del establecimiento.
 - *date*: Es la fecha en la que se publicó la reseña.
 - *comment*: Almacena el comentario que el paciente puso del establecimiento.
 - *stars*: Son las estrellas con las que se calificó al establecimiento.
 - *Longitude*: Coordenadas del establecimiento.
 - *Latitude*: Coordenadas del establecimiento.
- **RESENIAS_PUNTUACION**: En esta tabla se guardan las reseñas más destacadas pasadas por nuestro analizador de texto.
 - *punt_id*: Es la clave primaria de la tabla para establecer un identificador a las reseñas más destacadas.
 - *resenia_id*: Es la clave foránea que referencia a qué reseña se refiere.
 - *puntuacion*: Son los puntos devueltos por nuestro analizador de sentimiento.
 - **PROCESED_FOTOS**: En esta tabla se guardan las fotos.
 - *p_fotos_id*: Es la clave primaria de la tabla para establecer un identificador a las fotos.
 - *user_id*: Es la clave foránea que referencia a qué usuario pertenece esta información.
 - *description*: Es el comentario que añadió el paciente a la foto.
 - *cover*: Valor que determina si la foto es de portada o no.
 - *uri*: Es la uri a la carpeta donde está el jpg.

Apéndice B

Manual de usuario y casos de uso

En este capítulo realizaremos un recorrido guiado sobre el uso de nuestra aplicación, así como los requisitos que el usuario tiene que tener instalados antes de poder utilizarla. Explicaremos paso a paso los programas que nosotros hemos utilizado, mencionando posibles alternativas que podrían sustituirlos, así como todas las opciones que hay a disposición del usuario en la parte de la aplicación, adjuntando también capturas de pantalla de cada una de las acciones a realizar.

B.1. Guía para la instalación

En este apartado explicamos los requisitos previos de la aplicación, es muy recomendable que esta instalación se lleve a cabo por una persona con conocimientos informáticos, ya que puede ser un poco compleja para una persona que no tenga una base en cuanto al uso de Python o la línea de comandos. Vamos a enumerar las herramientas, tecnologías y librerías necesarias para poder correr la aplicación en local. Vamos a enumerar las que nosotros hemos utilizado, pero cualquier aplicación con su misma funcionalidad podría servir. En primer lugar debes instalarte el servidor local **Xampp** para que la aplicación pueda conectarse con la base de datos. A continuación debes instalar la herramienta **Visual Studio Code** con la que correrás nuestro código. Una vez te la hayas instalado debes descargarte el intérprete **Anaconda** para poder instalar las siguientes librerías Python con estos comandos:

- pip install tk
- pip install sqlite
- pip install selenium

- pip install googletrans
- pip install text2emotion
- pip install zipfile38
- pip install inspect-it
- pip install python-time
- pip install pytest-shutil
- pip install stats
- pip install mysql-connector-python
- pip install jsonlib
- pip install webbrowser
- pip3 install jicson
- pip install python-csv

Una vez realizadas estas instalaciones para la parte del back-end, debes instalarte **node.js** para el front-end, y una vez lo tengas (lo cual te permitirá usar ‘npm’ para instalar componentes react), busca en la barra de Windows la consola PowerShell para instalar los siguientes componentes de React:

- npm install
- npm i -S react-router-dom
- npm i -S react-router
- npm i -S react-photo-gallery
- npm i -S react-images
- npm i -S simple-react-google-maps
- npm i -S react-circle-modal
- npm i -S react-vertical-timeline-component
- npm i -S rsuite-table
- npm i -S react-awesome-button
- yarn add react-router-dom
- yarn add react-router
- react-photo-gallery
- yarn add react-images

- yarn add simple-react-google-maps
- yarn add react-circle-modal
- yarn add react-vertical-timeline-component
- yarn add rsuite-table
- yarn add react-awesome-button

Una vez tengas todo esto instalado debes arrancar Visual Studio Code e importar el proyecto o descargarlo de nuestro Github, para ello, una vez lo tengas en local, selecciona “abrir”, “Proyecto existente” y busca en tu explorador de archivos el proyecto. Despues selecciona el archivo master.py y ejecuta el código con Python o python3 (el botón de play verde, si usas VScode). Te saltará una ventana, pero todavía tienes que abrir otros archivos. Despues, hemos proporcionado un archivo para PowerShell que ejecuta la aplicación en yarn, el cual se llama “powershell.ps1” solo tienes que ejecutarlo como administrador y él hará el resto, también puedes iniciar yarn directamente desde la consola. Por ultimo, para tratar los datos, deber iniciar los apartados de Apache y MySQL de Xampp, para que se puedan tratar los datos. Con esto ya estaría todo listo para que empieces a trabajar con nuestra aplicación siguiendo el manual de usuario.

Recuerda tener disponible también un navegador web, recomendablemente Google Chrome, para poder descargar los datos de las redes objetivo y posteriormente correr la aplicación web y visualizar los datos.

B.2. Manual de usuario

En este apartado vamos a ver cuál es la forma de utilizar la aplicación una vez instaladas todas las dependencias, la cual consta de cuatro fases diferenciadas, el login en la aplicación de escritorio, la descarga de los datos de Facebook y Google, el tratamiento de los datos por parte de nuestro programa y por último la visualización de los mismos a través de la interfaz web. A continuación los explicaremos en más detalle:

B.2.1. Bienvenida y login

Esta es la primera ventana con la que se encuentra el usuario, consta de tres botones, uno de descarga, donde el usuario se va a registrar y descargar sus datos para que podamos tratarlos, otro de creación de las bases de datos, así como la visualización de los mismos y un último botón para salir de la aplicación:

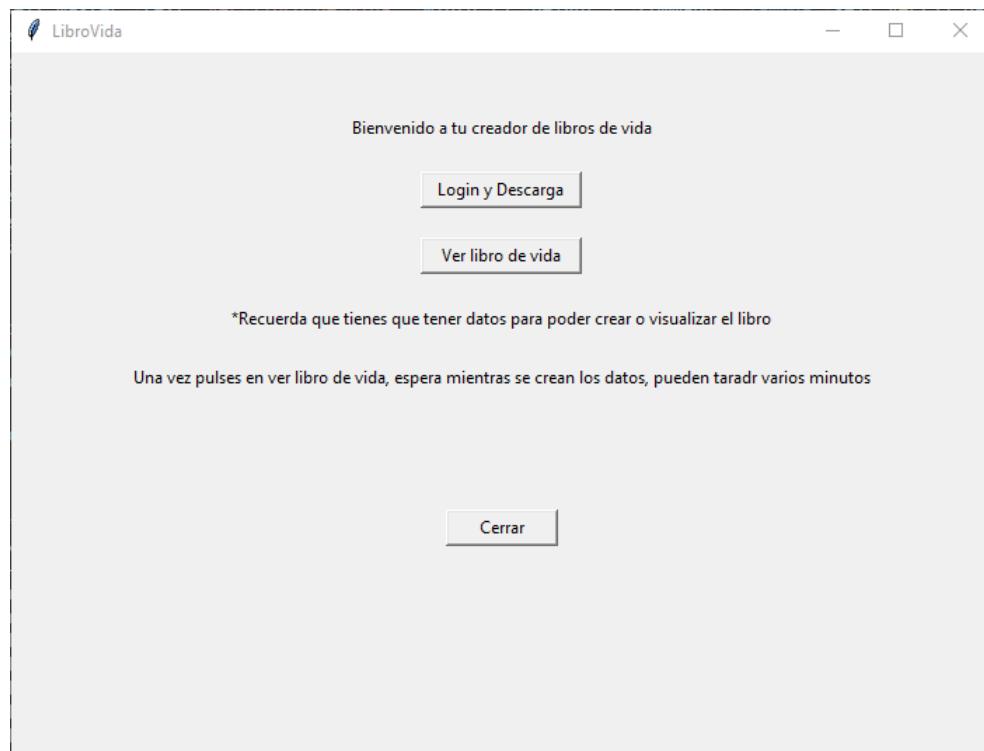


Figura B.1: Inicio de sesión



Figura B.2: Login para descarga

Si intentas crear datos o visualizarlos antes de tenerlos descargados, la aplicación es posible que falle, por lo que si no tienes las carpetas de datos (Carpeta_G y Carpeta_FB) ya creadas, lo primero va a ser la descarga. Para ello, el botón de “Login y descarga” te lleva a la pestaña donde comprobamos si ya te has descargado los datos, donde se descargarán primeramente los de Facebook y después pasaremos a los de Google:

B.2.2. Descarga de Datos

Esta parte está comandada por Selenium, una aplicación que, manejando una ventana de Google Chrome, para ayudarte a descargar los datos de Facebook. Esta ventana se abrirá directamente en la página de Login de Facebook, en la que te tendrás que registrar, Selenium ya te facilita el mismo correo que has añadido en el registro de la aplicación para ahorrar tiempo al usuario. Este Login se puede ver en la **figura B.1 y B.2**.

Así solo tendrás que escribir tu contraseña y Selenium te enviará directamente a la página de descarga, donde te mostrara un vistoso tutorial que te indica los pasos a seguir (Mostrados en la **figura B.4**), los cuales son:

- Cambiar el formato de HTML a JSON
- Pulsar el botón de “Crear archivo”

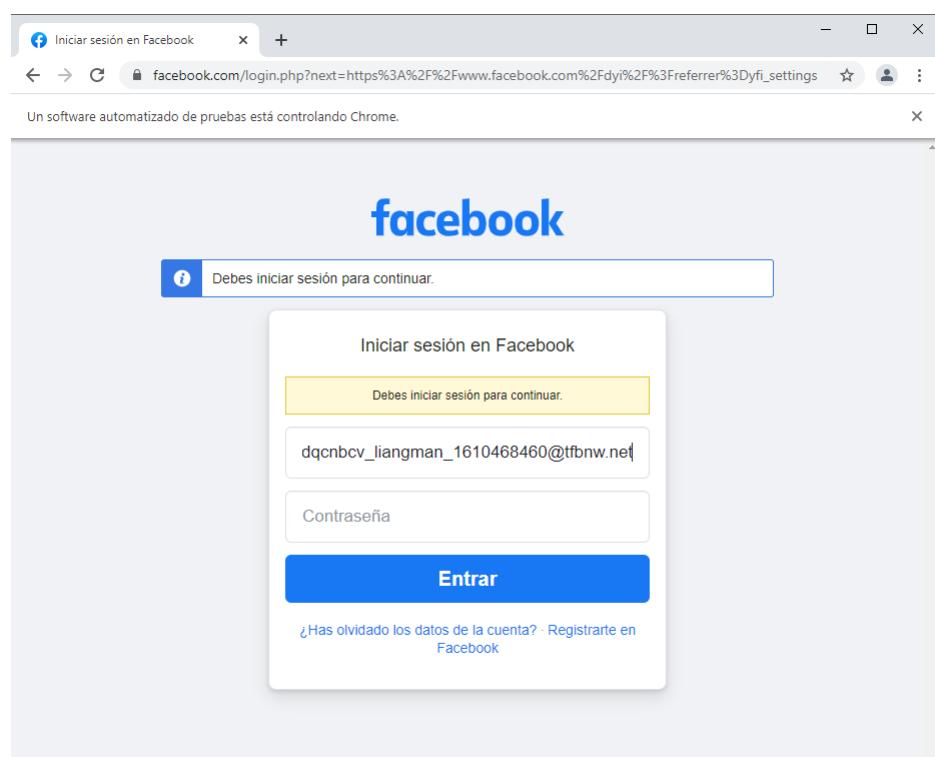


Figura B.3: Login Facebook con el correo

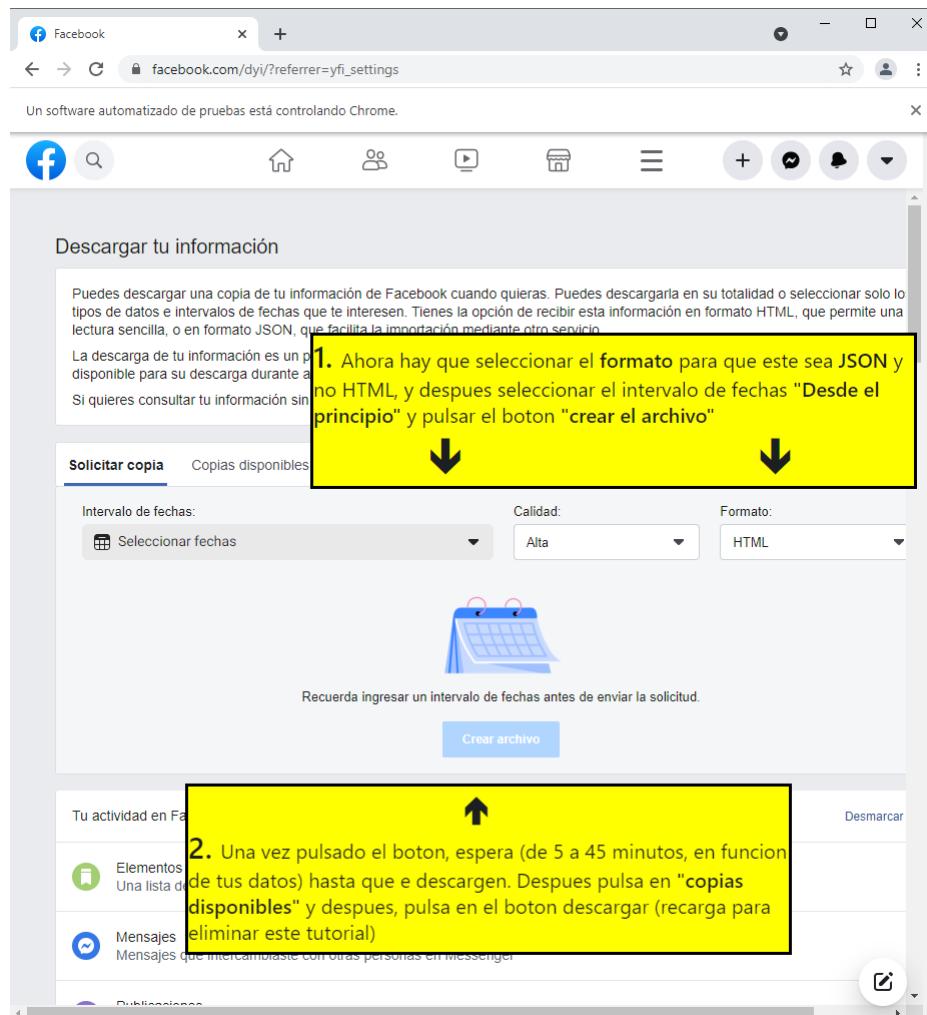


Figura B.4: página de descarga de datos

- Esperar hasta que haya una copia disponible
- Pulsar en “Copias Disponibles” y después en “Descargar”

Una vez pulses el botón de descarga en la copia que se acaba de crear, la aplicación hará el resto. ¡No cierres la ventana! Se cerrará automáticamente.

Ten en cuenta que el tutorial puede variar, ya que Facebook esta siempre en constante cambio, si esto ocurre, te rogamos que se lo hagas saber al responsable de la aplicación.

Una vez Facebook está descargado, se te abrirá una nueva ventana de la aplicación donde se te explicara como descargar los datos de Google, en el que tendrás que pulsar “Descargar datos Google” si deseas que se descarguen,

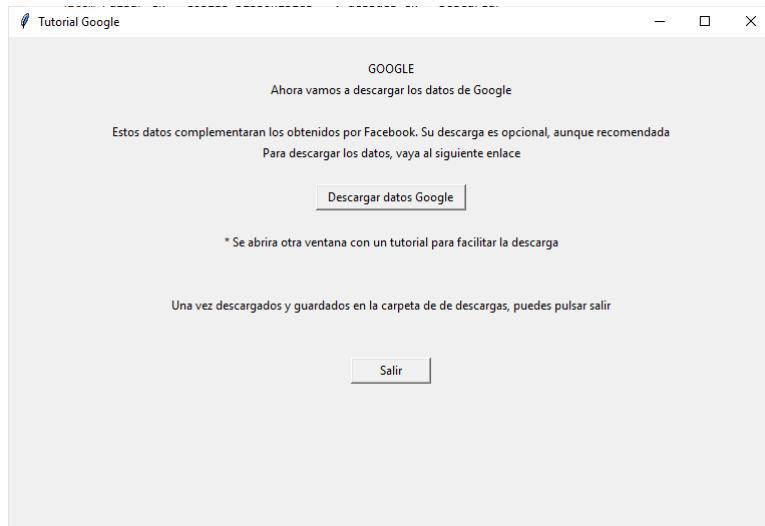


Figura B.5: Tutorial de Google

este se puede ver en las **Figuras de B.5 a B.9**:

Al iniciar este tutorial de Google se deberían haber abierto dos cosas, un navegador con la página de descarga y una nueva ventana de la aplicación con un tutorial donde se detalla paso a paso los requerimientos necesarios para descargarlos. Google bloquea Selenium, por lo que esta vez será el usuario que está leyendo esto el que tiene que realizar todos los pasos en la página de descarga.

A continuación mostramos algunas páginas del tutorial:

Una vez hecho esto, hay que esperar a que los datos se descarguen, tienen que estar en la carpeta de descargas, si no no se podrán analizar. Después de eso ya se puede cerrar el tutorial y tendríamos todos los datos que necesitamos descargados.

B.2.3. Creación de las tablas

Esta Sección no necesita más acción por parte del usuario que pulsar el botón de “Ver libro de vida” una vez los datos estén descargados, se procesara todo automáticamente y ya se podrán visualizar los datos en la interfaz. Puede tardar un rato, así que ten paciencia.

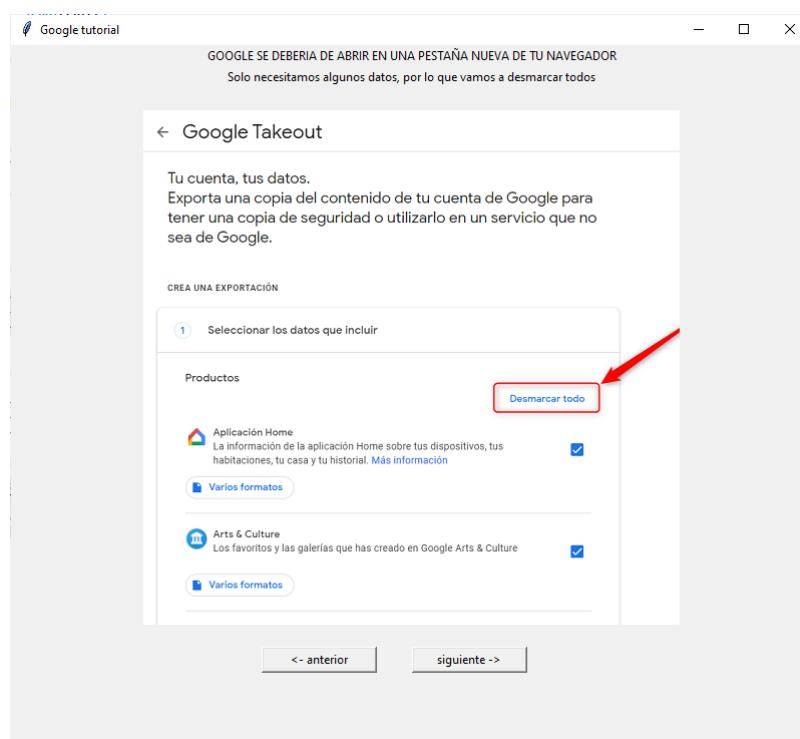


Figura B.6: Tutorial de Google

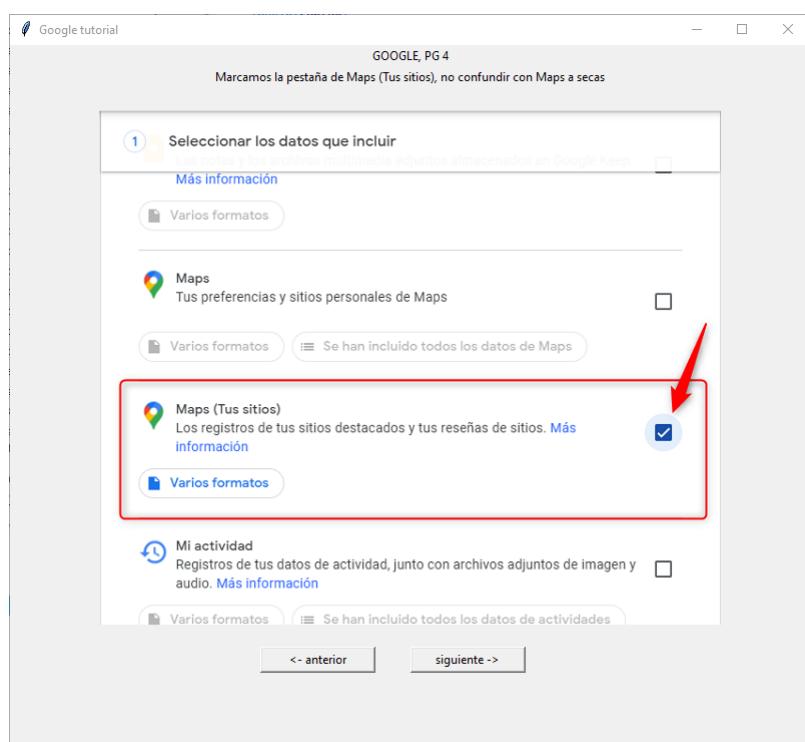


Figura B.7: Primero desmarcamos todo

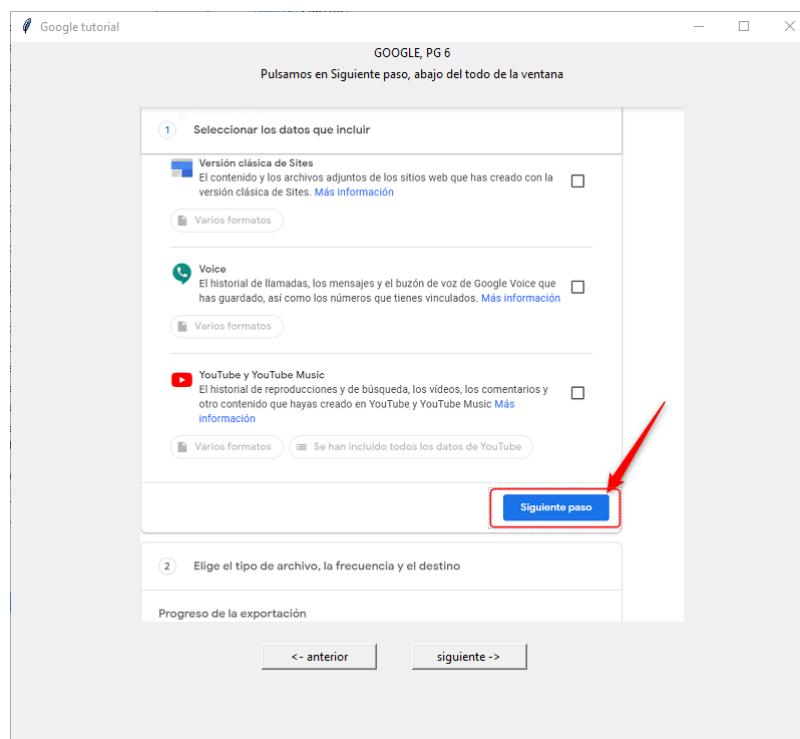


Figura B.8: Despu s marcamos los datos correctos

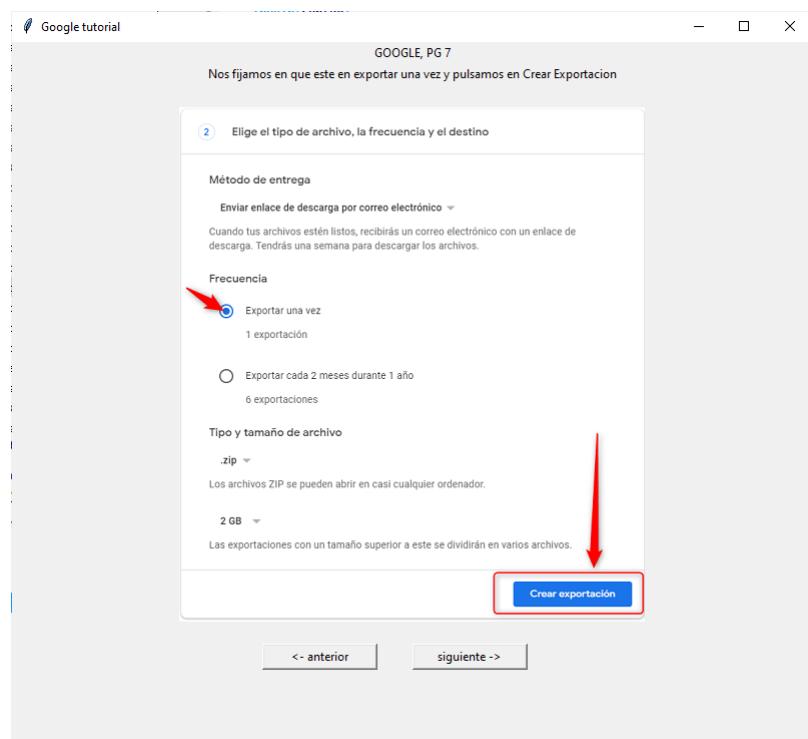


Figura B.9: Por último nos descargamos los datos



Figura B.10: Interfaz de usuario, apartado de la Página principal

B.2.4. Interfaz gráfica para visualizar datos

A partir de la **figura B.10** vamos a mostrar la parte de la aplicación en la que más tiempo vas a pasar, es decir, la interfaz donde mostramos todos los datos una vez descargados y tratados. En las siguientes figuras iremos explicando los apartados paso a paso.

La **figura B.10** muestra la parte de Home, donde presentamos el proyecto y donde se explica brevemente todo lo que se puede hacer en los diferentes apartados de la web.

En la **figura B.11** puedes ver el apartado de Amigos, en el que se muestran los diez amigos con los que más interactúas, en la figura se aprecia una tabla en la que en la primera columna se encuentra el nombre de tu amigo, seguido de la fecha en la que os hicisteis amigos en Facebook y de la puntuación obtenida basándonos en el número de veces que comentas o te comentan las publicaciones, así como las veces que reaccionáis a vuestros posts.

En la **figura B.12 y B.13** puedes ver el apartado de Lugares, que corresponde a los distintos sitios en los que el usuario ha estado, ha publicado en Facebook o Google, así como las reseñas que ha dejado en algunos establecimientos. Se puede apreciar un mapa con marcadores que corresponde a todos esos sitios, una lista de las personas con las que ha estado, así como una galería si es que se publicó alguna foto en alguno de esos sitios.

En la **figura B.14** puedes apreciar el apartado de fechas, que corresponde con una tabla de todos los datos de Facebook y Google ordenados en el

Amigo /	Fecha /	puntuacion
Noelia	2011-05-24 17:24	62
Pablo	2014-12-06 14:39	37
Helena	2014-12-06 00:43	17
Javier	2015-01-29 22:38	14
Sergi	2015-06-22 15:25	13
Bárbara	2015-06-04 16:28	8
Yoli	2015-07-16 20:57	7
Marta	2012-09-26 18:02	5
Mario	2013-03-21 22:36	5
Carmela	2015-09-20 06:08	5
Maria	2014-12-12 01:17	5

Figura B.11: Interfaz de usuario, apartado de amigos

ID /	Nombre del lugar /	Fecha /	Url /	Comentario	Nº de perso...	Nº de fotos	Puntuación
1	Str. General Tríana Mosoll, nr. 24, Brus, Rom...	2016-06-03 15:55	https://www.instagram.com...		0	0	5
2	16000 Cuenca	2015-09-21 16:34	---	SAN MATEOOO!	7	8	5
3	Barhop Madrid	2021-07-30 20:19	https://www.google.com/m...	Cervezaaaa :D	0	0	6
4	Plaza del Dos de Mayo	2021-07-08 19:47	https://www.google.com/m...	Muy buena plaza la verdad, pero a las 9 llega l...	0	0	6
5	Restaurante La Guitarr	2021-07-03 10:55	https://www.google.com/m...	Los camareros muy majos y la comida increib...	0	0	8
6	Macuto Latino	2021-06-18 23:22	https://www.google.com/m...	Increible bar, con una comida riquisima y cop...	0	0	8
7	Bar El Trocadero	2021-06-18 19:44	https://www.google.com/m...	Mu majos	0	0	9
8	Zombie Bar	2021-06-11 23:26	https://www.google.com/m...	Vaya hamburguesas...	0	0	6
9	Callao Smart Parking	2021-06-03 20:18	https://www.google.com/m...	Jajajaja es un ascensor de coches	0	0	6
10	Parque Carlos París	2021-06-03 20:18	https://www.google.com/m...		0	0	6

Figura B.12: Interfaz de usuario, apartado de lugares, mapa y tabla

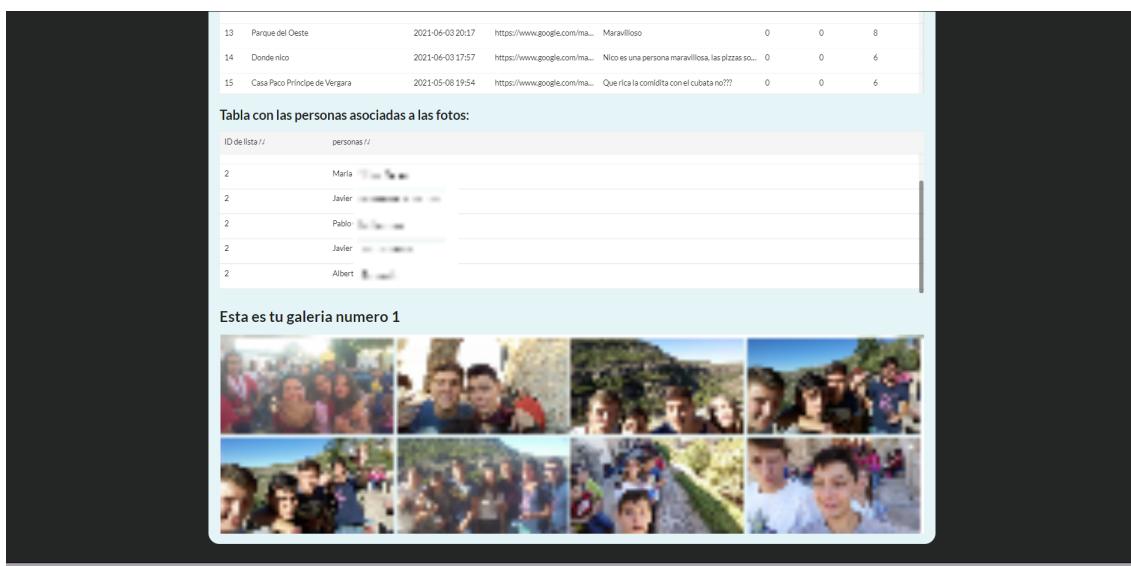


Figura B.13: Interfaz de usuario, apartado de lugares, personas y fotos

tiempo, así como el suceso que tuvo lugar en esta fecha. También hay un botón que te lleva a un timeline de estas fechas.

La **figura B.15** muestra los datos de fechas que ya se han mostrado en el apartado de fechas, pero de una forma más lineal y fácil de visualizar, puedes verlo de la forma que te resulte mas cómoda.

La **figura B.16** muestra las imágenes que nuestra aplicación ha considerado como más importantes para el usuario dentro de Facebook y Google, junto a la descripción que el usuario ha añadido en estas y si han sido fotos de perfil de este o no. En caso de no serlo, ha sido nuestra aplicación la que le ha dado la relevancia.

La **figura B.17** muestra las galerías de imágenes en las que el paciente ha sido etiquetado con sus mejores amigos, estos se corresponden con los mejores amigos calculados en el anterior apartado de amigos.

La **Figura B.18** Muestra las páginas con las que el paciente ha interactuado, ordenadas en base a la puntuación que hemos calculado, que indican la relevancia que estas han tenido para él.

Las últimas **figuras, B.19 y B.20** muestran los logros académicos y laborales del paciente y la información personal que el usuario tiene en Facebook, y por último también hay un apartado de familia donde se muestran los familiares que tiene registrados en la misma aplicación.

The screenshot shows a user interface for managing important dates. At the top, there is a navigation bar with tabs: Home, Amigos, Lugares, Fechas (which is highlighted in green), Fotos, Foto_tags, Páginas, Logros, Info, and Familia. Below the navigation bar, the title "Pagina con las fechas importantes" is displayed. A "Ver timeline" button is located above a table. The table has two columns: "fecha /:" and "contenido /:". The data in the table is as follows:

fecha /:	contenido /:
2021-08-07	Evento: BEL-AIR
2021-07-30	Guardaste en Google el sitio "Barhop Madrid" Con el comentario: Cervezaaaa:O
2021-07-08	Guardaste en Google el sitio "Plaza del Dos de Mayo" Con el comentario: Muy buena plaza la verdad, pero a las 9 llega la poli y u... Que pereza
2021-07-03	Guardaste en Google el sitio "Restaurante La Guitarra" Con el comentario: Los camareros muy majos y la comida increíble, un bar muy recomendable, de ...
2021-06-18	Guardaste en Google el sitio "Macuto Latino" Con el comentario: Increíble bar, con una comida riquísima y copazos baratos <3
2021-06-18	Guardaste en Google el sitio "Bar El Trocadero" Con el comentario: Mu majos
2021-06-11	Guardaste en Google el sitio "Zombie Bar" Con el comentario: Vaya hamburguesas...
2021-06-03	Guardaste en Google el sitio "Callao Smart Parking" Con el comentario: Jajajajajaja es un ascensor de coches
2021-06-03	Guardaste en Google el sitio "Parque Carlos París" Con el comentario: Vacío
2021-06-03	Guardaste en Google el sitio "FaroBrt Alcalá 61" Con el comentario: Vacío
2021-06-03	Guardaste en Google el sitio "RUTA 108" Con el comentario: Vacío
2021-06-03	Guardaste en Google el sitio "Parque del Oeste" Con el comentario: Maravilloso
2021-06-03	Guardaste en Google el sitio "Donde nico" Con el comentario: Nico es una persona maravillosa, las pizzas son increíbles y las cervezas para acompañarla ...
2021-05-08	Guardaste en Google el sitio "Casa Paco Príncipe de Vergara" Con el comentario: Que rica la comida con el cubata no???
2021-01-30	Evento: M26 - Equipo Caxi for Data
2020-09-18	Evento: Cumple Helena
2020-05-06	Evento: The Next Brain: Feeling Machines with Antonio Damasio

App dentro de proyecto CANTOR de la Universidad Complutense de Madrid

Figura B.14: Interfaz de usuario, apartado de fechas en forma de tabla

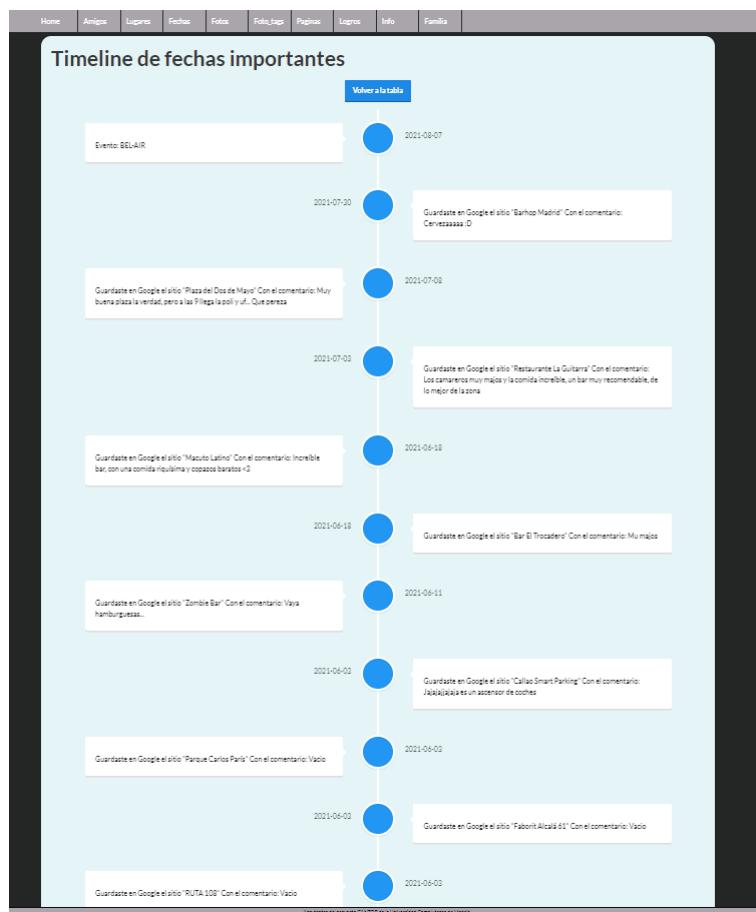


Figura B.15: Interfaz de usuario, apartado de fechas en forma de timeline

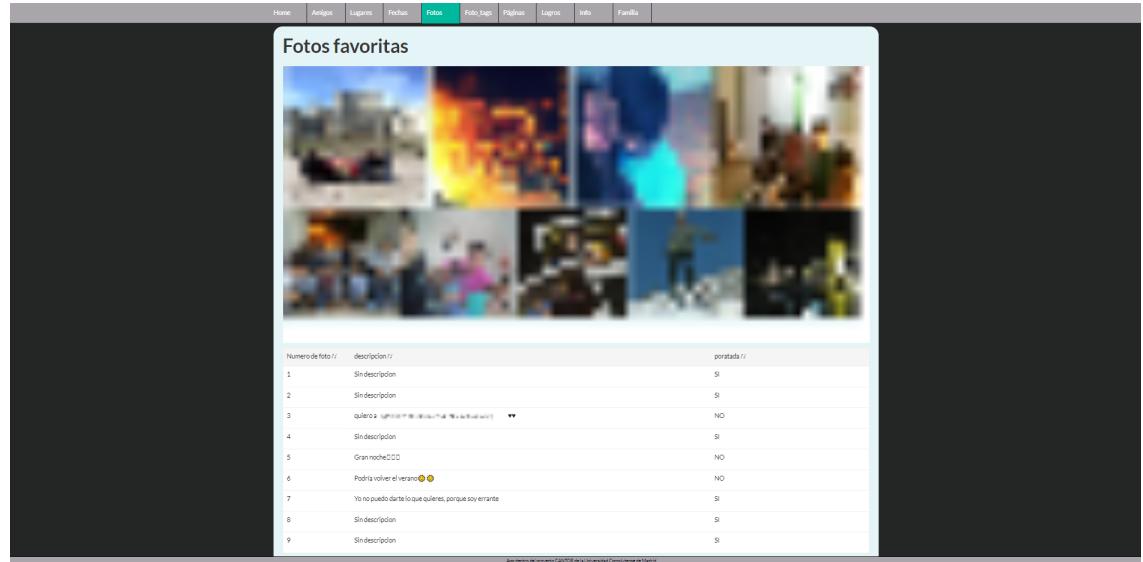


Figura B.16: Interfaz de usuario, apartado de fotos



Figura B.17: Interfaz de usuario, apartado de fotos etiquetadas con amigos

nombre de la página /	La sigues desde: /	Puntuación /
A Summer Story	2012-04-12 15:15	5
David Coto Medranda	2017-05-19 05:12	3
Bloqula	2015-10-04 19:24	1
David Palomo Téjero	2017-10-08 15:40	0
vibuk.com	2017-09-06 09:23	0
BeMyEye - ES	2017-08-21 18:47	0
Picos Almazan	2017-08-10 14:34	0
Energy Control	2017-08-03 10:04	0
Chollometro	2017-05-21 21:19	0
Centro Complutense de Interpretación de la Ciudad Universitaria de Madrid	2017-05-11 13:22	0
Trampalos	2017-03-18 20:50	0
Conciertos en Madrid	2016-10-01 13:55	0
Ilmfinity	2015-08-08 07:28	0
Monster Energy	2013-03-26 07:17	0
Haces Más con SIGAUS	2013-01-22 16:23	0
Los Mares oficial	2013-01-09 21:14	0
Extremaduro (oficial)	2013-01-09 21:13	0
Recogida de Alimentos	2012-11-27 17:50	0

App dentro del proyecto CANTOR de la Universidad Complutense de Madrid

Figura B.18: Interfaz de usuario, apartado de páginas favoritas

Nombre /	Fecha inicio /	Fecha fin	Graduado	Description
Universidad Complutense de Madrid	2014-01-01 09:00	Aun no se ha acabado	NO	Sin descripción
IES Gaya Nuño	2008-01-01 09:00	2014-01-01 09:00	SI	Sin descripción
Empresa de Ejemplo	2008-01-01 09:00	2014-01-01 09:00	Desconocido	hago cosas
Empresa de Ejemplo dos	2008-01-01 09:00	2014-01-01 09:00	Desconocido	hago mas cosas

App dentro del proyecto CANTOR de la Universidad Complutense de Madrid

Figura B.19: Interfaz de usuario, apartado de logros académicos y laborales

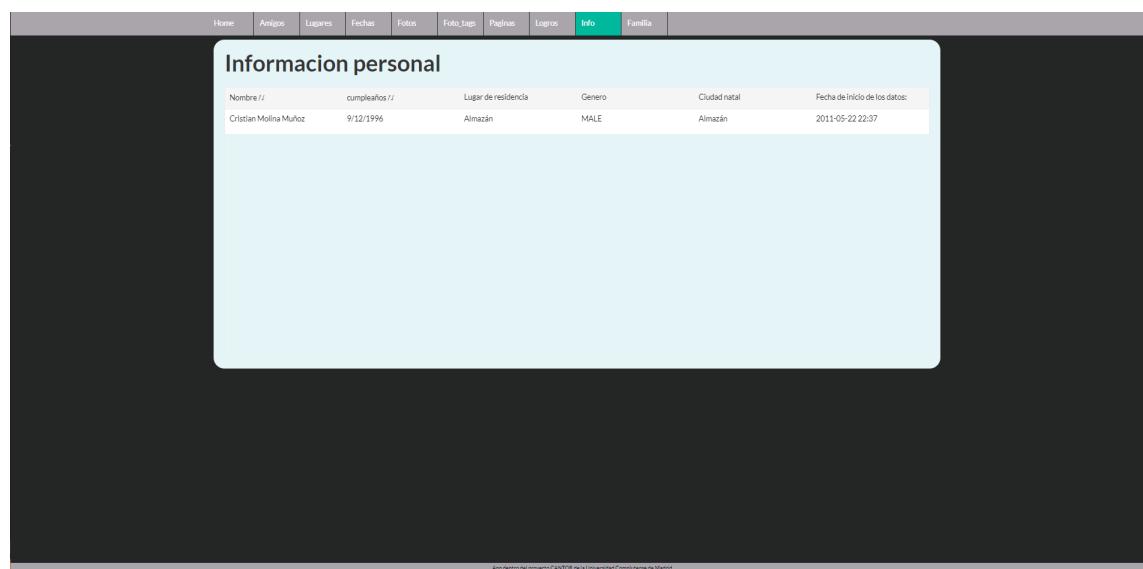


Figura B.20: Interfaz de usuario, apartado de información personal

Bibliografía

- ABU KHAIT, A., REAGAN, L. y SHELLMAN, J. Uses of reminiscence intervention to address the behavioral and psychosocial problems associated with dementia: An integrative review. *Geriatric Nursing*, vol. 42(3), páginas 756–766, 2021. ISSN 0197-4572.
- AIZEL BOTO, A., PORTILLO TORRES, R. y SANZ MAYO, D. Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de deeplearning, 2021. Trabajo de Fin de Grado en Ingeniería del Software, Facultad de Informática UCM, Departamento de Ingeniería del Software e Inteligencia Artificial, Curso 2020/2021, todos los recursos del proyecto, ejecutables y código se pueden encontrar en: <https://github.com/NILGroup/TFG2021RecuerdosMultimedia>.
- AIZPURUA, A. y KOUTSTAAL, W. A matter of focus: Detailed memory in the intentional autobiographical recall of older and younger adults. *Consciousness and Cognition*, vol. 33, páginas 145–155, 2015. ISSN 1053-8100.
- ARCHIBALDO, D. S. Deterioro y demencia orientacion para medicos no especialistas. *Cuadernos de neuropsicología*, vol. 1, páginas 115 – 126, 2007.
- BAND, A. Text2emotion. 2020a.
- BAND, A. Text2emotion: Python package to detect emotions from textual data. 2020b.
- BEDINGER OLIVERAS, D., CANO MUÑOZ, C., PRATS ULLOA, D. y SÁNCHEZ ÁLVAREZ, C. Sistema de asistencia para cuidados de enfermos del alzheimer, 2021. Trabajo de Fin de Grado en Ingeniería del Software y Grado en Ingeniería Informática, Facultad de Informática UCM, Departamento de Ingeniería del Software e Inteligencia Artificial, Curso 2020/2021.
- BHAGESHPUR, K. Data is the new oil - and that's a good thing. 2019.

- BRILLOUT, R. Absolutely awesome react components and libraries. 2019.
- CARRIERE-SWALLOW, Y. y HAKSAR, V. The economics and implications of data : An integrated perspective. 2019.
- CODERASHA. Advanced automation tips with python | selenium. 2019.
- DUMITRACHE, C., RUBIO, L. y CORDÓN-POZO, E. Successful aging in spanish older adults: the role of psychosocial resources. *International Psychogeriatrics*, vol. 31, páginas 1–11, 2018.
- ELPAÍS. Publicar un libro con las andanzas personales en facebook. 2010.
- EUGERCIOS SUÁREZ, G., GUTIÉRREZ MERINO, P. y KALOYANOVA POPOVA, E. Análisis emocional para la inclusión digital, 2018. Trabajo de Fin de Grado en Ingeniería Informática (Universidad Complutense, Facultad de Informática, curso 2017/2018).
- FAMILY, T. S. M. Iv estudio sobre los usuarios de facebook, twitter e instagram en españa. 2018.
- GUTIÉRREZ MERINO, P. Emotraductor 2.0. análisis emocional de textos en español, 2020. Trabajo de Fin de Máster en Ingeniería Informática, Facultad de Informática UCM, Departamento de Ingeniería de Software e Inteligencia Artificial, Curso 2019/2020.
- JSPINER. Python ics to json library. 2017.
- LIVINGSTON, G., HUNTLEY, J., SOMMERLAD, A., AMES, D., BALLARD, C., BANERJEE, S., BRAYNE, C., BURNS, A., COHEN-MANSFIELD, J., COOPER, C., COSTAFREDA, S. G., DIAS, A., FOX, N., GITLIN, L. N., HOWARD, R., KALES, H. C., KIVIMÄKI, M., LARSON, E. B., OGUNNIYI, A., ORGETA, V., RITCHIE, K., ROCKWOOD, K., SAMPSON, E. L., SAMUS, Q., SCHNEIDER, L. S., SELBÆK, G., TERI, L. y MUKADAM, N. Dementia prevention, intervention, and care: 2020 report of the lancet commission. *The Lancet*, vol. 396(10248), páginas 413–446, 2020. ISSN 0140-6736.
- ORACLECORPORATION. Mysql connector/python developer guide. 2021.
- PERIS, I. D., PONS, E. S., GONZALEZ, J. y MELENDEZ, J. C. Eficacia de la terapia de reminiscencia en adultos mayores con demencia tipo alzheimer. *1 EL PAPEL DE LA DEPRESIÓN EN LA PREDICCIÓN DE LA CALIDAD DE VIDA DE LAS PERSONAS MAYORES*, página 94, ????.
- POWER, J. M., TANG, J., LAWLOR, B., KENNY, R. A. y KEE, F. Mediators of the relationship between social activities and cognitive function among older irish adults: results from the irish longitudinal study on ageing. *Aging & Mental Health*, vol. 22(1), páginas 129–134, 2018. PMID: 27676290.

- PYTHONSOFTWAREFOUNDATION. Csv file reading and writing. 2018.
- PYTHONSOFTWAREFOUNDATION. Operaciones comunes de cadena de caracteres. 2019.
- PYTHONSOFTWAREFOUNDATION. google-trans-new 1.1.9. 2020a.
- PYTHONSOFTWAREFOUNDATION. Regular expression operations. 2020b.
- PYTHONSOFTWAREFOUNDATION. webdriver — cómodo controlador de navegador web. 2020c.
- PYTHONSOFTWAREFOUNDATION. Codificador y decodificador json. 2021a.
- PYTHONSOFTWAREFOUNDATION. Db-api 2.0 interface for sqlite databases. 2021b.
- PYTHONSOFTWAREFOUNDATION. Inspect live objects. 2021c.
- PYTHONSOFTWAREFOUNDATION. Interface de python para tcl/tk. 2021d.
- PYTHONSOFTWAREFOUNDATION. Interpreting stat() results. 2021e.
- PYTHONSOFTWAREFOUNDATION. Miscellaneous operating system interfaces. 2021f.
- PYTHONSOFTWAREFOUNDATION. Operaciones de archivos de alto nivel. 2021g.
- PYTHONSOFTWAREFOUNDATION. Parámetros y funciones específicos del sistema. 2021h.
- PYTHONSOFTWAREFOUNDATION. Time access and conversions. 2021i.
- SÁEZ, E. P. Terapia de reminiscencia. 2020.
- SOFTWAREFREEDOMCONSERVANCY. Selenium webdriver oficial website. 2021.
- STEVEN BIRD, E. K., EDWARD LOPER. Natural language toolkit. 2021.
- YANG, T. Tratar datos json. 2019.
- YARRITU, J. P. P. Desarrollo de un sistema de recolección de datos sobre medios digitales para su análisis cibermétrico en twitter. 2017.

