Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

Mención en Tecnologías de la Información

Transcripción del audio de vídeos y podcasts

Autor: Samuel Bernaldo de Quirós Lalinde
Tutor: Blas Torregrosa García

A todos los que me han apoyado en el camino.

Agradecimientos

Quiero agradecer a todas esas personas que han puesto de su parte para hacer de este proceso algo más sencillo.

A mi familia, por brindarme todo el apoyo incondicional y en todas sus formas en los momentos más complicados, y por siempre ayudarme a no ceder.

A mis amigos, por ser el mejor acompañamiento todos estos años. Por saber cuándo y cómo hacer que me olvidara de todos los problemas.

A todos los compañeros de la carrera a los que he tenido la oportunidad de conocer, por todas las horas y momentos compartidos tratando de conseguir el objetivo.

Gracias a todos. No habría sido posible sin vosotros

Resumen

El siguiente proyecto ha sido realizado en la Escuela de Ingeniería Informática de la Universidad de Valladolid, constituyendo el Trabajo Final de Grado en la mención de Tecnologías de la Información.

El objetivo de este proyecto es desarrollar una aplicación para la transcripción de ficheros de audio de diversos formatos a ficheros de texto. Se pretende ofrecer a los usuarios una aplicación de fácil manejo que permita subir uno o varios ficheros de audio y obtener un fichero en formato PDF que contenga el texto del audio escrito.

Las aplicaciones de traducción de audio son herramientas muy útiles en el apoyo de personas con problemas auditivos, o personas desplazadas con dificultades con el idioma. Aplicaciones de este carácter y similares ofrecen a estas personas un amplio abanico de posibilidades de acceso a información y comunicación del cual, de otro modo, estarían privados.

Esta aplicación ha sido diseñada con AngularTS, HTML, CSS y Spring Boot, y utilizando la metodología ágil Scrum.

Palabras clave: Conversión, transcripción, podcast, audio, texto, aplicación, Java, TypeScript.

Abstract

The following project has been carried out at the School of Computer Engineering of the University of Valladolid, constituting the Final Degree Project in the mention of Information Technologies.

The proyect's objective is to develop an application for the transcription of audio files of various formats to text files. It is intended to provide users with an easy-to-use application that allows uploading one or several audio files and obtaining a PDF file containing the text of the audio.

Audio translation applications are very useful tools in supporting people with hearing impairments, or moved out persons with language difficulties. Applications of this nature, offer these people a lot of possibilities for access to information and communication that they usually has been deprived of.

This applicatios has been developed with AngularTS, HTML, CSS and Spring Boot, and using the agile Scrum methodology.

Keywords: Conversion, transcription, podcast, audio, text, application, Java, TypeScript.

Índice

Índice de figuras

Índice de cuadros

1 Introducción

1.1. Introducción

El Reconocimiento Automático del Habla (RAH) [1][2], también conocido por sus siglas en inglés ASR (Automatic Speech Recognition) [3], es una rama de la inteligencia artificial enfocada en el desarrollo de sistemas con la capacidad de interpretar y comprender el lenguaje humano hablado.

Es una tarea fundamental en el campo del Procesamiento del Lenguaje Natural (PLN) [4][5], campo de la Inteligencia Artificial encargado de la investigación de formas de comunicación de las personas con las máquinas a través del uso de lenguajes.

A lo largo de este trabajo se explorarán y desarrollarán soluciones para la transcripción de audio a texto a través del uso de métodos avanzados del aprendizaje automático y procesamiento de señales de audio.

1.2. Objetivos

1.2.1. Objetivos de desarrollo

El objetivo del proyecto se basa en la creación de una aplicación automática que permita la transcripción de audio de distintos entornos, como pistas de audio o de vídeo, en texto. Con la creación de esta aplicación se pretende facilitar la disponibilidad de la información a través del acceso a contenido multimedia en forma de texto de manera rápida y eficiente. Además, se pretende contribuir al avance de la accesibilidad digital al proporcionar una herramienta que facilite el acceso a la información para personas con discapacidad auditiva.

1.2.2. Objetivos personales

En cuanto a los objetivos personales, al tratarse de un proyecto realizado en un entorno educativo, éstos se centran en la obtención de nuevos conocimientos.

- Mejorar mis habilidades de programación y desarrollo de software, especialmente en el campo del procesamiento de audio y texto.
- Aprender sobre las tecnologías y algoritmos utilizados en el reconocimiento automático del habla y el procesamiento del lenguaje natural.
- Obtener experiencia práctica en el diseño, desarrollo y despliegue de aplicaciones web basadas en tecnologías de inteligencia artificial y aprendizaje automático.
- Adquirir nuevos conocimientos sobre el desarrollo de proyectos y la aplicación de metodologías de trabajo.

1.3. Estructura de la memoria

El contenido de éste documento de memoria se ha estructurado como se detalla a continuación:

- A lo largo del **capítulo 2**, se detallará a grandes rasgos el estado actual de la tecnología de conversión de audio a texto; junto un pequeño repaso de las etapas y la evolución que ha tenido a lo largo del tiempo, y se detallarán brevemente los retos futuros a los que se enfrenta.

- En el **capítulo 3** se explicará la planificación del proyecto, las metodologías utilizadas, su explicación e integración en el proyecto, y se detallará el trabajo realizado.
- En el **capítulo 4** se abordará la enumeración y explicación de todas las tecnologías implicadas en el desarrollo de la aplicación, así como cuál ha sido su uso.
- El **capítulo 5** comprende los aspectos generales del funcionamiento de la transcripción y las limitaciones de éste.
- Durante el **capítulo 6**, se verán en profundidad el análisis técnico de la aplicación desarrollada.
- Dentro del **capítulo 7**, se explicarán los aspectos del diseño de la aplicación.
- A lo largo del **capítulo 8**, se explicará y profundizará en la implementación final de la aplicación.
- Por último, en el **capítulo 9**, detallaremos las pruebas realizadas sobre el producto final.

1.4. Cómo obtener el audio de vídeos y podcasts

Pese a ser el proyecto una aplicación de transcripción del audio de vídeos y podcasts, ésta ha sido diseñada únicamente partiendo de la base de que el usuario cuenta con el fichero de audio para poder subirlo a la aplicación; por tanto, en este apartado, se van a enumerar diferentes maneras para obtener dicho archivo y poder realizar la correcta transcripción. Por razones evidentes, se van a omitir los métodos que no cumplen con la legislación sobre la propiedad intelectual y los derechos de autor.

1.4.1. Grabación del audio

El método más común y simple para obtener la pista de audio, consiste en directamente realizar una grabación sobre la misma con un dispositivo externo como puede ser un teléfono móvil. Este tipo de grabaciones deterioran parcialmente la calidad del audio pero no afecta al funcionamiento de la aplicación.

También se pueden realizar grabaciones del audio con software especializado, como OBS o Audacity. Si bien este método permite una mejor calidad del audio, también se trata de un método más laborioso.

1.4.2. Descarga desde la aplicación

Algunas aplicaciones de contenidos permiten la descarga de los ficheros de audio desde sus propias plataformas, como Apple Podcasts, Google Podcasts o iVoox. Este método es el principal siempre que sea posible, dado que cumple con los requisitos legales de derechos de autor, genera ficheros de audio de la mejor calidad y no requiere de ningún tipo de pago.

1.4.3. Pago de las versiones Premium

Una gran parte de las aplicaciones de contenidos de audio y podcasts, como YouTube, Spotify o Audible, permiten al usuario el pago de una cantidad para obtener la versión premium de la aplicación, y, con ella, poder descargar los ficheros de audio.

Esta opción es bastante conveniente, dado que cumple con la legalidad vigente y permite obtener las pistas de audio con la mejor calidad posible, si bien requiere de un gasto por parte del usuario.

2 Estado del Arte

2.1. Historia del reconocimiento del habla

2.1.1. Primeras etapas del reconocimiento del habla

La tecnología del reconocimiento del habla [6][7][8][9][10] no se trata de algo novedoso , ya en la década de 1950 la compañía Bell Labs creó un sistema, denominado "Audrey", capaz del reconocimiento de dígitos aislados.

Durante las décadas posteriores, se fue produciendo un avance paulatino en dichas tecnologías; se comenzaron a tener en cuenta factores del habla humana como los signos de puntuación, reconocimiento de las cadenas de voz independientemente del hablante, etc.

Gracias a estos progresos, se fueron desarrollando sistemas capaces de un reconocimiento más complejo y sofisticado, como la creación, por parte de la empresa IBM, del dispositivo "Shoebox", capaz de comprender 16 palabras y algunas operaciones matemáticas básicas, o el sistema "HARPY", diseñado por la Universidad Carnegie Mellon, capaz de identificar hasta mil palabras.

2.1.2. Modelos ocultos de Markov

Los Modelos Ocultos de Markov (HMM) [11][12][13] son modelos estadísticos orientados a determinar parámetros desconocidos a partir de parámetros observables. Estos modelos asocian a cada estado no observable una distribución de probabilidad, proporcionando información sobre la secuencia de estados.

Durante la segunda mitad de la década de 1980, se comenzaron a utilizar los modelos ocultos de Markov en el reconocimiento del habla. A través de la síntesis del audio, se consigue asignar a cada fonema una probabilidad de patrones definidos, lo que permitía al sistema reconocer secuencias y generar mejores transcripciones.

Este nuevo enfoque, capaz de proporcionar una estructura probabilística al habla humana, permitió un gran crecimiento en la calidad de los sistemas de reconocimiento del habla. Gracias a estos avances, la compañía IBM desarrolló un sistema de reconocimiento de palabras denominado "Tangora". Dicho sistema, si bien requería de un entrenamiento individual, era capaz de reconocer y escribir 20000 palabras.

2.1.3. Redes neuronales artificiales

Las Redes Neuronales Artificiales (RNA)[14][15] son modelos computacionales inspirados en el funcionamiento del cerebro humano, diseñados para procesar información de manera similar a las neuronas biológicas.

Su funcionamiento se basa en la combinación de conexiones entre elementos. Gracias a una combinación de datos de entrada y el cálculo de salidas a través de conexiones, pueden aprender y reconocer patrones complejos en los datos.

El uso de redes neuronales artificiales ha revolucionado el campo del reconocimiento del habla, superando las limitaciones de los enfoques tradicionales. Gracias a su capacidad de aprendizaje, se han incrementado considerablemente los niveles de precisión y robustez, permitiendo obtener resultados precisos en entornos ruidosos o con hablantes no nativos. Además, ha permitido la creación

de avances como la traducción automática de voz o la generación de voz sintética de alta calidad.

2.1.4. Modelos de Lenguaje Grande

El mayor uso de las redes neuronales en varios ámbitos desembocó en el uso de Redes Neuronales Recurrentes (RNN) [16][17], redes neuronales especializadas en el procesamiento de datos secuenciales como el lenguaje. Y, unos años más tarde, en el uso de Redes Transformer [18][19][20]. Este tipo de redes se diferencian en que, a la hora de procesar la secuencia de entrada, se realiza de forma paralela y no serial como se hacía en las RNA, permitiendo así a la red aprender el contexto de relaciones en datos secuenciales.

Los Modelos de Lenguaje Grande [21][22][23], conocidos por sus siglas en inglés LLM (Large Language Model), son modelos de lenguaje compuestos por una red neuronal con millones de parámetros, entrenados con grandes cantidades de texto y aprendizaje autosupervisado. Estos lenguajes están, en su gran mayoría, implementados con el uso de redes transformer.

Los sistemas de reconocimiento del habla se han beneficiado de estos avances en los LLM, gracias a su capacidad para comprender las complejidades intrínsecas a la comunicación hablada. La capacidad de estos modelos de analizar el contexto de las palabras en una secuencia de audio de forma más eficiente y el entrenamiento con grandes cantidades de datos permite a los nuevos sistemas de reconocimiento adaptarse a distintos acentos, dialectos o velocidades del habla, aumentando su versatilidad.

2.2. Estado actual

Estos avances en la transcripción del habla han desembocado en la creación de infinidad de sistemas capaces de realizar un reconocimiento del habla de manera eficiente y precisa. Además, se trata de una tecnología cada día más extendida; hoy en día, es común el uso de estas tecnologías en sistemas de dictado automático integrados en cualquier smartphone, asistentes virtuales, sistemas de navegación o centros de llamadas.

En este trabajo vamos a centrarnos en los sistemas capaces de realizar el reconocimiento del habla junto con su transcripción a texto, de manera similar al dictado automático, o a la generación automática de subtítulos. Estos sistemas deben tener la capacidad de comprender y procesar el habla, así como de la generación de subtítulos o documentos de texto.

Las herramientas y plataformas de desarrollo juegan un papel crucial en el diseño, implementación y despliegue de sistemas de reconocimiento automático del habla. A continuación se describen algunas de las herramientas y plataformas más utilizadas en este campo.

2.2.1. Bibliotecas de Código Abierto

Las bibliotecas de código abierto [24][25] son ampliamente utilizadas en la investigación y desarrollo de sistemas de reconocimiento automático del habla y su conversión a texto. Proporcionan una amplia gama de funcionalidades para el procesamiento de voz y texto, incluyendo modelos preentrenados, algoritmos de aprendizaje automático y herramientas de evaluación de rendimiento. Además, la comunidad de código abierto ofrece soporte y colaboración para el desarrollo continuo de estas bibliotecas. Algunas de las más utilizadas son Whisper [26], TensorFlow [27], PyTorch [28] o DeepSpeech [29].

2.2.2. Plataformas de Desarrollo de Nube

Las plataformas de desarrollo de nube ofrecen servicios de reconocimiento automático del habla basados en la nube que permiten a los desarrolladores integrar fácilmente capacidades de transcripción de voz en sus aplicaciones y servicios. Estas plataformas proporcionan APIs y SDKs que permiten el acceso a potentes modelos de reconocimiento de voz y ofrecen funcionalidades adicionales como la detección de idiomas, la transcripción en tiempo real y la traducción automática. Dentro de este grupo podemos observar algunas como Google Cloud Speech-to-Text [30], Amazon Transcribe [31], Microsoft Azure Speech-to-Text [32] o Assembly AI [33].

2.2.3. Software Especializado

El software especializado, como Dragon NaturallySpeaking [34], está diseñado para usuarios finales y profesionales que requieren funcionalidades avanzadas de reconocimiento automático del habla. Estas aplicaciones ofrecen características como comandos de voz personalizados, integración con aplicaciones de productividad y soporte para vocabulario técnico y médico especializado. El software especializado es ampliamente utilizado en entornos comerciales, médicos y educativos donde la precisión y la eficiencia en la transcripción de voz son críticas.

2.3. Áreas de investigación futuras

Como hemos podido comprobar, si bien el reconocimiento del habla se encuentra en un estado muy avanzado, aún se enfrenta a desafíos significativos. A pesar de que se haya alcanzado un alto grado de precisión, es un área claro de mejora, sobre todo cuando se añaden condicionantes externos como entornos ruidosos o locutores con problemas del habla como tartamudez. Además, estas tecnologías aún no son especialmente eficaces a la hora de distinguir aplausos o risas.

Así mismo, aún se presentan retos a resolver en cuanto a la adaptación plurilingüe de las aplicaciones. Dichas tecnologías aún tienen un amplio margen de crecimiento en lo referente a la adaptación a distintos idiomas, especialmente minoritarios, o distintos acentos y dialectos comprendidos dentro de un mismo idiomas. Esta adaptación es crucial para poder crear tecnologías más globalizadas.

Otro punto a tener en cuenta con el creciente desarrollo de estas tecnologías, es el tratamiento de la integridad y confidencialidad de los datos. A medida que se van integrando en la vida cotidiana, se deben tomar decisiones cruciales en torno a la posibilidad de manipulación de pistas de audio, almacenamiento y seguridad de las mismas, su uso para traducciones fraudulentas, etc. Si bien este es un punto en común con todas las tecnologías en expansión, es de especial sensibilidad en todas las relacionadas con el habla humana.

Además de los retos referentes a la parte del reconocimiento de voz; el área más clara de mejora viene dado en la generación de los textos. Es más habitual en estas tecnologías encontrar más fallos a la hora de transcribir palabras homófonas, nombres propios poco habituales, casos donde se trabaja con un vocabulario específico, o a la hora de diferenciar el uso de fonemas que se usan para nombrar letras y a la vez constituyen palabras, como pueden ser, en castellano, la pronunciación de las letras B y la palabra "ve"de los verbos ver o ir; en muchos de estos casos, la transcripción en texto de la oración coloca la letra B cuando el uso aplicado del fonema es el contrario. También es habitual encontrarse errores gramaticales leves, como dificultades a la hora de comprender dónde acaba una oración y empieza la siguiente. Otro error comúnmente dado en algunas de estas tecnologías, se produce cuando se trata de una conversación entre dos o más hablantes, observándose dificultades a la hora de transcribir de forma eficiente las frases de cada uno de los hablantes sin solaparlas entre ellas.

3 Planificación

3.1. Metodologías Ágiles

Las metodologías ágiles [35][36] son un conjunto de prácticas para la gestión de proyectos enfocadas en los requisitos de flexibilidad y entrega incremental.

La primera aparición de metodologías de gestión evolutiva de proyectos, se dio durante la década de 1990, con la creación de métodos "ligeros"de desarrollo software, como el Desarrollo Rápido de Aplicaciones (RAD), Scrum, la Programación Extrema (XP) o el Desarrollo Basado en Funcionalidades (FDD).

Una década más tarde de la aparición de dichas tecnologías, en el año 2001, diecisiete desarrolladores en conjunto realizaron la publicación del Manifiesto por el Desarrollo Ágil de Software [37], que sentó las bases de esta metodología. En este manifiesto expone los principios sobre los que se debe fundamentar un desarrollo ágil:

- Valorar la comunicación entre personas sobre los procesos y herramientas.
- Priorizar la entrega de software funcional a la generación de documentación extensa.
- Dar importancia a forjar una relación de colaboración con el cliente.
- Tener la capacidad de adaptación ante los cambios.

Estas formas de trabajo permiten realizar una planificación y desarrollos del producto de una forma más cercana al cliente, involucrándolo en las etapas del desarrollo y con una entrega continua de valor, proporcionando entregables a cada iteración. Esto refuerza las relaciones personales y permite aumentar el grado de adaptabilidad y de adaptación a las necesidades del cliente, generando así mayor satisfacción durante todo el proceso.

3.2. Metodología Scrum

Scrum [38] es un marco de trabajo que promueve la colaboración entre equipos para el desarrollo de productos complejos. Es un proceso que emplea metodologías ágiles a través de un enfoque incremental en periodos temporales fijos y cortos, con el objetivo de reducir la complejidad de los proyectos. Dada su naturaleza iterativa y de retroalimentación, es adecuado para proyectos propensos a sufrir cambios durante las etapas de desarrollo. Un proceso Scrum está basado en tres principios fundamentales:

- **Transparencia.** Todos los miembros del equipo y personas implicadas deben ser conocedores de los aspectos más importantes del proyecto.
- **Inspección.** Se debe evaluar la información de avance del producto cada cierto tiempo; así como su adopción en el mercado y su calidad.
- **Adaptación.** Como consecuencia de la inspección, es necesario modificar el proceso de desarrollo a las necesidades o variaciones que sean necesarias.

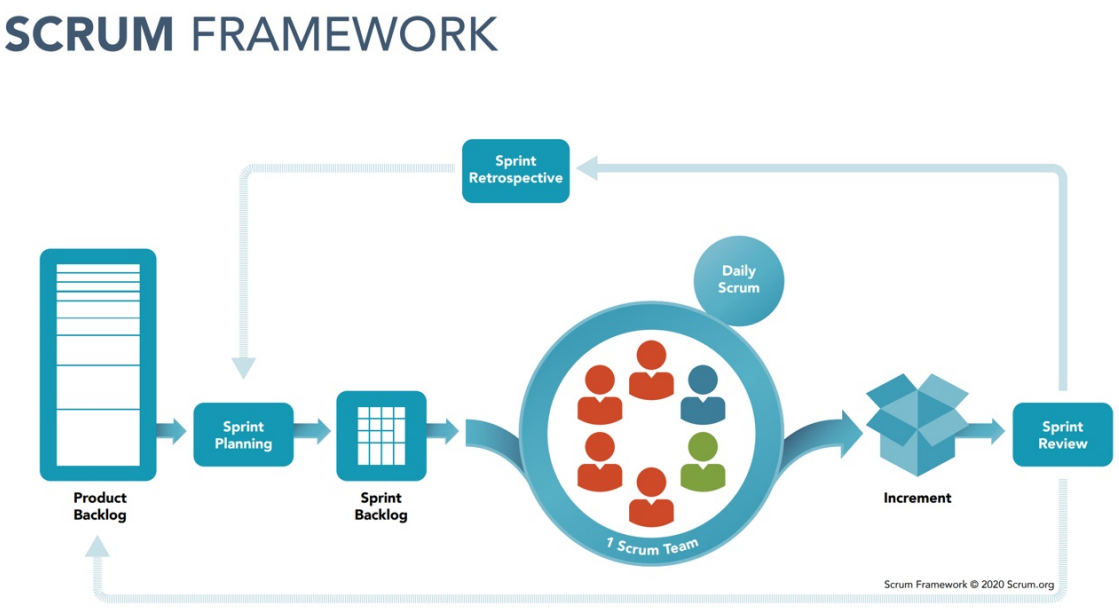

Figura 1: Metodología Scrum. Fuente: [39]

3.2.1. Equipo Scrum

Dentro de la metodología Scrum, un equipo de trabajo se divide en distintos roles [40], cada uno de ellos con distintas responsabilidades. Estos roles son:

- **Product Owner.** Es el encargado de maximizar el valor del producto y de gestionar el Product Backlog, una lista priorizada de funcionalidades o de tareas pendientes. Adicionalmente, también es el encargado de la comunicación con las partes interesadas del proyecto.
- **Scrum Master.** Es la persona encargada de la gestión del proyecto Scrum. Su labor dentro del equipos se basa en gestionar y facilitar que el proceso Scrum se realice correctamente y eliminar los posibles impedimentos que afecten en la capacidad del equipo de realizar entregas de valor. Debe asegurarse y facilitar que el equipo cumple los principios de la metodología.
- **Equipo de Desarrollo.** Es el grupo de personas más amplio y es el encargado del desarrollo del producto de forma auto-gestionada para conseguir entregas incrementales del producto en cada ciclo de desarrollo.

3.2.2. Eventos Scrum

Dentro de la metodología Scrum, se definen varios eventos [41] necesarios para facilitar el control de procesos y mantener los pilares de Scrum, así como para fomentar la comunicación durante el ciclo de desarrollo.

- **Sprint.** Es un periodo de tiempo, generalmente fijo, durante el cual el equipo trabaja para entregar un incremento. Dentro de cada sprint, se deben abordar todos los eventos scrum; y es la figura del Scrum Master la encargada de supervisar dicho cumplimiento. Inmediatamente tras la finalización de un sprint, da comienzo el siguiente.
- **Sprint Planning.** Es una reunión de todo el equipo que se debe realizar antes del comienzo del sprint. Durante la misma, se seleccionan los elementos del Product Backlog que van a ser abordados durante el sprint que se va a comenzar y se define el alcance, objetivos y trabajo a realizar del mismo.

- **Daily Scrum.** Es una reunión diaria y breve, realizada por el equipo de desarrollo, donde se abordan las tareas realizadas, las tareas a realizar y los posibles bloqueos o impedimentos para su realización. El objetivo es alinear el trabajo diario del equipo y controlar el progreso de la realización de las tareas.
- **Sprint Review.** Se realiza al final del sprint, y consiste en una reunión del equipo para comprobar si se han cumplido las tareas y requisitos que se han abordado durante el sprint. En caso de no haberse finalizado, da como resultado una adaptación del Product Backlog para el siguiente sprint.
- **Sprint Retrospective.** Consistente en una reunión realizada al finalizar un sprint dedicada a realizar un análisis sobre el mismo. A través de ésta, se busca obtener un conjunto de aspectos positivos y negativos que se hayan producido, de cara a optimizar y mejorar el funcionamiento para los siguientes sprints.

3.2.3. Artefactos Scrum

A través de la realización de los eventos scrum, se obtienen los artefactos [42], que son los elementos físicos que registran la información del proceso. Se pueden destacar los siguientes artefactos:

- **Product Backlog.** Consiste en una lista de requisitos del producto o historias de usuario, ordenadas por prioridad. Se trata de un artefacto que puede sufrir modificaciones a lo largo del proceso de desarrollo y debe ser revisada continuamente por el Product Owner. Constituye la principal fuente de información sobre el trabajo a realizar, mejoras, correcciones y funcionalidades a añadir en el producto.
- **Sprint Backlog.** Derivado del Product Backlog, se trata de una lista de historias de usuario, tareas o correcciones que serán abordadas en el sprint actual. Es el resultado de la Sprint Planning y constituye la guía del trabajo que se debe realizar en el ciclo actual, por ello, se trata de una lista de tareas estática que no se debe modificar durante la realización del sprint. Es gestionado por el equipo de desarrollo y ofrece una visión de la evolución del trabajo.
- **Incremento.** Es el resultado de cada sprint obtenido de la suma de las tareas, historias de usuario, correcciones y casos de uso que se hayan desarrollado. Debe consistir en un producto software utilizable y que aporte valor respecto a las entregas anteriores.

3.3. Adaptación del proyecto a Scrum

El uso de metodología Scrum en este proyecto, al tratarse de un trabajo de fin de grado realizado de forma individual, hace que no puedan existir todos los roles que se han detallado anteriormente. A continuación se va a tratar de describir cómo se ha reestructurado la organización del proyecto para poder adaptarse a la metodología.

El alumno asumirá los roles de Product Owner y de Equipo de Desarrollo, siendo el encargado de la creación, gestión, realización y comprobación de todas las tareas a realizar. Por su parte, el tutor, tendrá el rol de Scrum Master, gestionando el proceso de desarrollo y encargándose de solucionar los posibles impedimentos que aparezcan durante su realización.

3.4. Planificación

El proyecto ha sido realizado mediante metodología Scrum con sprints que se han fijado en una duración de dos semanas, siendo el viernes el periodo de finalización de los mismos. En este día se realizarán el Sprint Planning, Sprint Backlog y Sprint Retrospective correspondientes a las siguientes dos semanas de sprint.

Utilizando como referencia la guía de Trabajos de Fin de Grado de la Universidad de Valladolid [43], podemos comprobar que el tiempo estimado para la realización de estos proyectos es de 300 horas. Por motivos laborales, la disponibilidad del alumno, asumiendo el rol de Equipo de Desarrollo, es de en torno a 30 horas por sprint, por lo que la estimación será de 10 sprints. Además, se añadirá un sprint de refuerzo en previsión de posibles errores o dificultades que puedan suceder.

El proyecto comenzará en febrero de 2024 con el Sprint 0. Este sprint se utilizará como etapa de planificación, investigación, análisis y elección de tecnologías a utilizar en el proyecto. Esta parte es fundamental para sentar las bases de cómo será la realización del proyecto y evitar la improvisación durante su desarrollo.

En la tabla 3.1 podemos observar la planificación inicial de los sprints:

Sprint	Inicio	Fin	Comentarios
Sprint 0	16/02/2024	01/03/2024	
Sprint 1	01/03/2024	15/03/2024	
Sprint 2	15/03/2024	29/03/2024	
Sprint 3	29/03/2024	12/04/2024	
Sprint 4	12/04/2024	26/04/2024	
Sprint 5	26/04/2024	10/05/2024	
Sprint 6	10/05/2024	24/05/2024	
Sprint 7	24/05/2024	07/06/2024	
Sprint 8	07/06/2024	21/06/2024	
Sprint 9	21/06/2024	05/07/2024	Sprint opcional de refuerzo

Cuadro 3.1: Planificación inicial de sprints

Debido a problemas con los plazos, no se pudo realizar la entrega del proyecto en las fechas previstas; por tanto, se acordó con el tutor modificar el apartado de pruebas de la aplicación 9 para automatizar el proceso.

Como consecuencia, tras un descanso durante el verano, se continuó el trabajo con el resultado de la modificación de los sprints realizados.

Sprint	Inicio	Fin	Comentarios
Sprint 10	04/10/2024	18/10/2024	
Sprint 11	18/10/2024	01/11/2024	
Sprint 12	01/11/2024	15/11/2024	
Sprint 13	15/11/2024	29/11/2024	
Sprint 14	29/11/2024	13/12/2024	

Cuadro 3.2: Modificación adicional de sprints

3.5. Sprint Backlog

Se ha definido una serie de códigos identificativos para diferenciar los tipos de tareas que se han realizado. Estos códigos son:

- **DOC.** Código identificativo de las tareas relacionadas con el estudio y documentación del proyecto.
- **DES.** Código identificativo de las tareas de desarrollo de la aplicación.
- **ERR.** Código identificativo de las tareas asociadas a la investigación y corrección de errores.
- **MEJ.** Código identificativo para las tareas realizadas para las mejoras o evoluciones de las funcionalidades.

- **PRU.** Código identificativo de las tareas de pruebas de la aplicación y sus funcionalidades.

3.5.1. Sprint 0

Este sprint abarca las fechas comprendidas entre el 16/02/2024 y el 01/03/2024, y se va a emplear, principalmente, en el desarrollo de la parte inicial de la documentación y en la investigación sobre los aspectos generales del proyecto y su planificación.

Además, se utilizará parte del sprint para realizar un pequeño boceto de lo que podría ser la aplicación, de cara a comenzar el desarrollo enfocándose en una idea concreta.

Estas tareas han requerido una inversión de 29 horas.

A continuación, y como se repetirá para el resto de sprints, se adjunta una tabla (3.3) con las tareas que se han planificado, su tiempo empleado, y el estado de la misma al finalizar el sprint. Las tareas planificadas que no puedan ser finalizadas en su sprint correspondiente, se volverán a añadir en el siguiente sprint para su finalización.

Tarea	Descripción	Estado	Tiempo
DOC - 001	Investigación del proyecto.	Completa	9h
DOC - 002	Investigación de las tecnologías a utilizar.	Completa	4h
DOC - 003	Definición de historias de usuario.	Incompleta	5h
DOC - 004	Redactar la parte de introducción del proyecto.	Completa	2h
DOC - 005	Diseño inicial de la aplicación.	Completa	4h
DOC - 006	Redactar el estado del arte.	Incompleta	5h
Total			**29 horas**

Cuadro 3.3: Tareas del Sprint 0

3.5.2. Sprint 1

Este sprint abarca las fechas comprendidas entre el 01/03/2024 y el 15/03/2024. Se utilizará para finalizar las tareas pendientes en el backlog, y se continuará con la parte inicial de documentación del proyecto. Además, se preparará el entorno de trabajo, instalando todos los programas, lenguajes, extensiones, etc. que se ha detectado que serán necesarios para el desarrollo del proyecto.

También se empleará este sprint para adquirir conocimientos más profundos sobre las tecnologías que se van a utilizar, en especial sobre Angular, ya que nunca he trabajado con esta versión.

Se dedicará una parte del sprint a la creación de un primer módulo para el inicio de sesión, de forma que se da comienzo a la parte de desarrollo, además de comprobar que los entornos están correctamente instalados y disponen de conexión entre ellos.

Estas tareas, recogidas en la tabla 3.4 han requerido el uso de 30 horas.

Tarea	Descripción	Estado	Tiempo
DOC - 003	Definición de historias de usuario.	Completa	2h
DOC - 006	Redactar el estado del arte.	Completa	2h
DES - 001	Preparar el entorno de trabajo.	Completa	4h
DOC - 007	Aprendizaje Angular 17.	Completa	11h
DOC - 008	Definición de la metodología.	Completa	2h
DOC - 009	Investigación sobre Spring Boot.	Completa	4h
DES - 002	Creación de la página inicial para iniciar sesión.	Incompleta	5h
Total			**30 horas**

Cuadro 3.4: Tareas del Sprint 1

3.5.3. Sprint 2

Este sprint abarca las fechas comprendidas entre el 15/03/2024 y el 29/03/2024. Se empleará para continuar con la parte de la documentación inicial que no se había realizado, para poder enfocarse en el desarrollo de la aplicación. En conjunto, como parte de la investigación, se dedicará una pequeña parte del sprint a buscar páginas y aplicaciones similares para profundizar en las capacidades actuales de estas tecnologías de conversión.

Además, se comenzará a investigar sobre las IA de conversión de audio a texto, con el fin de decidir cuál se utilizará en el proyecto; junto con la investigación de otras herramientas que no se habían tenido en cuenta pero van a ser necesarias para el proyecto, como aplicaciones de diseño de diagramas o de gestión de tareas.

Se continúa con el desarrollo de la aplicación. Una vez se dispone de la interfaz gráfica necesaria para poder iniciar sesión, es necesaria la creación en base de datos de una tabla que almacene los usuarios y sus datos, las funciones necesarias para este almacenamiento y todas las conexiones entre los distintos entornos, desde la introducción del usuario de los datos hasta su almacenamiento; junto con la creación de un módulo que permita a los usuarios registrarse en el sistema.

Este sprint, detallado en la tabla 3.5 ha abarcado un total de 33 horas de trabajo.

Tarea	Descripción	Estado	Tiempo
DES - 002	Creación de la página inicial para iniciar sesión.	Completa	4h
DOC - 010	Búsqueda de páginas similares.	Completa	2h
DOC - 011	Redactar el apartado de planificación.	Completa	6h
DOC - 012	Redactar el apartado de tecnologías utilizadas.	Incompleta	4h
DOC - 013	Investigación de otras herramientas necesarias.	Completa	2h
DOC - 014	Investigación sobre las IA de transcripción de audio a texto.	Completa	4h
DES - 003	Creación de la página de registro.	Completa	4h
DES - 004	Creación de la estructura de BBDD y su conexión con la aplicación.	Incompleta	7h
Total			**33 horas**

Cuadro 3.5: Tareas del Sprint 2

3.5.4. Sprint 3

Este sprint abarca las fechas comprendidas entre el 29/03/2024 y el 12/04/2024. A lo largo de este sprint, se terminará la parte de la documentación que quedaba pendiente, incluida la parte de documentar los sprints que se han realizado.

Sobre la página de login que se había creado, se han añadido nuevas funcionalidades, teniendo la opción de iniciar sesión sin registro. Para diferenciar las funciones de un usuario que inicia sesión de uno que no, se ha creado una barra de menú que contendrá las opciones extra que puede hacer un usuario registrado.

En cuanto al formulario para registrarse en la aplicación, se han aplicado mejoras para obligar a que la contraseña cumpla unos estándares de seguridad, que el correo tenga una estructura determinada y que tanto correo como nombre de usuario no estén en uso.

Además, se han modificado los estilos visuales de la aplicación, para adaptarla a un enfoque más moderno y colorido. Junto con ello, en la parte de las mejoras, se descubrió que las contraseñas estaban visibles al redireccionar desde la página del login, por lo que se aplicaron las correcciones necesarias para ocultar esta información.

Podemos observar el detalle de este sprint en la tabla 3.6.

Tarea	Descripción	Estado	Tiempo
DES - 004	Creación de la estructura de BBDD y su conexión con la aplicación.	Completa	1h
DOC - 012	Redactar el apartado de tecnologías utilizadas.	Completa	4h
DOC - 015	Redactar el trabajo de los sprints realizados.	Completa	3h
DOC - 016	Redactar el apartado de análisis del proyecto.	Completa	6h
MEJ - 001	Añadir funcionalidad para recuperar contraseña en la página de iniciar sesión.	Completa	3h
DES - 005	Añadir funcionalidad para iniciar sin registro.	Completa	1h
MEJ - 002	Modificar los estilos de la aplicación.	Completa	3h
MEJ - 003	Modificar la página de registro para poner contraseñas seguras y no dejar enviar el formulario si no se cumplen varias condiciones.	Completa	5h
DES - 006	Diseñar barra de menú y la navegación de la aplicación.	Completa	4h
ERR - 001	Ajustar las funciones de navegación a la página principal para almacenar el token de sesión de forma segura.	Completa	2h
Total			**32 horas**

Cuadro 3.6: Tareas del Sprint 3

3.5.5. Sprint 4

Este sprint abarca las fechas comprendidas entre el 12/04/2024 y el 26/04/2024. Durante este sprint, se ha mejorado de nuevo la página de registro; ahora, las comprobaciones sobre si el nombre de usuario o correo están en uso se realizarán de forma dinámica, permitiendo al usuario saberlo antes de dar al botón de registro.

También se ha modificado la página para recuperar contraseña, de forma que primero se deba introducir correo y nombre de usuario y solo si es válido y coincide se permite poner una nueva contraseña, de esta forma, optimizamos las llamadas a la base de datos.

Además, se han corregido las funciones de almacenamiento de los usuarios en la base de datos, ya que estaban dando algunos errores en el almacenamiento de las contraseñas con caracteres extraños.

Por último, se ha probado toda esta funcionalidad entera, y se han comenzado a crear algunas funciones exclusivas de usuarios registrados, como el borrar su cuenta o el modificar sus datos.

Estas tareas, detalladas en la tabla 3.7 han requerido un total de 30 horas.

Tarea	Descripción	Estado	Tiempo
MEJ - 004	Modificar las comprobaciones de usuario y para realizarse de forma dinámica.	Completa	3h
MEJ - 005	Modificar el registro para pedir dos veces la contraseña.	Completa	2h
MEJ - 006	Modificar la página de recuperar contraseña para realizarse en dos pasos.	Completa	3h
ERR - 002	Modificar las funciones de almacenamiento de los usuarios en la BBDD.	Completa	4h
MEJ - 007	Añadir mensajes de error explicativos.	Completa	2h
PRU - 001	Probar todas las funcionalidades implementadas, registro, inicio de sesión y redirección a la página de la aplicación.	Completa	4h
DES - 007	Creación de la función para eliminar cuenta.	Completa	3h
DES - 008	Creación de la vista y las funciones para modificar los datos del usuario.	Completa	7h
DES - 009	Implementar la función para cerrar sesión y eliminar el token de sesión.	Completa	2h
Total			**30 horas**

Cuadro 3.7: Tareas del Sprint 4

3.5.6. Sprint 5

Este sprint abarca las fechas comprendidas entre el 26/04/2024 y el 10/05/2024. Por motivos personales, durante este sprint se dispuso de menos horas para poder trabajar, lo que queda reflejado en la tabla.

Durante este sprint, se ha modificado la función para borrar la cuenta del usuario, anteriormente, cuando el usuario pulsaba la opción del menú para borrar la cuenta, esta se borraba directamente; ahora, se ha mejorado añadiendo un modal para pedir confirmación antes de realizar la operación de forma que no se borre al pulsar la opción por error.

También se ha continuado el desarrollo, ya enfocándose en la parte de las transcripciones, creando la tabla donde se almacenarán y se ha creado la vista donde el usuario puede consultar sus transcripciones realizadas, junto con las llamadas a BBDD para la carga de los datos.

En total, en este sprint se han podido emplear 21 horas de trabajo, detalladas en la tabla 3.8.

Tarea	Descripción	Estado	Tiempo
MEJ - 008	Modificar la función de eliminar cuenta para pedir confirmación.	Completa	1h
DES - 010	Creación de la tabla para almacenar las transcripciones y sus relaciones con la tabla usuario.	Completa	2h
PRU - 002	Pruebas de inserción y borrado sobre la tabla creada.	Completa	2h
DES - 011	Añadir datos de prueba en la nueva tabla.	Completa	1h
DES - 012	Creación de la página y las funciones para mostrar transcripciones anteriores del usuario.	Completa	13h
DOC - 017	Redactar el trabajo de los sprints realizados.	Completa	2h
Total			**21 horas**

Cuadro 3.8: Tareas del Sprint 5

3.5.7. Sprint 6

Este sprint abarca las fechas comprendidas entre el 10/05/2024 y el 24/05/2024. Durante el transcurso del mismo, se van a desarrollar algunas mejoras de la aplicación, como la inserción de mensajes de error, deshabilitar botones y funcionalidades si no pueden ser usados en ese momento, o realizar el borrado de la cuenta a través de un modal para evitar que el usuario pueda realizar otras operaciones.

Además, se ha diseñado y probado el módulo que contiene las estadísticas del usuario, así como sus estilos acordes al resto de la aplicación.

Por último, se ha comenzado a desarrollar la página principal de la aplicación, consistente en la realización de las transcripciones de los ficheros de audio.

Estas tareas, que podemos ver en la tabla 3.9 han requerido un total de 29 horas de trabajo.

Tarea	Descripción	Estado	Tiempo
MEJ - 009	Añadir paginación en la tabla de transcripciones anteriores.	Completa	1h
DES - 013	Crear el módulo para mostrar estadísticas del usuario.	Completa	10h
DES - 014	Añadir estilos en las nuevas páginas.	Completa	6h
DOC - 018	Redactar el sprint anterior.	Completa	1h
DES - 015	Diseñar un modal para borrar la cuenta.	Completa	1h
MEJ - 010	Añadir mensajes de error en todos los módulos de la aplicación.	Completa	2h
DES - 016	Diseñar la interfaz de la página para generar transcripciones.	Completa	5h
DES - 017	Diseñar la función para transcribir archivos.	Incompleta	3h
Total			**29 horas**

Cuadro 3.9: Tareas del Sprint 6

3.5.8. Sprint 7

Este sprint abarca las fechas comprendidas entre el 24/05/2024 y el 07/06/2024. El desarrollo del mismo está marcado por la implementación de la funcionalidad principal, la conversión de audio a texto. Esto ha traído varios problemas, dado que la API que se utilizaba, el modelo Whisper de

OpenAI, comenzó a reportar mensajes de cuota excedida. Esto hizo que se exploren otras alternativas completamente gratuitas, pero tras los problemas o poca fiabilidad que aportaban, se ha decidido volver a la idea original realizando el pago necesario para aumentar la cuota de peticiones.

A pesar de ello, aunque este modelo debería poder detectar el idioma del audio, daba resultados nulos, por lo que se tuvo que diseñar una funcionalidad extra para obtenerlo.

Una vez se ha tenido la funcionalidad, se ha utilizado el resto del sprint para recopilar audios, realizar pruebas, realizar la funcionalidad para crear y almacenar los textos en ficheros PDF, y mejoras sobre este módulo y el resto de módulos que han sufrido modificaciones una vez se obtienen los datos de las transcripciones, en especial el módulo de las estadísticas del usuario.

En total, en este sprint se han utilizado 32 horas, como se desglosa en la tabla 3.10.

Tarea	Descripción	Estado	Tiempo
DOC - 019	Recopilar audios de prueba.	Completa	2h
DES - 017	Añadir la API para las conversiones y realizar la funcionalidad.	Completa	7h
DES - 018	Realizar la funcionalidad de la transcripción de audio.	Completa	6h
DES - 019	Completar la interfaz de la página para la visualización de los datos.	Completa	2h
DES - 020	Añadir funcionalidad para detectar el idioma del audio.	Completa	3h
DES - 021	Añadir funcionalidad para la creación de ficheros PDF con el texto obtenido.	Completa	2h
PRU - 003	Pruebas sobre las funciones creadas.	Completa	2h
ERR - 002	Modificación de la función de transcripción para que se realicen los ficheros en orden.	Completa	1h
MEJ - 011	Añadir listado de formatos permitidos, mensajes de error y mejoras de usabilidad.	Completa	1h
MEJ - 012	Añadir otras mejoras menores como bloquear la página durante el proceso y manejo de archivos a través de carpetas temporales hasta la ruta final.	Completa	2h
MEJ - 013	Realizar modificaciones sobre el resto de módulos una vez se han obtenido las transcripciones.	Completa	4h
Total			**32 horas**

Cuadro 3.10: Tareas del Sprint 7

3.5.9. Sprint 8

Este sprint abarca las fechas comprendidas entre el 07/06/2024 y el 21/06/2024. Durante su transcurso, se terminará el desarrollo de la aplicación, realizando pruebas globales sobre todos los módulos y funcionalidades de la misma.

También se empleará para redactar los sprints que no han sido redactados durante estas últimas etapas centradas en el desarrollo.

Además, se retomará el trabajo de la documentación del proyecto, realizando actualizaciones sobre las partes que hayan sufrido modificaciones y no estén alineadas con el desarrollo. Y se continuará el desarrollo de la misma con los apartados restantes para su finalización.

Estas tareas, detalladas en la tabla 3.11, han consumido un total de 31 horas de trabajo.

Tarea	Descripción	Estado	Tiempo
PRU - 004	Pruebas sobre todas las funcionalidades de la aplicación.	Completa	8h
DOC - 020	Redactar los sprints anteriores.	Completa	2h
DOC - 021	Actualizar el apartado de las tecnologías utilizadas y otros aspectos de la memoria.	Completa	2h
DOC - 022	Desarrollo del capítulo de Diseño	Completa	6h
DOC - 023	Desarrollo del capítulo de Implementación	Completa	7h
DOC - 024	Desarrollo del capítulo de Pruebas	Completa	7h
Total			**32 horas**

Cuadro 3.11: Tareas del Sprint 8

3.5.10. Sprint 9

Este sprint abarca las fechas comprendidas entre el 21/06/2024 y el 05/07/2024. Durante su transcurso, se han realizado las mejoras en la documentación obtenidas de la interacción con el tutor del proyecto.

Tarea	Descripción	Estado	Tiempo
DOC - 025	Mejoras por retroalimentación del tutor y últimas correcciones	Completa	4h
Total			**4 horas**

Cuadro 3.12: Tareas del Sprint 9

3.5.11. Sprint 10

Este sprint abarca las fechas comprendidas entre el 04/10/2024, día donde se retomó el trabajo, y el 18/10/2024. A lo largo de este sprint, se comenzó a trabajar en realizar pruebas automáticas para la aplicación. La primera idea fue realizarlas a través de pruebas unitarias en angular. Debido al desconocimiento de su funcionamiento por no haberlas utilizado en anteriores ocasiones, una parte de las horas empleadas en el sprint fueron dedicadas a profundizar sobre el tema. Una vez se tenían los conocimientos básicos para el desarrollo, se comenzó con el mismo, abarcando de esta manera todo el sprint; sin embargo, se han ido encontrando errores a la hora de probar los endpoints, por lo que no ha podido finalizarse la tarea.

Tarea	Descripción	Estado	Tiempo
DOC - 026	Estudio sobre la tecnología de pruebas unitarias en Angular	Completa	8h
DES - 022	Comienzo del desarrollo de las pruebas unitarias	Incompleta	15h
Total			**23 horas**

Cuadro 3.13: Tareas del Sprint 10

3.5.12. Sprint 11

Este sprint abarca las fechas comprendidas entre el 18/10/2024 y el 01/11/2024. Para comenzar el sprint, se continuó el trabajo en las pruebas unitarias. Sin embargo, debido a los errores encontrados en el sprint anterior, se hacía difícil el avance, por lo que se habló con el tutor para proponer el uso de la herramienta Postman [4.10] para probar los endpoints de la aplicación. Una

vez se obtuvo el visto bueno, se abandonó la alternativa de las pruebas unitarias en angular para comenzar la creación de un conjunto de pruebas en Postman.

Tarea	Descripción	Estado	Tiempo
DES - 022	Continuación del desarrollo de las pruebas unitarias	Abandonada	7h
DES - 023	Comienzo del desarrollo de colección de pruebas en Postman	Incompleta	15h
Total			**22 horas**

Cuadro 3.14: Tareas del Sprint 11

3.5.13. Sprint 12

Este sprint abarca las fechas comprendidas entre el 01/10/2024 y el 15/11/2024. A lo largo de este sprint, se continuó el trabajo de creación de las pruebas en Postman. Además, debido al trabajo para realizar las pruebas, se fueron realizando mejoras en el código de algunos de los endpoints para corregir pequeños errores que se detectaban, en especial, errores con los parámetros recibidos, de forma que no se controlaran únicamente desde el front de la aplicación. Una vez finalizado, se realizaron pruebas para comprobar que funcionaba correctamente, y se comenzó a redactar las últimas partes de la memoria, tanto los sprints adicionales como realizar las modificaciones necesarias para actualizarla con los nuevos cambios y añadir las nuevas tecnologías empleadas.

Tarea	Descripción	Estado	Tiempo
DES - 023	Continuación del desarrollo de colección de pruebas en Postman	Completa	11h
MEJ - 014	Mejoras en el código de los endpoints de la aplicación	Completa	5h
DOC - 026	Modificaciones de la memoria para su actualización con los cambios realizados, redacción de las nuevas pruebas y redacción de los sprints adicionales.	Incompleta	5h
Total			**21 horas**

Cuadro 3.15: Tareas del Sprint 12

3.5.14. Sprint 13

Este sprint abarca las fechas comprendidas entre el 15/10/2024 y el 29/11/2024. A lo largo de este sprint, se ha continuado el trabajo de redacción de la memoria que no fue posible finalizar en el sprint anterior.

Tarea	Descripción	Estado	Tiempo
DOC - 026	Modificaciones de la memoria para su actualización con los cambios realizados, redacción de las nuevas pruebas y redacción de los sprints adicionales.	Completa	7h
Total			**7 horas**

Cuadro 3.16: Tareas del Sprint 13

3.5.15. Sprint 14

Este sprint abarca las fechas comprendidas entre el 29/11/2024 y el 13/12/2024. Durante este último sprint, se han realizado modificaciones en la memoria debido a retroalimentaciones apor-

tadas por el tutor, centrándose en el apartado acerca de la funcionalidad y limitaciones de las tecnologías empleadas.

Tarea	Descripción	Estado	Tiempo
DOC - 027	Modificaciones de la memoria por retroalimentaciones del tutor. Redacción del apartado de funcionalidad y limitaciones.	Completa	8h
Total			**8 horas**

Cuadro 3.17: Tareas del Sprint 14

4 Tecnologías

4.1. Angular

Angular [44] es un framework de desarrollo de aplicaciones web de JavaScript escrito en TypeScript, desarrollado por Google. Su marco de diseño está centrado en la creación de aplicaciones de una sola página (SPA).

Utiliza el patrón de arquitectura Modelo Vista Controlador (MVC) [45], que sirve como guía de organización de los componentes. Este patrón separa los componentes para la representación de la información con los componentes para la interacción del usuario.

- El **modelo** es el encargado del manejo de datos y lógica.
- El **controlador** se encarga del manejo de eventos e invocación de peticiones.
- La **vista** presenta el modelo al usuario a través de una interfaz.

En dicho modelo, los componentes interactúan entre sí de la siguiente manera:

- El usuario interactúa con la interfaz (Vista) a través de enlaces, botones, imputs, etc.
- Estas interacciones son recibidas por el controlador, que se encarga de gestionar el evento y acceder al modelo realizando las modificaciones necesarias a raíz de la acción realizada por el usuario.
- El modelo gestiona las peticiones recibidas y actualiza la vista, quedando a la espera de nuevas acciones del usuario.

Se ha utilizado la versión de Angular 17 [46], lanzado en noviembre de 2023. Sin entrar al detalle en las novedades e historia de las versiones Angular, esta versión está caracterizada por el uso de "standalone components"[47][48]; un nuevo tipo de componentes que no necesitan ser declarados en un NgModule, es decir, pueden ser utilizados en otro componente sin necesidad de importarse en un NgModule, lo que proporciona una forma más simplificada de la creación de aplicaciones.

En Angular, los componentes son la piedra angular de la aplicación. Cada componente encapsula una parte de la interfaz de usuario y la lógica asociada. Los componentes consisten en tres archivos principales:

- **Archivo TypeScript (.ts):** TypeScript es el lenguaje principal utilizado en Angular. Está basado en JavaScript y añade elementos de la Programación Orientada a Objetos como el interfaces y clases. Contiene la lógica del componente, como la manipulación de datos y la gestión de eventos.
- **Archivo HTML (.html):** Definen la estructura de la interfaz de usuario. Están formados por etiquetas HTML estándar y directivas de Angular como ngFor o NgIf.
- **Archivo CSS (.css o .scss):** Definen los estilos y la apariencia de los componentes en un proyecto Angular. Estos archivos contienen reglas de estilo que se aplican a los elementos HTML dentro de los componentes.

4.2. Spring Boot

Spring Boot [49] es un framework de desarollo basado en Java, diseñado para facilitar la configuración y el desarrollo rápido de aplicaciones. Está basado en el patrón de la Programación Orientada a Aspectos (POA) [50]; paradigma que permite la separación de los módulos de una

aplicación eliminando las dependencias entre los mismos.

Se ha utilizado Spring Boot para realizar la parte relacionada con la lógica de la aplicación y las operaciones sobre bases de datos. Para aplicar el enfoque ya explicado anteriormente de la arquitectura de Modelo-Vista-Controlador, se ha aplicado una división en módulos característica de los proyectos en Spring Boot:

- **Model.** Contiene las clases que representan la estructura de datos. Define las entidades de la aplicación en conjunto con sus atributos.
- **Controller.** Contiene las clases encargadas de la gestión de peticiones. Estas clases coordinan la recepción de peticiones HTTP, interactúan con los modelos y emiten una respuesta en función de los datos obtenidos del servicio.
- **Service.** Contiene las clases que implementan la lógica de negocio. Son las encargadas de la implementación de las operaciones y la devolución de los valores al controlador.
- **Repository.** Contiene las clases que gestionan el acceso a las Bases de Datos. Contienen los métodos necesarios para realizar operaciones CRUD.

4.3. MySQL Workbench

MySQL Workbench [51] es una herramienta de diseño y gestión de bases de datos relacionales que utiliza lenguaje SQL (Structured Query Language), diseñado para poder realizar operaciones de creación, inserción, actualización y eliminación de datos.

Esta herramienta proporciona un único entorno de desarrollo para la gestión, creación, modificación y consulta de las bases de datos, facilitando todas las tareas de administración de las distintas bases de datos de la aplicación.

4.4. Maven

Maven [52] es una herramienta de gestión de proyectos Java utilizada para la gestión de dependencias del proyecto a través de una configuración basada en formato XML. Se utiliza por defecto en proyectos de Spring Boot.

El fichero de configuración de Maven es denominado pom.xml (Project Object Model), y contiene la información del proyecto, su estructura, dependencias, configuración y complementos.

4.4.1. Ciclo de vida Maven

El ciclo de vida de un proyecto Maven [53][54] permite la organización de la gestión del proyecto desde la etapa de compilación hasta la etapa de despliegue de la aplicación. Se compone de una serie de fases que representan etapas específicas del proceso de desarrollo. Estas partes del ciclo de vida son:

- **Validate.** Se valida que el proyecto sea correcto y esté disponible toda la información.
- **Compile.** Se compila el código fuente y se generan los archivos del directorio Target.
- **Test.** Se ejecutan pruebas unitarias del código compilado sin necesidad de que el código ya se encuentre empaquetado.
- **Package.** Se empaqueta el código compilado en el formato requerido , como JAR (Java ARchive) o WAR (Web Application Resource).

- **Verify.** Se verifica el resultado de las pruebas para cumplir con los criterios de calidad necesarios.
- **Install.** Se instala el artefacto generado en el repositorio local de Maven, pudiendo así usarse en otros proyectos Maven.
- **Deploy.** Se copia el paquete generado en un repositorio remoto.

4.4.2. Estructura de un proyecto Maven

Al ser una herramienta de gestión de proyectos, Maven nos provee de una estructura de directorios en los que organizar el proyecto de forma eficiente siguiendo varias convenciones:

- **src.** Directorio raíz del proyecto.
  - **main**
    - **java.** Contiene el código fuente separado en los distintos paquetes.
    - **resources.** Contiene los recursos necesarios como imágenes o ficheros de configuración.
  - **test**
    - **java.** Contiene el código fuente de las pruebas unitarias.
    - **resources.** Contiene ficheros de recursos necesarios para las pruebas, como archivos de configuración específicos.
- **target.** Se genera durante la compilación del proyecto y contiene los artefactos generados.
- **pom.** Fichero de configuración que indica plugins, dependencias, versiones, etc.

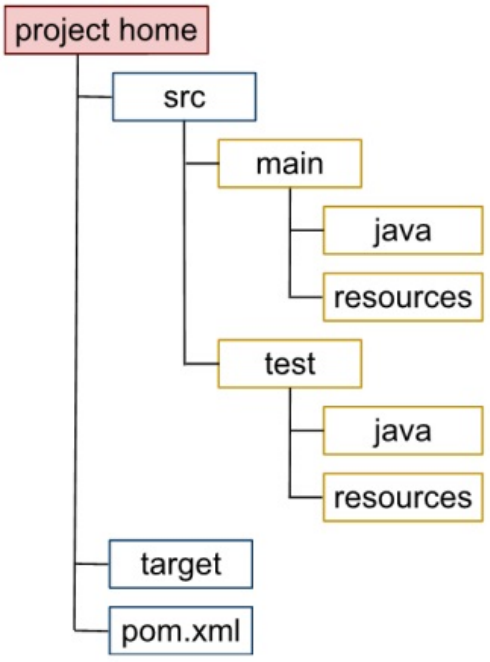

Figura 2: Estructura de proyectos Maven. Fuente: [55]

4.5. Whisper

El modelo Whisper [26] es un sistema de reconocimiento automático de voz, desarrollado por la compañía OpenAI. Utiliza la inteligencia artificial para la transcripción de los audios a texto, y es capaz de detectar una amplia variedad de idiomas y formatos de audio.

Esta tecnología ha sido entrenada con 680000 horas de datos multilenguaje, colocándola como una de las más precisas gracias a una baja tasa de errores, y a la capacidad para interpretar correctamente las pausas de la conversación, aportando mayor precisión en el uso de los signos de puntuación.

En este proyecto, se ha utilizado el modelo de Whisper large-v2 para la inserción de las funcionalidades requeridas para la conversión de ficheros de audio a ficheros de texto. En la etapa de desarrollo, se probaron otras dos alternativas para la implementación de esta funcionalidad: Google Cloud Speech-to-Text y Amazon Transcribe, si bien fueron descartadas en favor de Whisper debido a su mayor facilidad de integración y de obtención de resultados precisos.

Para la integración de este modelo con el lenguaje Java, se ha utilizado la biblioteca openai-gpt3-java [56], que facilita la integración de la API de OpenAI de una forma sencilla. Esta biblioteca construye, de manera interna, una solicitud HTTP con una clave de API para la autenticación, y crea un objeto JSON con parámetros como el texto a transcribir y el modelo a utilizar.

Una vez construido el formato de la petición y el cuerpo de la solicitud, se encarga de realizar una solicitud POST al servidor de OpenAI; y recibe una respuesta en formato JSON que deserializa a objetos Java fáciles de manejar.

Esta biblioteca permite la utilización del modelo large-v2 [57]. Este modelo es el más avanzado y preciso de todos los desarrollados, aunque también es el modelo más exigente. Durante la realización del trabajo se comenzaron realizando pruebas con este modelo, y, al ser satisfactorias, se optó por utilizarlo sin probar el resto para implementar la mayor fiabilidad posible.

Entre las características de este modelo, destacan una mejora de precisión respecto a anteriores, su capacidad para realizar transcripciones en un gran número de idiomas y la capacidad de detección automática de idiomas. Además, permite realizar transcripciones de ficheros de audio más grandes a través de algoritmos de fragmentación.

Pese a ser el modelo más avanzado, tiene unos límites, pudiendo arrojar resultados inexactos en contextos ruidosos o audios de baja calidad, así como dificultades en la precisión de audios con acentos que no permitan entender correctamente el idioma.

Otro de los limitantes del modelo, reside en la cantidad y calidad de entrenamiento al que haya sido sometido; esto puede ocasionar que no realice transcripciones fidedignas en lenguajes menos representados en los datos de entrenamiento; ocasionando una fuerte dependencia de su robustez a los datos de entrenamiento.

4.5.1. Funcionamiento de Whisper

Respecto al funcionamiento de Whisper [58], se basa en un sistema de codificador-decodificador. El audio recibido, se muestrea a 16000Hz y se calcula una representación del espectograma log-Mel (que es una transformación similar al cálculo del logaritmo de un espectograma, y muestra el contenido de una señal de audio a lo largo del tiempo, pero aproxima la respuesta de frecuencia no lineal del oído humano)[59] de 80 canales en ventanas de 25ms, con un desplazamiento de 10ms.

A continuación, se escala el conjunto de entradas para normalizarlas en valores entre 1 y -1, con una media aproximada de 0. Estas entradas, son procesadas por el codificador a través de

un módulo inicial de dos capas de convulción con un ancho de filtro igual a 3, y la función de activación GELU [60], donde la segunda capa de convolución tiene un paso de dos. Tras esto, se añaden incrustaciones sinusoidales a la salida y se aplican los bloques Transformer del codificador.

Por último, el decodificador, que tiene el mismo ancho y número de bloques transformer que el codificador, utiliza incrustaciones aprendidas y representaciones de tokens de entrada y salida compartidas. Esta arquitectura se muestra en la siguiente figura.

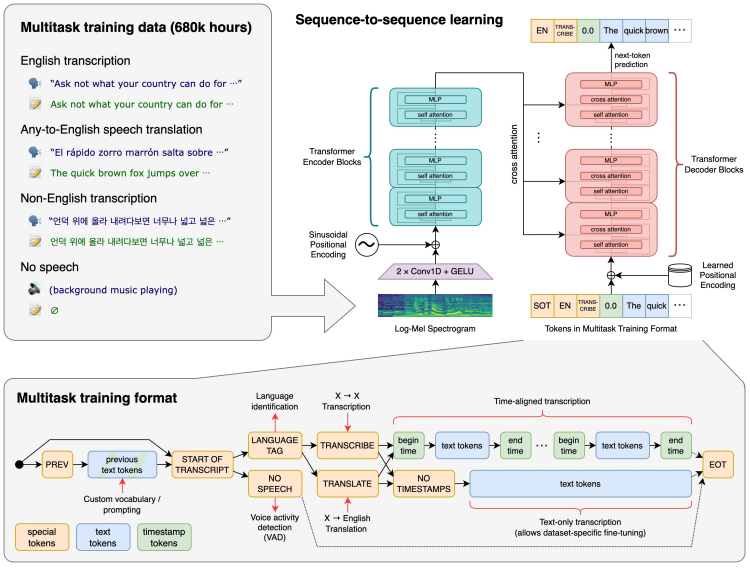

Figura 3: Arquitectura del modelo Whisper. [61]

4.5.2. Modelos Whisper

Whisper cuenta con seis distintos modelos, indicados en la tabla 4.1. La fiabilidad de cada uno de los modelos, se calcula a través de la tasa de error de palabras (WER) [62], que calcula la frecuencia con la que se comenten errores durante el proceso de transcripción. Esta tasa se calcula teniendo en cuenta el número de sustituciones de palabras (S), el número de eliminaciones de palabras (D), el número de inserciones de nuevas palabras (I), y el número de aciertos (C); a través de la fórmula $WER = (S + D + I)/(S + D + C)$.

Modelo	Parametros	Idiomas soportados	VRAM Requerida	Velocidad relativa
tiny	39 millones	Multilingüe	1 GB	10 veces más rápido
base	74 millones	Multilingüe	1 GB	7 veces más rápido
small	244 millones	Multilingüe	2 GB	4 veces más rápido
medium	769 millones	Multilingüe	5 GB	2 veces más rápido
large	1550 millones	Multilingüe	10 GB	Tiempo real
turbo	809 millones	Multilingüe	6 GB	8 veces más rápido

Cuadro 4.1: Modelos Whisper

El rendimiento de cada modelo está condicionado al idioma utilizado. En la imagen 4 podemos ver la tasa de error de palabras de los modelos large-v2 y large-v3 respecto a dos conjuntos de datos de prueba; Common Voice 15 [63], una base de datos de contribuyentes voluntarios con casi 21000 horas validadas de múltiples idiomas; y FLEURS [64], un conjunto de datos de habla de 102 idiomas y 1400 horas de duración.

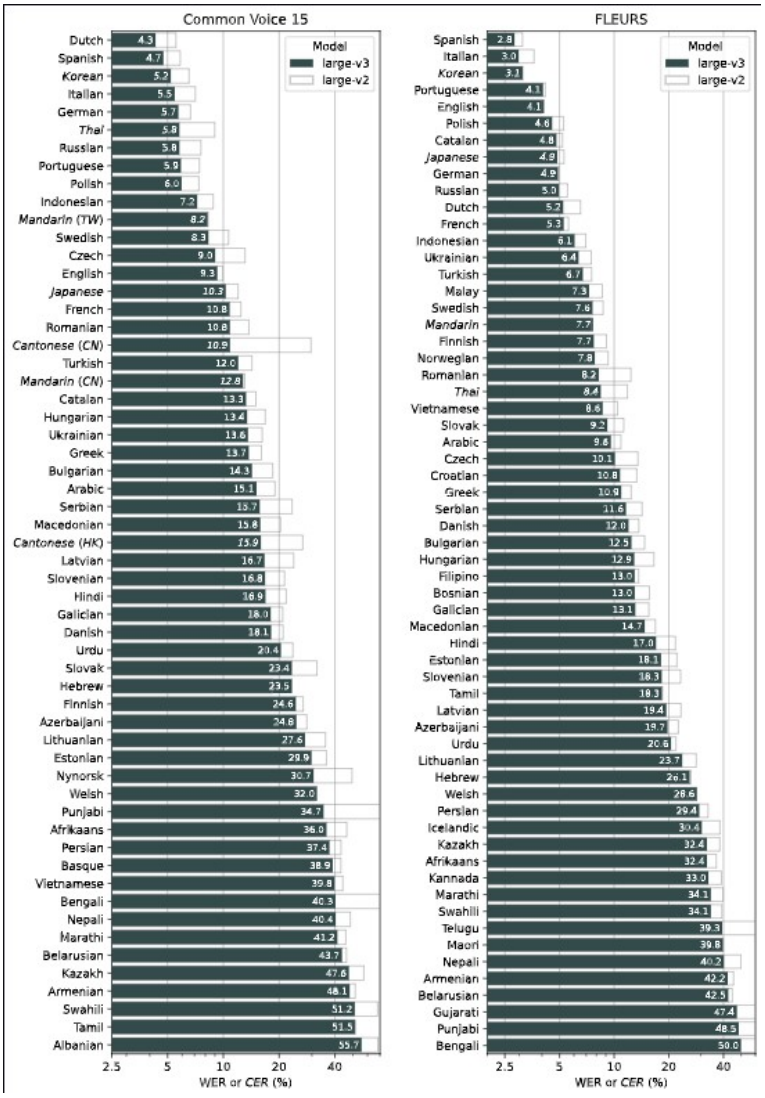

Figura 4: Tasa de error de palabras de los modelos large-v2 y large-v3. [65]

Podemos observar que la tasa de error de palabras de los principales idiomas, en especial en español, es una tasa de error bastante baja, lo que convierte al modelo en una buena opción debido a su robustez. Sin embargo, también podemos comprobar que hay idiomas con una tasa de error cercana o superior al 50 %; debemos tener esto en cuenta ya que significa que las transcripciones realizadas de dichos idiomas es posible que contengan una gran cantidad de errores.

4.6. Eclipse

Eclipse [66] es un Entorno de Desarrollo Integrado (IDE) de código abierto utilizado para el desarrollo de aplicaciones en distintos lenguajes de programación. Fue desarrollado por la compañía

IBM en 2001 y traspasado unos años más tarde a la organización Eclipse Foundation.

Es un entorno de desarrollo ampliamente utilizado debido a que tiene muchas características destacables, como un editor de código con analizador sintáctico, autocompletado, navegación rápida por el código, soporte para múltiples lenguajes, una gran variedad de módulos y plugins para proporcionar distintas funcionalidades, integración con diversos sistemas, etc. Además de contar con grandes funciones de adaptación y personalización a través de distintas vistas, perspectivas y complementos

Este IDE ha sido utilizado en su versión 2024-03 para el desarrollo y mantenimiento de la parte realizada con Spring Boot, debido a su amplia compatibilidad con el lenguaje de programación Java, ya que facilita las tareas de desarrollo, depuración y despliegue de aplicaciones Spring Boot.

4.7. Visual Studio Code

Visual Studio Code [67] es un entorno de desarrollo de código abierto desarrollado por Microsoft en 2015 y con compatibilidad con sistemas operativos Windows, Linux y macOs.

Entre sus características más destacadas están la facilidad de uso y el soporte que ofrece a una gran variedad de lenguajes de programación, además de contar con una amplia gama de extensiones, permitiendo ser configurado para unas necesidades específicas. Cuenta con características como resaltado de sintaxis, autocompletado, control de versiones con Git y una terminal integrada.

En el desarrollo del proyecto, se ha utilizado esta herramienta para el desarrollo y mantenimiento de la parte realizada con Angular, gracias a su compatibilidad con Typescript, HTML y SCSS, las extensiones existentes para angular y la integración con la herramienta Angular CLI, herramienta básica para la creación de competentes a través de comandos predefinidos.

4.8. Overleaf

Overleaf [68] es un editor de texto LaTeX desarrollado por la empresa WriteLaTeX Limited en 2011. Utiliza un enfoque para la redacción de documentos de carácter científico y técnico.

El texto sin formato LaTeX, es un sistema de composición de textos basado en instrucciones de bajo nivel orientado a la creación de documentos de alta calidad tipográfica. Por ello, ha sido el editor elegido para la creación de esta memoria.

4.9. Astah

Astah [69] es una herramienta de modelado UML desarrollada por la empresa Change Vision en el año 2006. Permite la creación de modelos y diagramas software, como diagramas UML, diagramas de clases, diagramas de secuencia, de casos de uso, etc.

En este proyecto, esta herramienta ha sido utilizada para la creación de los diagramas del apartado de análisis del proyecto, incluyendo los diagramas de casos de uso y sus descripciones, y los diagramas de secuencia de la aplicación.

4.10. Postman

Postman [70] es una herramienta para la monitorización y testeo de una API de forma automatizada, a través de la realización de solicitudes HTTP. Fue creada como una extensión de Chrome,

pero evolucionó hacia una aplicación con una interfaz gráfica sencilla que permite agrupar las solicitudes en colecciones, ofreciendo una forma ordenada y sencilla de automatizar el proceso de pruebas.

En la realización del proyecto, postman ha sido utilizado para crear una colección de pruebas de todos los endpoints que se han creado en la aplicación.

4.11. OpenAPI

OpenAPI [71], anteriormente conocido como Swagger, es una librería creada para la documentación de aplicaciones. A través de anotaciones predefinidas en el código, genera un recurso web que contiene toda la documentación desarrollada, tanto de controladores con sus endpoints, descripciones, parámetros y comportamientos, como del modelo de datos de la aplicación; añadiendo una interfaz simple y agradable.

En la realización del proyecto, se ha utilizado OpenAPI para el desarrollo de la documentación. (9.3).

5 Funcionamiento y limitaciones del proceso de transcripción

En esta sección se va a tratar de describir los aspectos técnicos más importantes de cómo la aplicación realiza las transcripciones de audio, unas guías básicas sobre las decisiones de diseño más relevantes para el desarrollo de la parte principal de la misma; así como las limitaciones que presenta.

5.1. Descripción del proceso

Para poder realizar transcripciones, la aplicación ofrece al usuario un cuadro sobre el que arrastrar o seleccionar uno o varios archivos.

Una vez que el usuario ha subido uno o varios archivos, comienza el proceso de transcripción de los mismos. Este proceso comienza bloqueando la aplicación para evitar que el usuario pueda realizar otras acciones que comprometan el correcto funcionamiento de la aplicación. Tras esto, se comprueba desde el front que todos los ficheros tienen alguna de las extensiones permitidas, indicando un mensaje de error en caso contrario, y no realizando la transcripción de ninguno de ellos.

Si todos los ficheros tienen un formato válido, se realiza una llamada al endpoint /uploadAudio, que mueve los ficheros a la carpeta temporal del sistema operativo.

La aplicación espera a obtener una respuesta (true o false) de este método, para asegurarse que todos los ficheros han sido movidos antes de comenzar con el proceso. Una vez se obtiene una respuesta afirmativa, se comienza a recorrer el array de ficheros uno a uno y, para cada uno, se realiza una llamada al endpoint /transcribe, que gestiona las operaciones necesarias para la transcripción.

Esta función comienza creando el servicio Whisper de OpenAI (4.5). Se ha elegido este modelo debido a su alta fiabilidad y su sencilla integración con el proyecto; durante las primeras etapas del desarrollo, se realizaron pruebas de integración y pruebas de conversión de distintos conversores, y se acabó eligiendo Whisper dado que, aunque no sea gratuito, arrojaba los mejores resultados.

Una vez la aplicación ha creado el modelo que usará para realizar las conversiones, obtiene el fichero recibido por parámetro de la carpeta temporal del S.O. donde se aloja, y lo mueve a una nueva carpeta temporal asociada a la ejecución.

A continuación, se crea una nueva petición de transcripción y se le asigna el modelo creado anteriormente. Si bien el modelo de Whisper es capaz de, además de obtener el texto de la transcripción, detectar otros valores como la duración o el idioma, en una gran mayoría de las ocasiones estos valores resultaban nulos al terminar la transcripción. Por tanto, para obtener el idioma de una forma más precisa, se ha utilizado la biblioteca Lingua [72], que se comprobó que obtenía resultados precisos.

Por último, una vez se han obtenido tanto el texto como el lenguaje de la transcripción, se elimina el fichero de la carpeta temporal para evitar dejar ficheros innecesarios, se crea un fichero PDF al que se le añade el texto obtenido, se calcula la fecha actual y se almacenan todos los registros en la base de datos.

El proceso finaliza devolviendo este objeto a la función del front que ha realizado la llamada, para que pueda gestionar el resultado y mostrárselo al usuario.

Toda la documentación de la aplicación y sus procesos, realizada con OpenAPI, se puede consultar en el Anexo A. (9.3).

5.2. Limitaciones de la aplicación

5.2.1. Limitaciones de los parámetros de entrada

Las primeras limitaciones que nos podemos encontrar dentro de la aplicación, residen en los ficheros aceptados. La cantidad de ficheros que se pueden subir al mismo tiempo no tiene una limitación, si bien es cierto que, por las capacidades del ordenador donde se ha probado, no se ha podido explorar el límite debido a la falta de recursos para realizar todas las operaciones necesarias de forma concurrente.

Aunque el número de archivos no tenga límite, sí existen ciertas restricciones. Los ficheros de audio solo pueden tener formato mp3, mp4, mpeg, mpga, m4a, wav y webm, debido a ser los únicos formatos que Whisper puede transcribir correctamente.

Además, otra limitación que debe tener el archivo de audio, reside en su nombre. Para ser almacenado en la base de datos, se ha diseñado la columna que almacena el nombre del fichero con un formato VARCHAR(45), lo que permite una longitud total de 45 caracteres. Teniendo en cuenta una extensión de 4 o 5 caracteres incluyendo el punto, nos da como resultado que ficheros con un nombre de más de 40 o 41 caracteres no podrán ser almacenados en la base de datos y, por tanto, no se podrá realizar su transcripción.

5.2.2. Limitaciones de Whisper

El mayor número de limitaciones que presenta la aplicación residen en las limitaciones del modelo Whisper de OpenAI. Como se ha comentado en el apartado anterior, el uso de este modelo acota la cantidad de distintos formatos de audio que se pueden transcribir correctamente.

Además, consultando la documentación, podemos observar otras limitaciones referentes a los archivos, ya que el peso de los ficheros está limitado a 25MB.

Otro punto a tener en cuenta derivado del uso de Whisper, son los idiomas que puede detectar. De nuevo, consultando la documentación, comprobamos que soporta los lenguajes afrikáans, árabe, armenio, azerbaiyano, bielorruso, bosnio, búlgaro, catalán, checo, chino, croata, danés, holandés, inglés, estonio, finlandés, francés, gallego, alemán, griego, hebreo, hindi, húngaro, islandés, indonesio, italiano, japonés, canarés, kazajo, coreano, letón, lituano, macedonio, malayo, maratí, maorí, nepalí, noruego, persa, polaco, portugués, rumano, ruso, serbio, eslovaco, esloveno, español, swahili, sueco, tagalo, tailandés tamil, turco, ucraniano, urdu, vietnamita y galés. Por lo que, si no son todos los idiomas existentes, son suficientes para considerar que no deberían existir restricciones a la hora de transcribir un fichero.

Otros de los problemas encontrados residen en cómo funciona el modelo cuando no se trata de un audio claro, de una única persona y en un único idioma. En contextos en los que el audio no es nítido y contiene ruido externo o con baja calidad, a raíz de las pruebas realizadas, se puede comprobar que se obtiene una transcripción que puede disminuir su precisión y aumentar el número de errores en el texto final; a pesar de ello, se pueden seguir realizando transcripciones de una precisión aceptable, a pesar de ser posible la omisión de partes del audio o segmentos del audio con transcripciones incorrectas.

Respecto al funcionamiento de la aplicación cuando se trata de un audio con varios interlocutores, por ejemplo, una conversación entre una o varias personas; el modelo Whisper no tiene la capacidad de diferenciar entre distintos hablantes, por lo que se genera una transcripción igual a la que se generaría si toda la conversación fuera hablada por una única persona. Uno de los

posibles puntos de mejora, sería introducir previamente una forma de diferenciar entre interlocutores para después, realizar la transcripción de cada uno de los segmentos de audio y maquetar el resultado en un fichero que diferenciara las intervenciones de cada una de las personas involucradas.

Por último, uno de los mayores limitantes de la aplicación se produce en contextos donde se utiliza más de un idioma en el mismo audio. En este caso, la aplicación traduce todos los idiomas utilizados al que se utilice de forma mayoritaria. De esta forma, el resultado final contiene todo el texto que se ha podido transcribir a partir del fichero de audio original en un único idioma.

5.2.3. Limitaciones de Lingua

Además de las limitaciones detalladas, podemos encontrar otras limitaciones en la aplicación derivadas del uso de otras librerías. Al utilizar la biblioteca Lingua para la detección de idiomas, volvemos a encontrar limitaciones respecto a los posibles idiomas que se transcriben y su correcta interpretación por parte de la aplicación; sin embargo, la lista de compatibilidades de esta librería incluye todos los mencionados para el modelo Whisper e nicluso alguno más; por tanto, no restringe los idiomas posibles de la aplicación.

Otros aspectos a tener en cuenta en el uso de esta librería para la identificación de los idiomas del texto, se produce en el caso de ser un fichero de audio con múltiples idiomas. Como se ha mencionado anteriormente, cuando se da esta casuística, el fichero resultado únicamente está compuesto del idioma mayoritario, con el resto del texto traducido a éste; por tanto, la librería no aporta ninguna restricción adicional, puesto que solo debe detectar un único idioma.

6 Análisis

6.1. Introducción

A lo largo de este capítulo se documentarán el análisis correspondiente a este proyecto, lo que nos permitirá comprender el alcance del mismo. A través de este análisis se pretende comprender en su totalidad el comportamiento de la aplicación.

Para ello, se detallarán las funcionalidades que debe cumplir la aplicación, entendidas a través de un análisis de requisitos, funcionales y no funcionales. Además, se presentarán diferentes acciones que se pueden realizar con la aplicación a través de los casos de uso.

6.2. Requisitos funcionales (RF)

Los requisitos funcionales son una definición de los servicios que debe implementar el sistema.

En la tabla 6.1 se detallan los requisitos funcionales que debe cumplir la aplicación.

ID	Descripción
RF-1	La aplicación deberá permitir registrar un nuevo usuario.
RF-2	La aplicación deberá permitir iniciar sesión a un usuario registrado.
RF-3	La aplicación deberá permitir a un usuario cambiar su contraseña antes de acceder.
RF-4	La aplicación deberá permitir iniciar sesión para usuarios no registrados.
RF-5	La aplicación deberá permitir a un usuario realizar una conversión.
RF-6	La aplicación deberá permitir descargar la conversión realizada.
RF-7	La aplicación deberá permitir a un usuario registrado consultar sus conversiones realizadas.
RF-8	La aplicación deberá permitir a un usuario registrado descargar sus conversiones realizadas.
RF-9	La aplicación deberá permitir a un usuario eliminar y consultar sus conversiones realizadas.
RF-10	La aplicación deberá permitir a un usuario registrado consultar estadísticas sobre sus conversiones.
RF-11	La aplicación deberá permitir a un usuario registrado modificar sus datos.
RF-12	La aplicación deberá permitir a un usuario registrado eliminar su cuenta.
RF-13	La aplicación deberá realizar conversiones de distintos idiomas.
RF-14	La aplicación deberá permitir diferentes formatos de audio.

Cuadro 6.1: Requisitos Funcionales

6.3. Requisitos no funcionales (RNF)

Los requisitos no funcionales no especifican funcionalidades del sistema, si no que atienden a las propiedades del sistema como seguridad, escalabilidad, restricciones, etc. Proporcionan una guía de la calidad de la aplicación.

En la tabla 6.2 se detallan los requisitos no funcionales que debe satisfacer la aplicación.

ID	Descripción
RNF-1	La aplicación deberá detectar el idioma de forma automática.
RNF-2	La aplicación deberá mantener el idioma del audio en la transcripción.
RNF-3	La aplicación deberá acceder a la base de datos para consultar usuarios y conversiones.
RNF-4	La aplicación deberá tener un tiempo de respuesta dentro de los estándares aceptables.
RNF-5	La aplicación deberá realizar transcripciones lo más precisas posible.
RNF-6	La aplicación debe implementar un mecanismo seguro de autenticación.
RNF-7	La aplicación no debe almacenar datos vulnerables durante las sesiones.
RNF-8	La aplicación deberá tener una interfaz atractiva y fácil de usar.
RNF-9	La aplicación deberá ser compatible con varios navegadores.

Cuadro 6.2: Requisitos No Funcionales

6.4. Casos de uso

Los casos de uso son una herramienta fundamental en el proceso de análisis del sistema. Proporcionan una descripción de las interacciones entre usuarios y sistema, a través de la descripción de las actividades que se pueden realizar.

Para un análisis de los casos de uso de la aplicación, debemos distinguir entre las posibilidades que tiene un usuario con una cuenta creada de un usuario que accede a la aplicación sin registro.

Un usuario sin registro podrá realizar las siguientes opciones:

- Generar una transcripción.
- Descargar la transcripción.

Sin embargo, un usuario registrado en la aplicación tendrá podrá realizar las operaciones:

- Generar una transcripción.
- Descargar la transcripción.
- Consultar sus transcripciones realizadas anteriormente.
- Eliminar sus transcripciones realizadas anteriormente.
- Descargar sus transcripciones realizadas anteriormente.
- Consultar estadísticas sobre sus transcripciones.

- Modificar sus datos de usuario.
- Eliminar todos sus datos.

En el diagrama de la figura 5 podemos ver el diagrama de casos de uso del sistema.

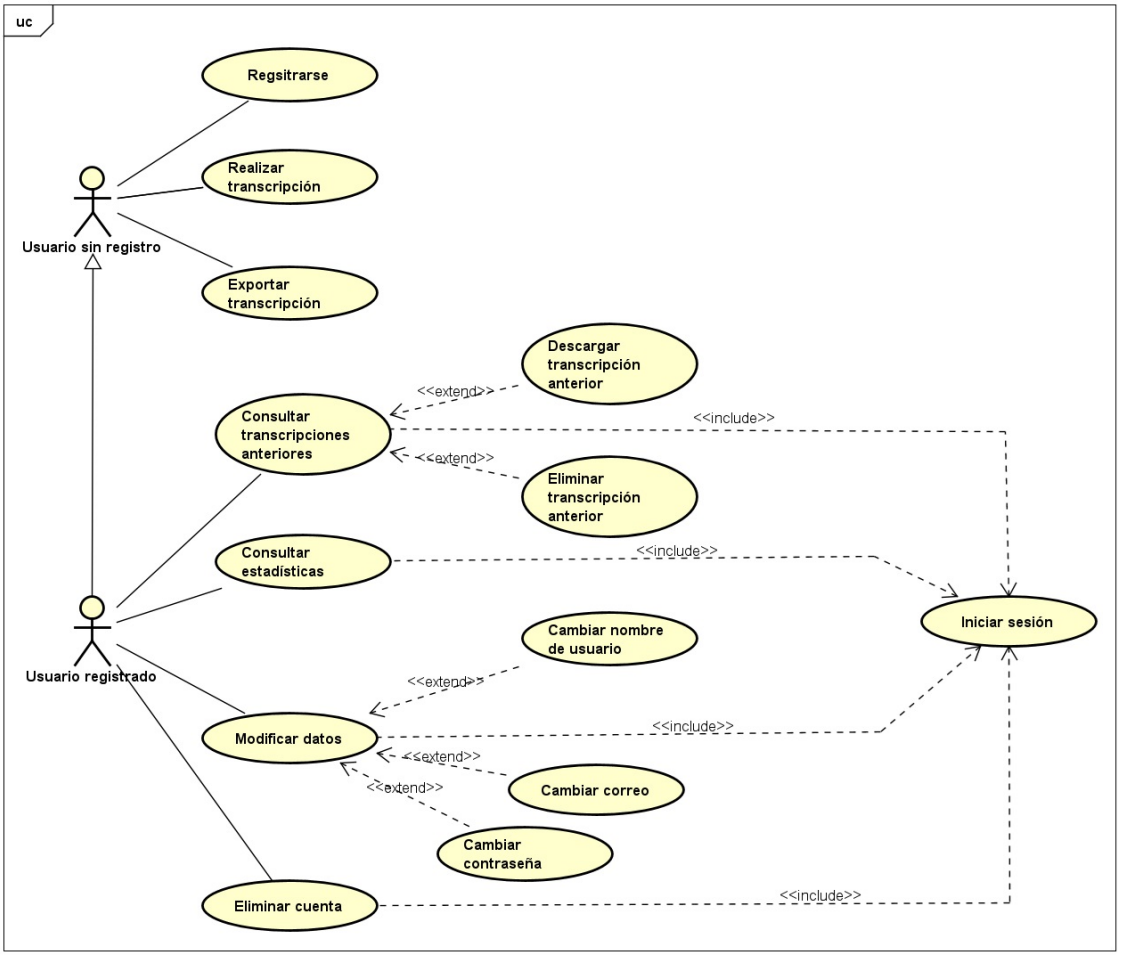

Figura 5: Diagrama de casos de uso

Cada una de estas operaciones tiene asociado un flujo de eventos necesario para llevarse a cabo. A continuación, se van a detallar los casos de uso de Realizar una Transcripción (figura 6) y Descargar una Transcripción Anterior (figura 7), para ejemplificar la función principal de la aplicación, y como ésta funciona en respuesta a las interacciones de un usuario.

ITEM	VALUE
UseCase	Realizar transcripción
Summary	
Actor	Usuario sin registro
Precondition	El usuario debe haber accedido a la aplicación.
Postcondition	1-Se realiza una transcripción
Base Sequence	1-El usuario entra a la aplicación. 2-El sitema muestra la pantalla para crear nuevas transcripciones. 3-El usuario arrastra o selecciona un archivo. 4-El sistema procesa el fichero y muestra la tabla de resultados.
Branch Sequence	
Exception Sequence	3.1-El usuario introduce un archivo con una extensión no permitida. Se vuelve al paso 2. 4.1-El sistema devuelve un error al transcribir el fichero. Se vuelve al paso 2.
Sub UseCase	

Figura 6: Caso de uso Realizar Transcripción

ITEM	VALUE
UseCase	Descargar transcripción anterior
Summary	El usuario registrado descarga una transcripción que ha realizado anteriormente.
Actor	
Precondition	1-El usuario ha iniciado sesión en la aplicación. 2-Este usuario ya ha realizado alguna transcripción.
Postcondition	1-El usuario obtiene la transcripción realizada.
Base Sequence	1-El usuario accede a la pestaña de "Mis Conversiones". 2-El sistema muestra una tabla con las transcripcioines anteriores del usuario. 3-El usuario pulsa el botón de descargar alguna de las transcripciones. 4-El sistema pide una confirmación. 5-El usuario selecciona "Si". 6-El sistema accede a la base de datos a consultar la transcripción. 7-El sistema accede al repositorio de ficheros y devuelve la transcripción.
Branch Sequence	5.1-El usuario selecciona "No". Se vuelve al paso 4.
Exception Sequence	2.1-El usuario no ha realizado transcripciones. Se vuelve al paso 2. 6.1- El sistema devuelve un error al consultar la base de datos. Se vuelve al paso 2. 7.1- El sistema devuelve un error al acceder al sistema de archivos. Se vuelve al paso 2.
Sub UseCase	

Figura 7: Caso de uso Descargar Transcripción Anterior

6.5. Diagramas de secuencia

Los diagramas de secuencia son una representación de la interacción de los objetos del sistema a lo largo del tiempo.

Estos diagramas están caracterizados por varios elementos principales:

- **Actores.** Son las entidades que interactúan con el sistema.
- **Mensajes.** Representan las interacciones entre los objetos del sistema.
- **Líneas de vida.** Representan la existencia de un objeto a lo largo de la secuencia de interacciones.
- **Activaciones.** Muestran el periodo de tiempo que está ocupado un objeto.

Al igual que con los casos de uso, existe una gran cantidad de diagramas de secuencia que describen el flujo de operaciones de las posibilidades que ofrece la aplicación, se van a ilustrar cuatro de ellos a modo de ejemplificar las operaciones.

El primero de estos diagramas (figura 8) representa el flujo de acciones de la aplicación cuando el usuario inicia sesión. En este caso, el usuario llama a la aplicación web introduciendo un nombre de usuario y contraseña. Estos datos son usados por la aplicación para realizar una consulta en la base de datos del sistema. En función del resultado de esta búsqueda, se pueden obtener dos resultados, que exista en la base de datos de los usuarios un registro coincidente, o que no. Este resultado de la búsqueda, es transmitido de nuevo por la aplicación web para informar al usuario de si la operación ha sido satisfactoria o no.

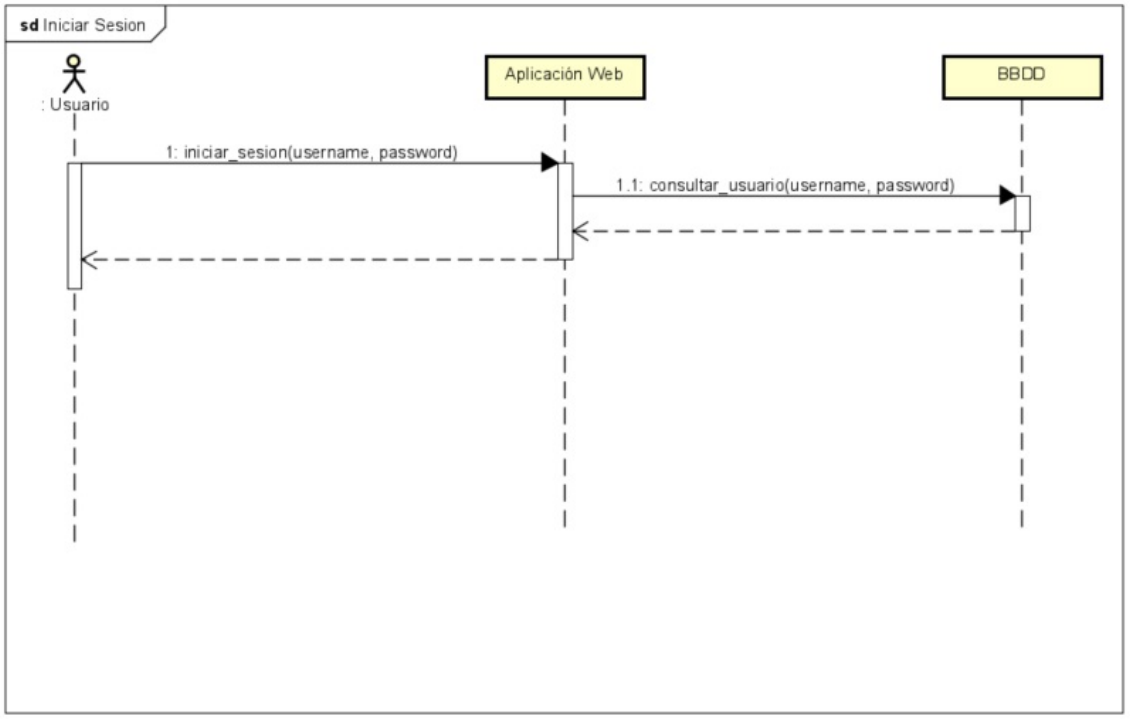

Figura 8: Diagrama de secuencia de Iniciar Sesión

En el segundo de los diagramas (figura 9), podemos observar el flujo de interacciones cuando el usuario realiza una nueva transcripción. En este proceso, el usuario (tanto registrado en la aplicación como no registrado), accede a la página para generar transcripciones. La aplicación reacciona a este mensaje mostrando la página encargada de recibir los datos para realizar transcripciones y espera a que el usuario adjunte el audio a transcribir. Una vez lo recibe, se encarga de transmitirlo a la API encargada de las transcripciones, que la realiza y devuelve la salida a a la aplicación para que pueda mostrársela al usuario.

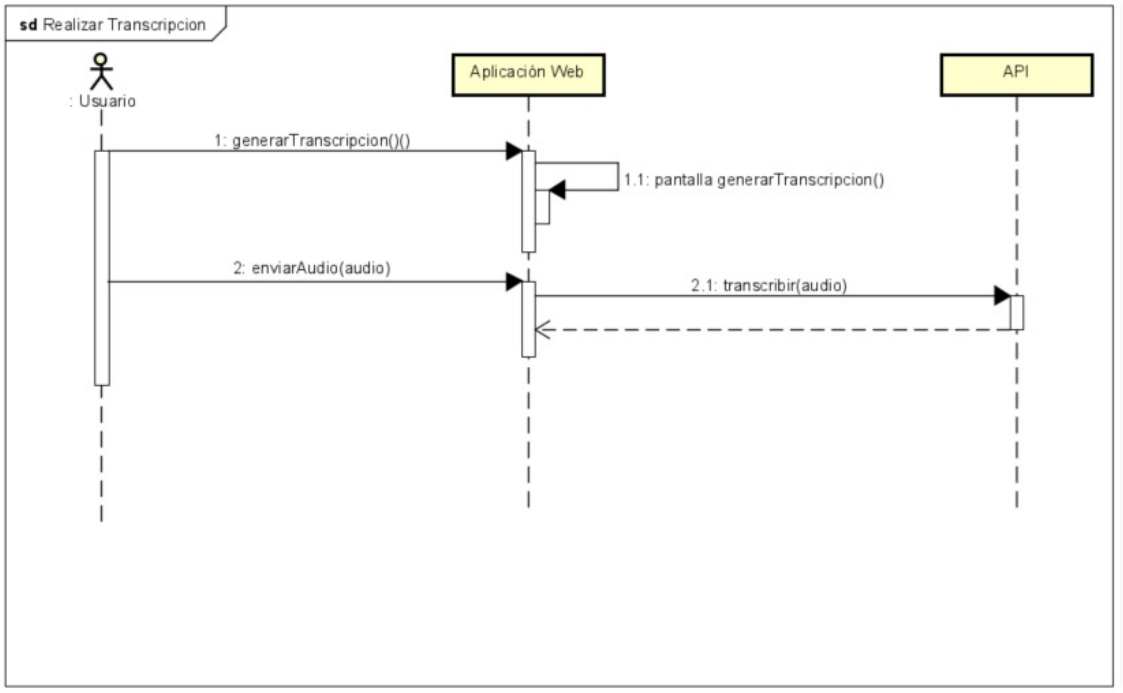

Figura 9: Diagrama de secuencia de Generar Transcripción

En el siguiente diagrama (figura 10), podemos observar la secuencia que sigue la aplicación cuando el usuario quiere eliminar una transcripción que había realizado anteriormente. En este caso, un usuario ya habiendo iniciado sesión en el sistema, accede a la pantalla para consultar las transcripciones que ha realizado. La aplicación, utilizando el identificador de ese usuario, realiza una consulta a la base de datos para obtener las transcripciones almacenadas para ese usuario, y muestra la pantalla que las contiene. A continuación, el usuario selecciona la que desea eliminar; esto hace que la aplicación vuelva a consultar la base de datos para comprobar la transcripción indicada, y utiliza estos datos para acceder al sistema de ficheros y eliminar la transcripción correspondiente. Una vez ha sido eliminada del sistema de archivos, realiza otra llamada a la base de datos para eliminar su referencia dentro de ésta y no mantener referencias a ficheros inexistentes.

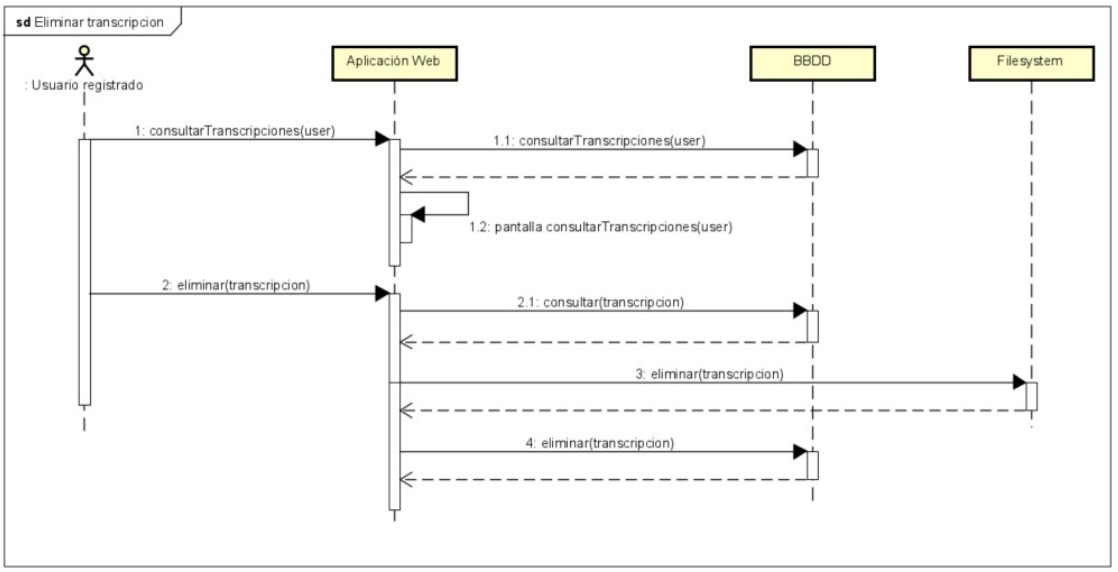

Figura 10: Diagrama de secuencia de Eliminar Transcripción

Por último, en el diagrama de la figura 11, podemos observar las operaciones que se realizan cuando el usuario quiere eliminar su cuenta y todos sus datos. En este caso, el usuario, ya habiendo accedido a la aplicación, pulsa el botón indicado para ello y la aplicación consulta en la base de datos los datos del usuario. Con estos datos, realiza otra consulta para obtener todas las transcripciones que ha realizado. Una vez tiene estos datos, accede al sistema de ficheros para eliminarlas. Por último, vuelve a realizar llamadas a la base de datos, para eliminar tanto los datos de las transcripciones, ya que estas no existen en el sistema de ficheros, como los datos del usuario. Finaliza las operaciones cerrando la sesión del usuario.

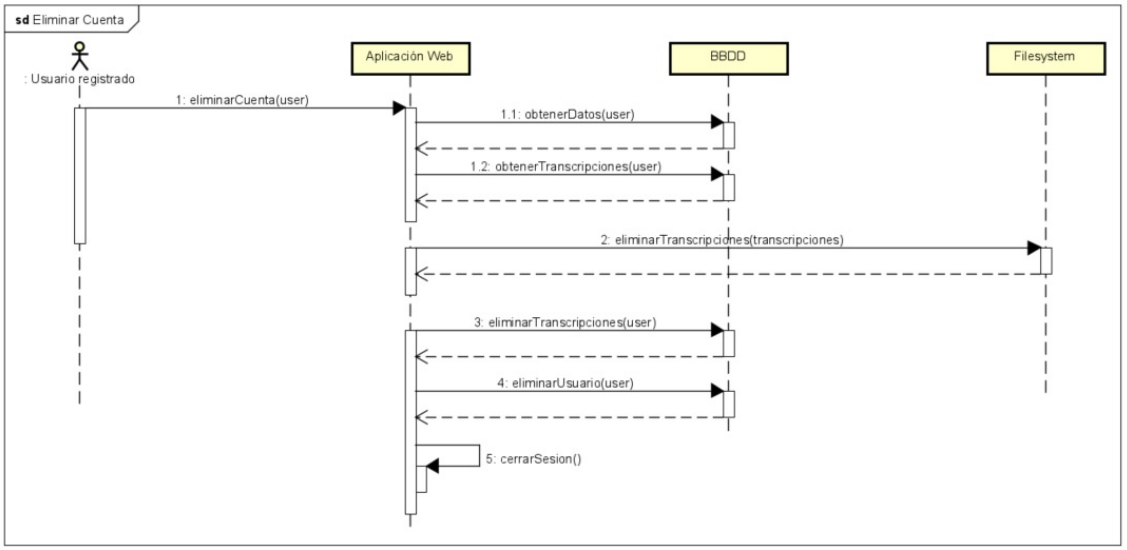

Figura 11: Diagrama de secuencia de Eliminar Cuenta

7 Diseño

Como se ha comentado en puntos anteriores, se ha diseñado la aplicación basándose en un desarrollo modular, facilitando la escalabilidad, la reutilización de módulos y la facilidad de comprensión y mantenimiento del código.

Además, se ha utilizado el patrón arquitectónico Modelo Vista Controlador, detallado en el apartado de las Tecnologías utilizadas.

7.1. Desarrollo basado en Componentes

El diseño basado en componentes [73][74], es un patrón de diseño orientado al desarrollo de un sistema a partir de la creación de componentes más pequeños reutilizables e independientes entre sí. Cada uno de los componentes contiene unas funcionalidades específicas de la aplicación, dotándolo de una independencia del resto de componentes que facilita tanto su implementación como su mantenimiento.

Dentro de las ventajas de este patrón, además de las ya comentadas de facilidad de mantenimiento e implementación del mismo, se encuentran un desarrollo más rápido y eficaz, un alto grado de escalabilidad y la posibilidad de la reutilización de dichos componentes.

A pesar de ello, también cuenta con ciertas desventajas; ya que añade complejidad a la hora de realizar pruebas sobre la aplicación, siendo necesaria la comprobación individual de cada uno de los componentes del sistema. Además, ligado a su capacidad de reutilización, tiene la desventaja de un menor grado de personalización por su posible uso en diferentes sistemas.

A continuación, se detallan los componentes que forman la aplicación.

7.1.1. Login

Se accede a este componente automáticamente al iniciar la aplicación, y está contenido en la ruta '/login'.

Permite al usuario iniciar sesión en la aplicación a través de un nombre de usuario y contraseña ya existentes en el sistema.

A través de él, en función de las operaciones realizadas, se puede acceder a los componentes de Home, Reset Password y Register.

En la tabla 7.1 se describen los endpoints de este componente, definidos bajo la etiqueta '/login':

Endpoint	Método	Parámetros	Respuesta	Descripción
/login/getUser	GET	username: String password: String	idUser: Long	Comprueba que en BBDD exista un registro con ese nombre de usuario y contraseña. En caso afirmativo, devuelve el identificador de usuario que almacena en el token de sesión.

Cuadro 7.1: Endpoints del componente Login

7.1.2. Reset Password

Para acceder a este componente, es necesario pulsar el enlace de "He olvidado mi contraseña"desde la pantalla de Login.

Permite al usuario modificar su contraseña para acceder a la aplicación, siempre y cuando introduzca un nombre de usuario y correo electrónico válidos y registrados en el sistema.

Desde este componente, únicamente se puede volver al componente del Login.

Los endpoints asociados a este componente son los recogidos en la tabla 7.2, agrupados bajo la etiqueta '/resetPassword'.

Endpoint	Método	Parámetros	Respuesta	Descripción
/resetPassword/checkUser	GET	username: String email:String	idUser: Long	Comprueba que el nombre de usuario y correo introducidos, existan en la base de datos para un mismo usuario. En caso afirmativo, devuelve el identificador del usuario.
/resetPassword/savePassword	POST	idUser: Long password: String		Almacena la nueva contraseña en la base de datos.

Cuadro 7.2: Endpoints del componente Reset Password

7.1.3. Register

Para acceder al componente de registro, es necesario, desde el componente del Login, pulsar el enlace para registrarse en la aplicación.

Este componente permite a un nuevo usuario registrarse en el sistema para poder acceder con una cuenta verificada y tener todas las opciones de la aplicación disponibles.

Una vez en este componente, solo se puede regresar al componente del Login.

Tiene asociados los endpoints que se indican en la tabla 7.3, agrupados en la etiqueta '/register'.

Endpoint	Método	Parámetros	Respuesta	Descripción
/register/checkUser	GET	username: String	boolean	Se llama automáticamente cuando se está escribiendo el nombre de usuario. Comprueba si ya existe en base de datos y devuelve un valor verdadero o falso.
/register/checkEmail	GET	email: String	boolean	Se llama automáticamente cuando se está escribiendo el correo electrónico de usuario. Comprueba si ya existe en base de datos y devuelve un valor verdadero o falso.
/register/saveUser	POST	username: String email: String password: String		Almacena en base de datos el nuevo usuario.

Cuadro 7.3: Endpoints del componente Register

7.1.4. Home

Es el componente principal de la aplicación, se accede a él únicamente desde el Login, pero cuenta con dos formas de acceso, iniciando sesión con un usuario registrado o iniciando sin registrarse. Ambas opciones permiten acceder a todas las funcionalidades de este componente, pero en caso de iniciar sin registro, el menú de la aplicación muestra menos opciones, sin permitir al usuario acceder al resto de componentes.

Permite al usuario, tanto si ha iniciado con o sin registro, generar nuevas transcripciones de ficheros de audio, visualizarlas y descargarlas.

Desde este componente, todos los tipos de usuario pueden volver al componente del Login, y los usuarios que han iniciado sesión, pueden acceder a los componentes de User Transcriptions, Profile y Data, además de otorgarles la funcionalidad para eliminar su cuenta de usuario y todos sus datos y transcripciones realizadas.

En la tabla 7.4 se recogen los endpoint correspondientes a este componente.

Endpoint	Método	Parámetros	Respuesta	Descripción
/home/deleteUser	POST	idUser: Long		Elimina la cuenta del usuario, sus datos y sus transcripciones realizadas.
/home/uploadAudio	POST	files: MultipartFile[]		Sube los archivos introducidos por el usuario a la ruta temporal del sistema.
/home/transcribe	GET	idUser: Long filename: String	Transcripcion: Object	Para cada uno de los ficheros subidos por el usuario, realiza su transcripción.
/home/seePDF	GET	idFile: Long	text: String	Recibe el identificador del archivo que se desea visualizar y devuelve al sistema el texto de dicho fichero.
/home/downloadFile	GET	idFile: Long	ResponseEntity <InputStreamResource>	Descarga el fichero seleccionado en la carpeta de descargas de la máquina.

Cuadro 7.4: Endpoints del componente Home

7.1.5. User Transcriptions

Este componente está accesible únicamente para los usuarios que han iniciado sesión en la aplicación. Se puede acceder a él desde el menú principal de la aplicación, disponible en los componentes Home, Profile, Data, y el propio componente.

Permite al usuario consultar las transcripciones que ha realizado anteriormente, así como visualizarlas, descargarlas o eliminarlas.

Tiene asociados los endpoints indicados en la tabla 7.5.

Endpoint	Método	Parámetros	Respuesta	Descripción
/userTranscriptions/seePDF	GET	idFile: Long	text: String	Recibe el identificador del archivo que se desea visualizar y devuelve al sistema el texto de dicho fichero.
/userTranscriptions/downloadFile	GET	idFile: Long	ResponseEntity <InputStreamResource>	Descarga el fichero seleccionado en la carpeta de descargas de la máquina.
/userTranscriptions/deleteFile	POST	idFile: Long		Elimina del sistema de archivos y de la base de datos el fichero indicado por el usuario.

Cuadro 7.5: Endpoints del componente User Transcriptions

7.1.6. Profile

Al igual que el componente anterior, únicamente pueden acceder al componente de Profile los usuarios que inicien sesión en el sistema.

Muestra al usuario una serie de estadísticas sobre su uso de la aplicación, como el número de transcripciones totales realizadas, una lista de los distintos idiomas que ha utilizado, el idioma más utilizado junto con su correspondiente número de usos, las fechas de la primera y la última transcripción realizada y una media de las transcripciones que realiza diariamente.

A pesar de ser únicamente de consulta y no necesitar de la interacción del usuario, tiene asociados varios endpoints que se ejecutan automáticamente al acceder al componente; además del correspondiente para eliminar la cuenta de usuario. Estos endpoints, identificados bajo la etiqueta '/profile', están descritos en la tabla 7.6.

Endpoint	Método	Parámetros	Respuesta	Descripción
/profile/getTranscriptions	GET	idUser: Long	List<Transcripciones>: List<Object>	Consulta en la base de datos las transcripciones realizadas por el usuario.
/profile/getLanguages	GET	idUser: Long	languageList: String	Consulta en base de datos la lista de idiomas distintos que ha utilizado el usuario.
/profile/getMaxLanguage	GET	idUser: Long	maxLanguage: String	Consulta en base de datos cuál es el idioma más utilizado por el usuario.
/profile/getMaxUse	GET	idUser: Long	uses: String	Consulta en base de datos cuál es el número de usos del idioma más utilizado por el usuario.
/profile/getFirstDate	GET	idUser: Long	firstDate: String	Consulta en base de datos cuál es la primera fecha de una transcripción realizada por el usuario.
/profile/getLastDate	GET	idUser: Long	lastDate: String	Consulta en base de datos cuál es la última fecha de una transcripción realizada por el usuario.
/profile/getAverage	GET	idUser: Long	average: String	Consulta en base de datos distintos valores y realiza los cálculos para obtener la media de transcripciones diarias realizadas.

Cuadro 7.6: Endpoints del componente Profile

7.1.7. Data

Este componente únicamente es accesible para los usuarios que hayan iniciado sesión en el sistema a través del menú de la aplicación.

Permite al usuario modificar sus datos, tanto nombre de usuario, como correo electrónico, siempre y cuando no coincidan con otros ya registrados en el sistema y el correo tenga un formato adecuado, y la contraseña, siempre que cumpla los estándares de seguridad.

Tiene asociados los endpoints que se recogen en la tabla 7.7, agrupados en la etiqueta '/data'.

Endpoint	Método	Parámetros	Respuesta	Descripción
/data/getData	GET	idUser: Long	List<Usuarios>: List<Object>	Se llama al acceder al componente y obtiene los datos actuales del usuario para mostrarlos.
/data/saveName	POST	idUser: Long username: String		Almacena en base de datos el nuevo nombre de usuario.
/data/saveEmail	POST	idUser: Long email: String		Almacena en base de datos el nuevo correo electrónico.
/data/savePassword	POST	idUser: Long password: String		Almacena en base de datos la nueva contraseña.

Cuadro 7.7: Endpoints del componente Data

7.2. Base de Datos

En este punto, vamos a centrarnos en conocer la base de datos desarrollada para el desarrollo de la aplicación. Realizada en MySQL Workbench, que utiliza el lenguaje de consulta estructurado SQL.

Para el correcto funcionamiento de la aplicación y el almacenamiento de datos, se han creado dos tablas, la tabla de usuarios y la tabla de transcripciones.

7.2.1. Usuarios

Es la tabla que almacena los datos correspondientes a los usuarios que se registran en el sistema. Esta compuesta por las columnas que se detallan en la tabla 7.8:

Columna	Tipo	Descripción	Otra información
ID	INT	Identificador del usuario	Es la clave primaria, por tanto, debe ser no nulo, único y se ha creado como valor autoincremental.
NAME	VARCHAR(45)	Nombre de usuario	
PASSWORD	VARCHAR(45)	Contraseña del usuario	
EMAIL	VARCHAR(45)	Correo electrónico del usuario	
CREATION_DATE	DATETIME	Fecha en la que fue creada la cuenta del usuario	

Cuadro 7.8: Tabla Usuarios

7.2.2. Transcripciones

Es la tabla que almacena la información sobre todas las transcripciones realizadas con la aplicación. Esta compuesta por las columnas que se detallan en la tabla 7.9:

Columna	Tipo	Descripción	Otra información
ID	INT	Identificador de la transcripción	Es la clave primaria de la tabla, por tanto, debe ser no nulo, único y se ha creado como valor autoincremental.
ID_USER	INT	Almacena la referencia al identificador del usuario que hizo la transcripción.	
DATE	DATETIME	Almacena la fecha en la que se realizó la transcripción.	
FILE	VARCHAR(45)	Contiene el nombre del fichero en el que se ha guardado la transcripción.	
LANGUAGE	VARCHAR(45)	Indica el idioma de la transcripción realizada.	
ORIGINAL_FILE	VARCHAR(45)	Contiene el nombre del fichero de audio original que se transcribió.	
TEXT	LONGTEXT	Contiene el texto extraído del fichero de audio.	

Cuadro 7.9: Tabla Transcripciones

8 Implementación

A lo largo de este punto, vamos a profundizar en los detalles de la implementación de la aplicación, a través de conocer la estructura del código del proyecto y en el resultado final desarrollado.

8.1. Implementación del Front-End

El front-end de la aplicación se ha desarrollado utilizando AngularTS y un diseño basado en componentes, por tanto, cada uno de los componentes de la aplicación, ha sido creado utilizando tres ficheros; un fichero HTML que define su estructura y diseño, un fichero TypeScript que define la lógica del componente, y un fichero CSS que aplica el estilo definido sobre la plantilla HTML.

A continuación, vamos a conocer cómo se han creado los ficheros HTML y TS de la aplicación. Omitiremos los ficheros CSS ya que su descripción no aporta un valor añadido para el conocimiento de la aplicación.

8.1.1. Implementación de componentes HTML

Los ficheros HTML son los encargados de la definición de las estructuras que se presentarán al usuario en cada uno de los componentes. En la imagen 12, podemos apreciar la estructura del código de uno de los componentes de la aplicación, en este caso, el componente de registro.

```
<div class="registro">
    <form>
        <div class="registro-form">
            <div class="registro-name">
                <label><b>Nombre de usuario</b><fa-icon class="icon" [icon]="faInfo" title="El nombre de usuario no debe coincidir con uno ya registrado"></fa-icon></label>
                <input type="text" [(ngModel)]="username" required [ngModelOptions]="{standalone: true}" (keyup)="comprobarUsername(username)">
                <span class="mensajeErrorRegistroNombre"> {{mensajeErrorRegistroNombre}} </span>
            </div>

            <div class="registro-correo">
                <label><b>Correo electrónico</b><fa-icon class="icon" [icon]="faInfo" title="El correo electrónico no puede estar en uso"></fa-icon></label>
                <input type="text" [(ngModel)]="email" required [ngModelOptions]="{standalone: true}" (keyup)="comprobarCorreo()">
                <span class="mensajeErrorRegistroCorreo"> {{mensajeErrorRegistroCorreo}} </span>
            </div>

            <div class="registro-password">
                <label><b>Contraseña</b><fa-icon class="icon" [icon]="faInfo" title="La constraseña usada debe tener al menos 8 caracteres, y contar, al menos, con
                    una mayúscula, una minúscula, un número y un símbolo"></fa-icon></label>
                <input type="password" [(ngModel)]="password" required [ngModelOptions]="{standalone: true}" (keyup)="comprobarPassword()">
            </div>
            <div class="registro-password">
                <label><b>Repita su contraseña</b></label>
                <input type="password" [(ngModel)]="password2" required [ngModelOptions]="{standalone: true}" (keyup)="comprobarPassword()">
                <span class="mensajeErrorPassword"> {{mensajeErrorPassword}} </span>
            </div>
            <div class="registro-button">
                <button class="boton-registro" type="submit" [disabled]="!registroHabilitado" (click)="registrar(username, password)">Registrarse</button>
            </div>
        </div>

    </form>
    <div class="mensajeErrorRegistro"><span > {{mensajeError}} </span></div>
    <div class="mensajeErrorRegistro"><span > {{mensajeErrorConexion}} </span></div>
    <div class="mensajeOK"><span> {{mensajeOK}} </span></div>
    <div class="boton-atras"><button type="submit" (click)="volver()">Volver</button></div>

</div>
<router-outlet></router-outlet>
```

Figura 12: Estructura del código HTML

Sin entrar en profundidad en todos los aspectos del diseño web, esta imagen nos sirve de ejemplo para conocer algunos aspectos generales seguidos en los componentes de la aplicación. En cada uno de ellos, se han elegido las estructuras más adecuadas para la funcionalidad del mismo, como puede ser la elección de tablas, formularios, distintos contenedores, etc.

Se ha optado por desarrollar los componentes utilizando un alto grado de encapsulación, agrupando cada uno de los elementos en distintos contenedores y clases con nombres descriptivos, para

favorecer la identificación y modificación de cada uno de ellos.

Otro de los aspectos destacables del diseño de los componentes HTML, ha sido la búsqueda de facilitar al usuario la utilización de cada uno de los componentes de la aplicación; para ello, se han introducido iconos descriptivos, como en este caso, el icono de información, que proporciona al usuario una guía de mensajes de lo que se espera que introduzca; el uso de abundantes mensajes de error para poder proporcionar más información acerca de los posibles fallos, o el habilitar o deshabilitar botones u opciones que requieren de otras acciones para poder ser utilizados.

8.1.2. Implementación de componentes TS

Los componentes TS son los encargados de añadir funcionalidad a los componentes HTML, a través del manejo de eventos o la implementación de funciones.

En la imagen 13, se puede conocer la estructura de uno de los componentes creados, en este ejemplo, el componente del Login.

```typescript
import { Component } from '@angular/core';
import { Router, RouterLink, RouterOutlet } from '@angular/router';
import { FormsModule } from '@angular/forms';
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { AppComponent } from '../app.component';


@Component({
  selector: 'app-login',
  standalone: true,
  imports: [
    FormsModule,
    HttpClientModule,
    RouterOutlet,
    RouterLink,
    AppComponent
  ],
  templateUrl: './login.component.html',
  styleUrl: './login.component.scss'
})
export class LoginComponent {

username: any;
password: any;
id: number = 0;
mensajeError = "";
mensajeErrorConexion = "";

private URL = "http://localhost:8080/login"

  constructor(private route: Router, private http: HttpClient) {}

  ngOnInit() {
    localStorage.clear();
  }

  login(username:string, password: string){

    const url = this.URL + "/getUser" + `?username=${username}&password=${password}`;

    this.http.get<number>(url)
    .subscribe(response => {
        if(response !== null && response!== undefined) {
          this.id = response;
          localStorage.setItem("username", this.username);
          localStorage.setItem("id", this.id.toString());
          this.route.navigateByUrl('/home');
        } else {
          this.mensajeError = "El nombre de usuario o la contraseña no son correctos"
        }
    }, error => {
      this.mensajeErrorConexion = "Se ha producido un error durante el proceso."
    });

  }
}
```

Figura 13: Estructura del código TS

En cuanto a decisiones importantes tomadas en el desarrollo de los componentes TypeScript, no hay nada reseñable que comentar, a excepción del diseño de los componentes de forma independiente, utilizando los standalone components que proporciona Angular.

8.2. Implementación del Back-End

La parte back-end de la aplicación ha sido desarrollada utilizando Spring Boot, y siguiendo una arquitectura de diseño en capas que divide las responsabilidades de cada una de las capas.

Estas capas son el controlador, encargado del manejo de las solicitudes HTTP; el servicio, encargado de la lógica y las operaciones; el repositorio, encargado de la interacción y las operaciones sobre la base de datos; y el modelo, que representa las tablas de la ´base de datos.

A continuación, vamos a detallar a través de varios ejemplos, cómo se han creado y desarrollado los siguientes componentes.

8.2.1. Implementación de Controller

Como se ha comentado, los 'Controller' son los encargados de la gestión de las solicitudes HTTP que se reciben en la aplicación, redireccionando las mismas hacia la capa encargada de su procesamiento y encargándose de realizar las respuestas.Son identificados con la anotación '@RestController'.

Otras anotaciones necesarias para su implementación son las siguientes:

- **@CrossOrigin**, que permite el intercambio de datos entre distintos dominios.
- **@Autowired**, que permite la inyección de dependencias.
- **@GetMapping**, se utiliza para manejar las solicitudes que esperan una devolución de datos.
- **@PostMapping**, se utiliza para manejar las solicitudes que envían datos.
- **@RequestParam**, utilizada para extraer los datos de la consulta.

En la imagen 14, podemos ver un ejemplo del código de un Controller, en este caso, el UserTranscriptionsController. En ella, podemos comprobar el uso de las anotaciones anteriormente mencionadas, y cómo la funcionalidad de estos archivos se basa en la recepción de las solicitudes HTTP, su gestión de las acciones a realizar y los parámetros recibidos, y su posterior envío hacia la capa de lógica de la aplicación.

```java
package com.TFG.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.TFG.model.Transcripciones;
import com.TFG.service.UserTranscriptionsService;

@RestController
@CrossOrigin(origins={"http://localhost:4200", "http://localhost:9876"})
public class UserTranscriptionsController {

    @Autowired
    private UserTranscriptionsService userTranscriptionsService;

    @GetMapping("/userTranscriptions/getTranscriptions")
    public List<Transcripciones> consultarTranscripciones(@RequestParam Long id) {
         return userTranscriptionsService.consultarTranscripciones(id);
    }

    @GetMapping("/userTranscriptions/seePDF")
    public String verPDF(@RequestParam Long idFile) {
        return userTranscriptionsService.verPDF(idFile);
    }

    @GetMapping("/userTranscriptions/downloadFile")
    public ResponseEntity<InputStreamResource> descargarFichero(@RequestParam Long idFile) {
         return userTranscriptionsService.descargarFichero(idFile);
    }

    @PostMapping("/userTranscriptions/deleteFile")
    public boolean borrarFichero(@RequestParam Long idFile) {
         return userTranscriptionsService.borrarFichero(idFile);
    }
```

Figura 14: Estructura del código del UserTranscriptionsController

8.2.2. Implementación de Service

Los elementos 'Service' son los encargados del manejo de la lógica de la aplicación, reciben los datos de los Controller y se encargan de realizar las acciones necesarias para devolver los datos al controlador, y están definidos por la anotación '@Service'

En las imágenes 15 y 16, podemos ver el código del HomeService, que contiene la lógica principal de la aplicación, y cómo se han desarrollado las distintas funciones, flujos de error y excepciones, cómo se realizan las transcripciones, y cómo interactúan estos elementos con los repositorios para acceder a la base de datos.

```java
ckage com.TFG.service;

port java.io.File;

ervice
blic class HomeService {

  @Autowired
  private HomeRepository homeRepository;

  @Autowired
  private HomeRepositoryTranscripciones homeRepositoryTranscripciones;

  public boolean borrarUsuario(Long id) {
      boolean correcto = false;
      List<Transcripciones> listatranscripciones = homeRepositoryTranscripciones.findAll();
      List<String> listaAudios = new ArrayList<String>();
      if(id!=null && id!=0) {
          try {
              //Borramos el usuario de BD
              homeRepository.deleteById(id);

              //Borramos sus transcripciones de BBDD
              for (int i = 0; i < listatranscripciones.size(); i++) {
                  if (listatranscripciones.get(i).getId_user().equals(id)) {
                      listaAudios.add(listatranscripciones.get(i).getFile());
                      homeRepositoryTranscripciones.deleteById(listatranscripciones.get(i).getId());
                  }
              }

              //Borramos sus audios del filesystem
              for (int j=0; j<listaAudios.size(); j++) {
                  String archivo = "C:\\Users\\samub\\TFG\\TFG\\audios\\resultados\\" + listaAudios.get(j) + ".pdf";
                  Path ruta = Paths.get(archivo);
                  Files.delete(ruta);
              }
              correcto = true;

          } catch (Exception e) {
              e.printStackTrace();
              return correcto;
          }
      }
      return correcto;
  }

  public boolean subirAudio(MultipartFile[] files) {
      boolean resultado = false;
      if (files.length !=0 ) {
          for (MultipartFile file : files) {
              if (!file.isEmpty()) {
                  try {
                      Path directorioTemporal = Paths.get(System.getProperty("java.io.tmpdir"));
                      Path archivoTemporal = directorioTemporal.resolve(file.getOriginalFilename());
                      Files.copy(file.getInputStream(), archivoTemporal);

                      resultado = true;
                  } catch (IOException e) {
                      e.printStackTrace();
                      resultado = false;
                  }
              } else {
                  resultado = false;
              }
          }
      }
      return resultado;
  }

  public Transcripciones transcribir(Long idUser, String filePath) throws IOException, DocumentException {

      // Almacenamos la transcripción y sus datos en BBDD
      Transcripciones nuevaTranscripcion = new Transcripciones();

      if (idUser!=null && filePath!="" && filePath!=null) {
          String nombrefinal = "";
          String idioma = "";

          String key = "sk-proj-tPrvFhz7iVuYAeZYgqKdT3BlbkFJxzplxXfBcjEFRuzLJsCo";
          OpenAiService service = new OpenAiService(key);

          // Mover el fichero a nuestra carpeta temporal
          String rutaFinal = moverATemp(filePath);

          // Realizamos la transcripción
          CreateTranscriptionRequest request = new CreateTranscriptionRequest();
          request.setModel("whisper-1");
          TranscriptionResult transcription = service.createTranscription(request, rutaFinal);
          String texto = transcription.getText();

          try {
              // Borramos de la carpeta temporal y creamos el PDF con el texto
              Files.delete(Paths.get(rutaFinal));
              nombrefinal = crearFichero(texto);
              idioma = obtenerIdioma(texto);

          } catch (IOException | DocumentException e) {
              e.printStackTrace();
          }

          Calendar c = Calendar.getInstance();
          Date date = c.getTime();

          nuevaTranscripcion.setId_user(idUser);
          nuevaTranscripcion.setDate(date);
          nuevaTranscripcion.setFile(nombrefinal);
          nuevaTranscripcion.setLanguage(idioma);
          nuevaTranscripcion.setOriginal_file(filePath);
          nuevaTranscripcion.setText(texto);
```

Figura 15: Estructura del código de HomeService I

```
        Calendar c = Calendar.getInstance();
        Date date = c.getTime();

        nuevaTranscripcion.setId_user(idUser);
        nuevaTranscripcion.setDate(date);
        nuevaTranscripcion.setFile(nombrefinal);
        nuevaTranscripcion.setLanguage(idioma);
        nuevaTranscripcion.setOriginal_file(filePath);
        nuevaTranscripcion.setText(texto);

        homeRepositoryTranscripciones.save(nuevaTranscripcion);

        return nuevaTranscripcion;
    } else {
        return null;
    }
}

public String moverATemp(String file) {

    String nombreArchivo = file;
    String carpetaDestino = "C:\\Users\\samub\\TFG\\TFG\\audios\\temp\\";
    String rutaDirectorioTemporal = System.getProperty("java.io.tmpdir");

    String rutaArchivoOrigen = rutaDirectorioTemporal + File.separator + nombreArchivo;
    String rutaCarpetaDestino = carpetaDestino + File.separator + nombreArchivo;

    try {
        // Movemos a nuestra carpeta y lo borramos de la anterior para no dejar archivos
        // residuales
        Files.copy(Paths.get(rutaArchivoOrigen), Paths.get(rutaCarpetaDestino));
        Files.delete(Paths.get(rutaArchivoOrigen));
    } catch (IOException e) {
        e.printStackTrace();
    }
    return rutaCarpetaDestino;
}

public String crearFichero(String texto) throws DocumentException {
    String nombre = "";
    try {
        Document document = new Document();

        Long id = homeRepositoryTranscripciones.count() + 1;
        nombre = "audio" + id;

        String destino = "C:\\Users\\samub\\TFG\\TFG\\audios\\resultados\\" + nombre + ".pdf";
        PdfWriter.getInstance(document, new FileOutputStream(destino));

        document.open();

        Phrase p = new Phrase(texto);
        document.add(p);

        document.close();

    } catch (FileNotFoundException | DocumentException ex) {
        ex.printStackTrace();
    }
    return nombre;
}

public String obtenerIdioma(String texto) {
    LanguageDetector detector = LanguageDetectorBuilder.fromAllLanguages().build();
    Language detectedLanguage = detector.detectLanguageOf(texto);
    return detectedLanguage.toString();
}
```

Figura 16: Estructura del código de HomeService II

```java
public String verPDF(Long idFile) {

    String res = "";
    if(idFile!=0 && idFile!=null) {
        Transcripciones transcripcion = homeRepositoryTranscripciones.getReferenceById(idFile);
        if (transcripcion != null) {
            res = transcripcion.getText();
        } else {
            res = "error";
        }
    }
    return res;
}
    public ResponseEntity<InputStreamResource> descargarFichero(Long id) {

    if(id!=null && id !=0) {
        Transcripciones transcripcion = homeRepositoryTranscripciones.getReferenceById(id);
        String fichero = "C:\\Users\\samub\\TFG\\TFG\\audios\\resultados\\" + transcripcion.getFile() + ".pdf";

        try {
            File file = new File(fichero);
            InputStreamResource resource = new InputStreamResource(new FileInputStream(file));

            HttpHeaders headers = new HttpHeaders();
            headers.add(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=" + file.getName());

            return ResponseEntity.ok()
                    .headers(headers)
                    .contentLength(file.length())
                    .contentType(MediaType.APPLICATION_PDF)
                    .body(resource);

        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Error al descargar el archivo: " + e.getMessage());
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }
    } else {
        return null;
    }
}
```

Figura 17: Estructura del código de HomeService III

8.2.3. Implementación de Repository

Los repositorios son los ficheros encargados de las interacciones con la base de datos de la aplicación, y de realizar las operaciones sobre la misma. Son definidos bajo la anotación @'Repository'.

Han sido creados utilizando JpaRepository, la interfaz de Spring Data JPA. Esta interfaz proporciona una serie de operaciones automatizadas sobre los repositorios, como save(), deleteById() o findAll(), que simplifican el diseño de las operaciones.

A pesar de ello, en ocasiones es necesario crear consultas más específicas para ajustarse a unas necesidades concretas, para ello,se ha utilizado la anotación '@Query', que permite realizar consultas en lenguaje SQL nativo o JPQL (Java Persistence Query Language).

En la imagen 18, podemos observar el código de la clase ProfileRepository como ejemplo para conocer la creación de estos elementos.

```
package com.TFG.repository;

import java.util.Date;

public interface ProfileRepository extends JpaRepository<Transcripciones, Long> {

    @Query("SELECT distinct t.idioma FROM Transcripciones t WHERE t.id_user = :id")
    List<String> getIdiomas(@Param("id") long id);

    @Query("SELECT t.idioma FROM Transcripciones t WHERE t.id_user = :id GROUP BY t.idioma ORDER BY COUNT(t.idioma) DESC")
    List<String> getMaxIdioma(@Param("id") long id);

    @Query("SELECT count(t.idioma) FROM Transcripciones t WHERE t.id_user = :id GROUP BY t.idioma ORDER BY COUNT(t.idioma) DESC")
    List<String> getMaxUsos(@Param("id") long id);

    @Query("SELECT t.fecha FROM Transcripciones t WHERE t.id_user = :id")
    List<Date> getFirstDate(@Param("id") long id);

    @Query("SELECT t.fecha FROM Transcripciones t WHERE t.id_user = :id")
    List<Date> getLastDate(@Param("id") long id);

}
```

Figura 18: Estructura del código de ProfileRepository

8.2.4. Implementación de Model

Las clases 'Model' son las encargadas de la representación de las tablas de la base de datos que forman la aplicación. Están definidas con la anotación '@Entity'.

Además de ésta, para su correcta creación es necesario el uso de otras anotaciones:

- **@Table**. Se utiliza para especificar el nombre de la tabla de la base de datos.
- **@Id**. Especifica qué atributo se utiliza como clave primaria de la tabla.
- **@GeneratedValue**. Se añade junto con la anotación '@Id' para especificar la forma de generación de la clave primaria.
- **@Column**. Especifica el mapeo de cada una de las columnas de la tabla.

A continuación se muestra una imagen (imagen 19) con el código de la clase 'Usuarios' para ejemplificar esta descripción de los ficheros de la capa Model. En ella podemos observar cómo se mapea la tabla 'Usuarios' de la base de datos a través del uso de las anotaciones comentadas, junto con el resto de métodos básicos como su constructor y los getters y setters de cada una de las variables.

```java
package com.TFG.model;

import java.util.Date;

@Entity
@Table(name="usuarios")
public class Usuarios {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nombre")
    private String nombre;

    @Column(name = "password")
    private String password;

    @Column(name = "correo")
    private String correo;

    @Column(name = "fecha_creacion")
    private Date fecha_creacion;

    public Usuarios(){

    }

    public Usuarios(Long id, String nombre, String password, String correo, Date fecha_creacion) {
        super();
        this.id = id;
        this.nombre = nombre;
        this.password = password;
        this.correo = correo;
        this.fecha_creacion = fecha_creacion;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }

    public Date getFecha_creacion() {
        return fecha_creacion;
    }

    public void setFecha_creacion(Date fecha_creacion) {
        this.fecha_creacion = fecha_creacion;
    }

}
```

Figura 19: Estructura del código de la clase Usuarios

8.2.5. Fichero pom.xml

El fichero pom.xml es la base de cualquier proyecto Maven, ya que es el archivo de configuración del proyecto. En él se especifican nombre del proyecto, versión, dependencias, plugins, etc.

A continuación se muestra una imagen (imagen imagen 20) de algunas las dependencias que se han añadido para el correcto desarrollo del proyecto y las funcionalidades de realización de transcripciones, identificación de idiomas y creación de ficheros PDF.

```
<!-- https://mvnrepository.com/artifact/com.theokanning.openai-gpt3-java/service -->
<dependency>
    <groupId>com.theokanning.openai-gpt3-java</groupId>
    <artifactId>service</artifactId>
    <version>0.18.2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.itextpdf/itextpdf -->
<dependency>
    <groupId>com.itextpdf</groupId>
    <artifactId>itextpdf</artifactId>
    <version>5.5.13.2</version>
</dependency>


<!-- https://mvnrepository.com/artifact/com.github.pemistahl/lingua -->
<dependency>
    <groupId>com.github.pemistahl</groupId>
    <artifactId>lingua</artifactId>
    <version>1.2.2</version>
</dependency>
```

Figura 20: Estructura del código del pom.xml

8.3. Interfaz

En este punto vamos a conocer las interfaces por las que está creada la aplicación. El objetivo del desarrollo del proyecto ha sido crear una página web que sea sencilla, con un diseño que resulte agradable al usuario, que sea muy intuitivo, que aporte ayudas al usuario y que cumpla las funcionalidades para las que ha sido pensada sin tratar de proporcionar otras funcionalidades que no aporten un valor extra al usuario, y le generen una sensación de desconocimiento de la aplicación.

En esta primera imagen (imagen 21) podemos observar la página del login de la aplicación. A través de él, el usuario puede iniciar sesión en la aplicación o iniciar sin registro, aunque accediendo a un menor número de opciones. Además, permite al usuario recuperar la contraseña o darse de alta en el sistema.

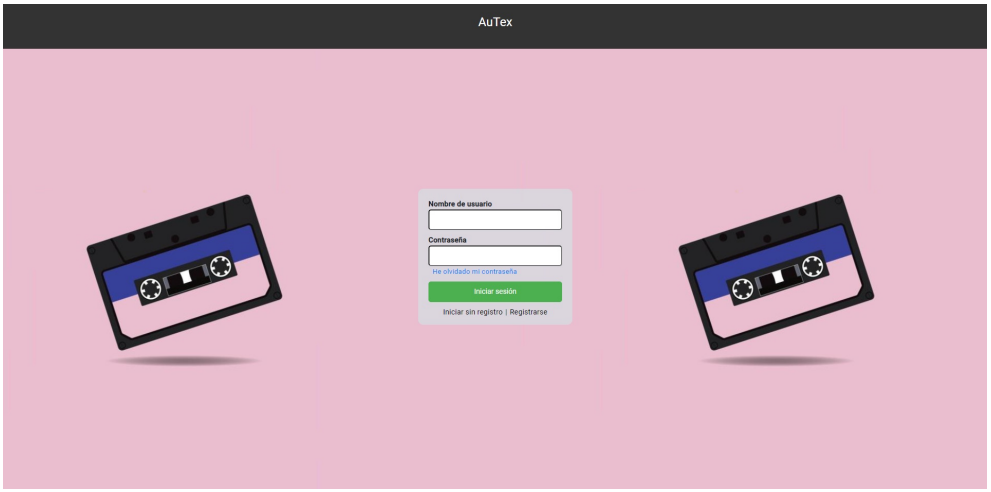

Figura 21: Interfaz del Login

En las siguientes imágenes (imagen 22) podemos observar la interfaz de usuario para registrarse en la aplicación. Observamos que se trata de un formulario básico que pide al usuario que ingrese un nombre, correo electrónico y una contraseña, que debe ser repetida dos veces para mayor seguridad.

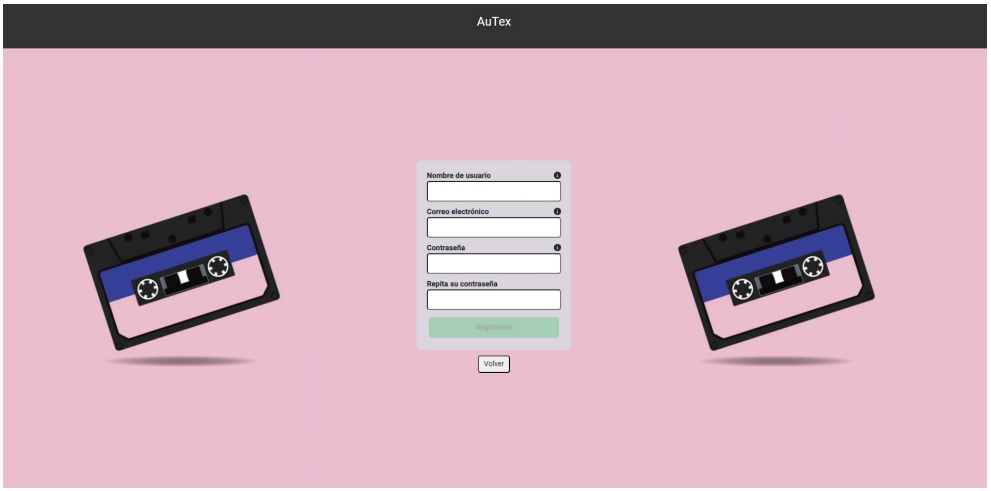

Figura 22: Interfaz del formulario de registro

Como podemos apreciar en las imágenes 23, 24, el formulario cuenta de ayudas para el usuario, de forma que pueda conocer por qué los datos no son válidos y no puede darse de alta, proporcionando ayudas a la usabilidad de la aplicación.

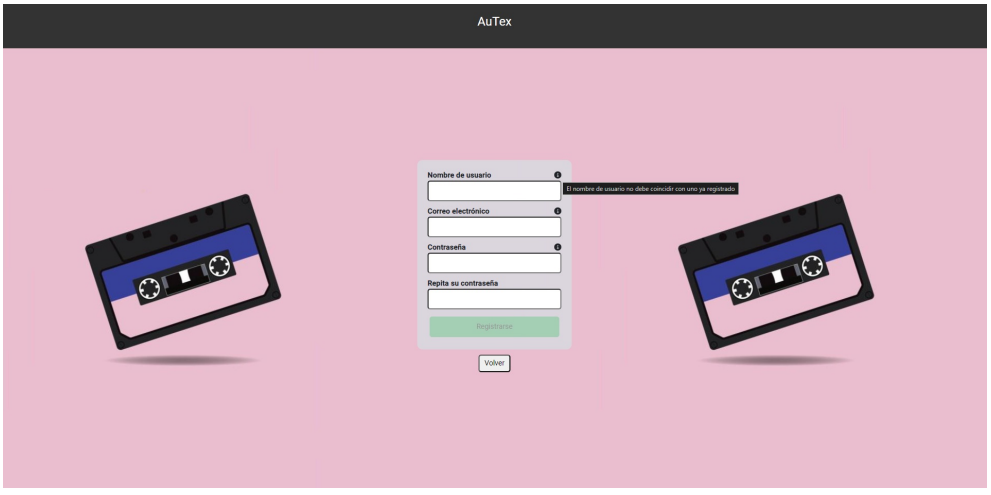

Figura 23: Ayudas al registro I

Figura 24: Ayudas al registro II

A continuación, también como parte de las ayudas proporcionadas al usuario, podemos observar en la imágenes 25 , 26 y 27 los mensajes de error que se muestran en caso de tratarse de datos erróneos, y cómo el botón permanece deshabilitado para evitar que el usuario trate de enviar datos que se le ha comunicado que son incorrectos.

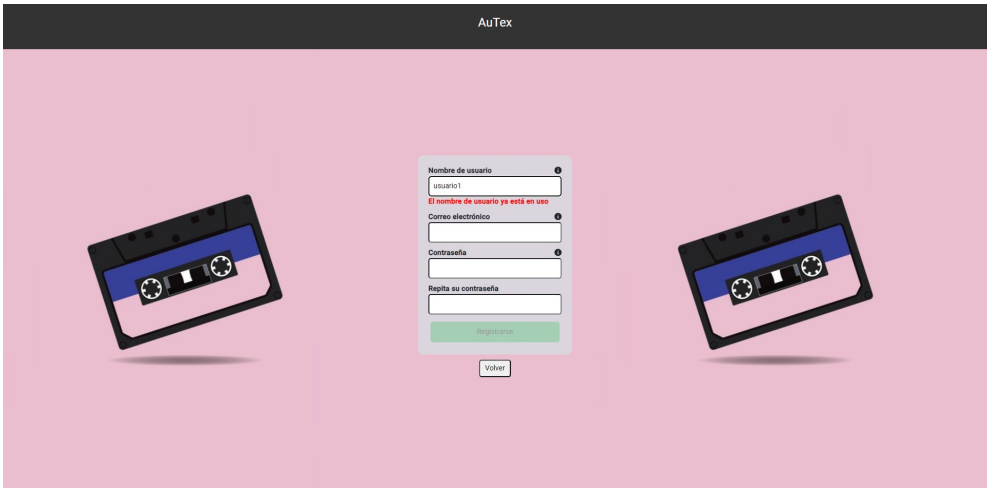

Figura 25: Ayudas al registro. Error en los datos

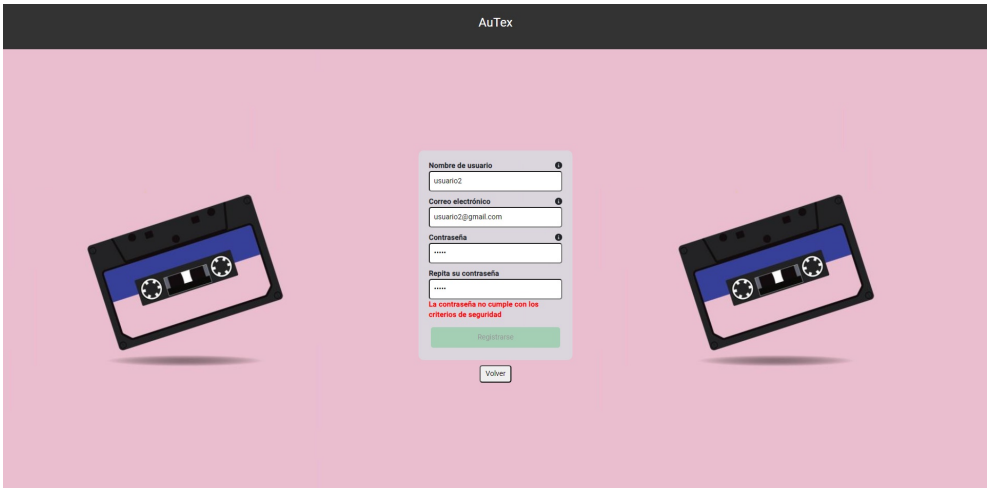

Figura 26: Ayudas al registro. Error en los datos II

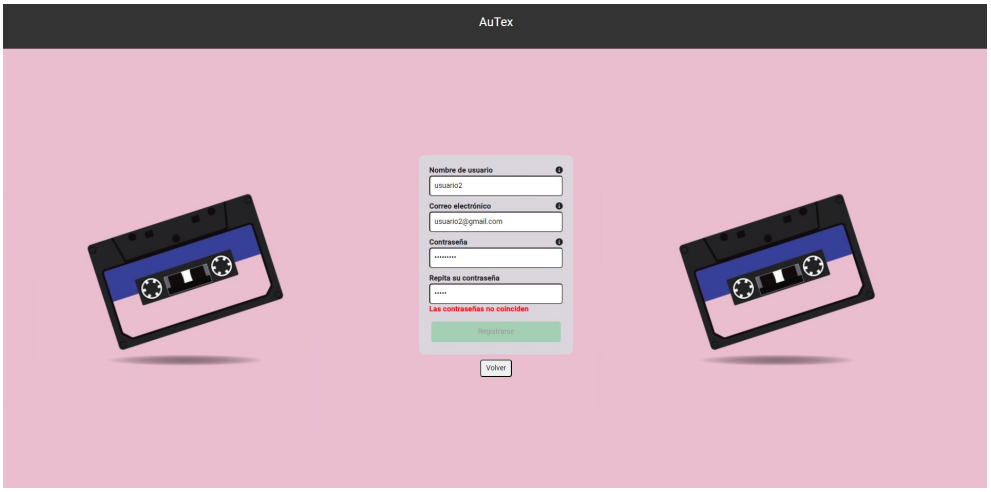

Figura 27: Ayudas al registro. Error en los datos III

Por último, en las imágenes 28, 29 podemos observar cómo el botón ya se ha habilitado dado que los datos son correctos, y cómo el usuario ha podido registrarse en el sistema y se le ha indicado que la operación ha sido satisfactoria.

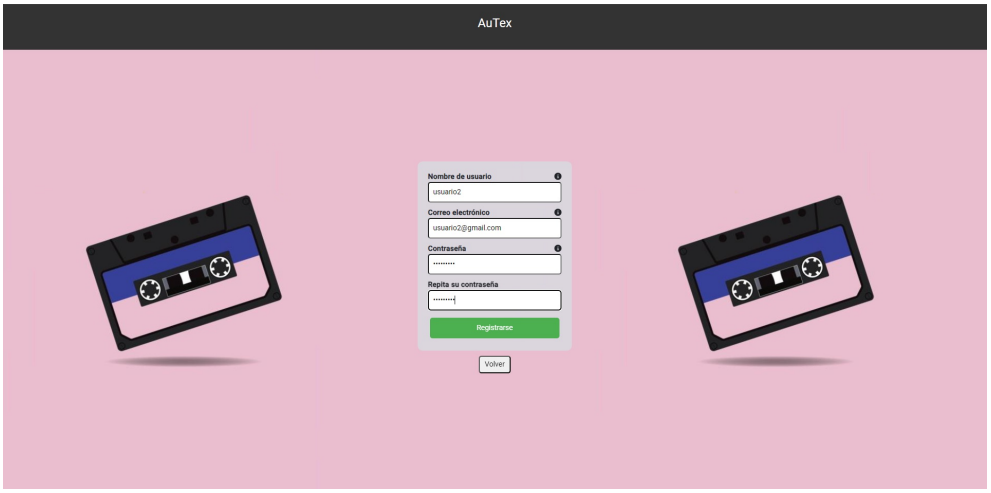

Figura 28: Formulario de registro correcto

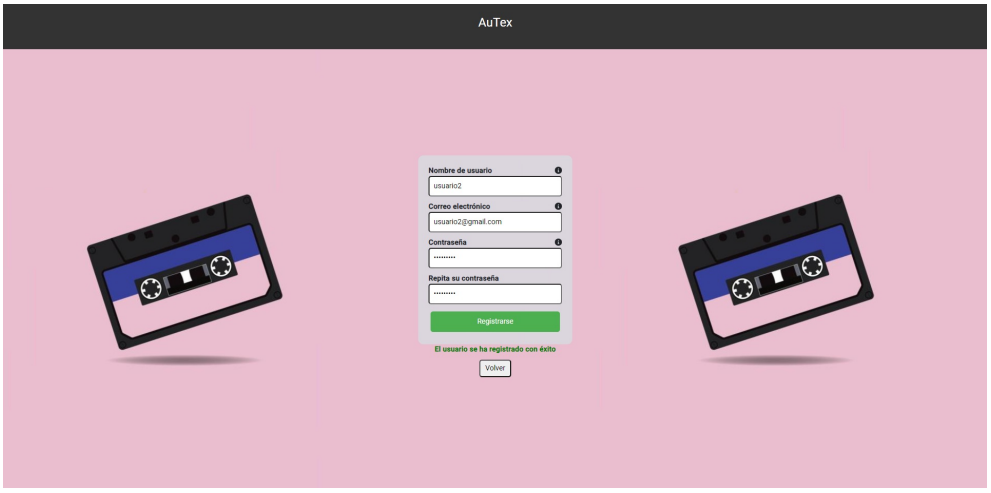

Figura 29: Confirmación del registro

En las imágenes 30, 31 y 32, podemos observar la interfaz que se le muestra al usuario cuando quiere recuperar la contraseña. Vemos como, en un primer paso, se le indica que introduzca un correo y un nombre de usuario, que, en caso de dejarse vacío o de introducir datos que no concuerden entre sí, no dejará avanzar a la segunda fase.

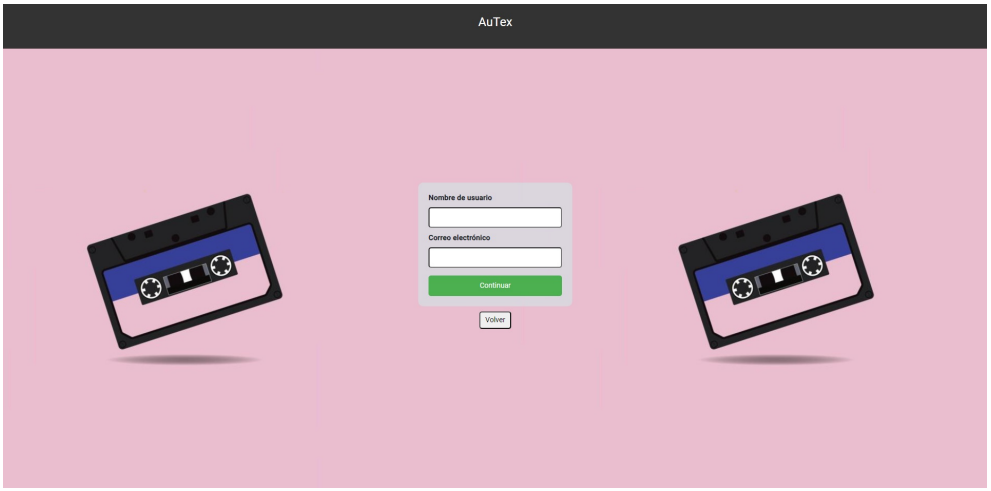

Figura 30: Interfaz para recuperar contraseña

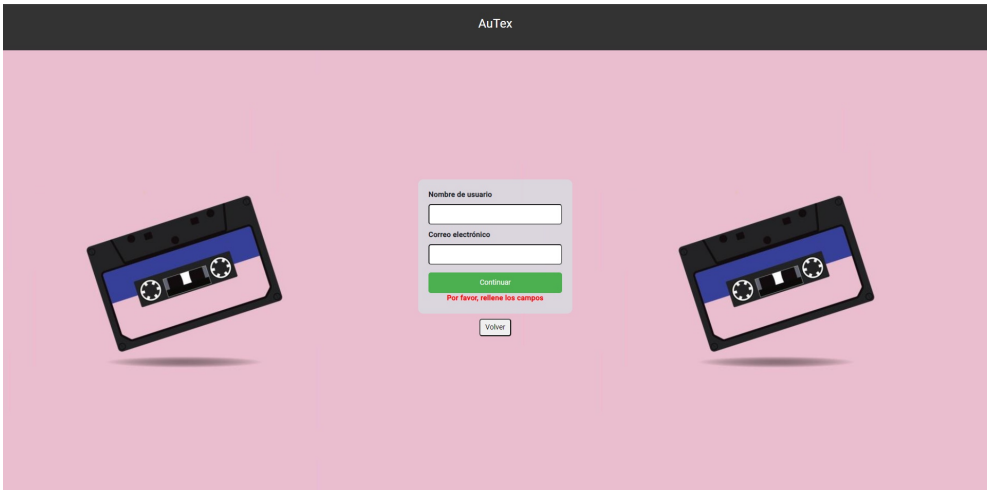

Figura 31: Recuperar contraseña. Error por campos vacíos

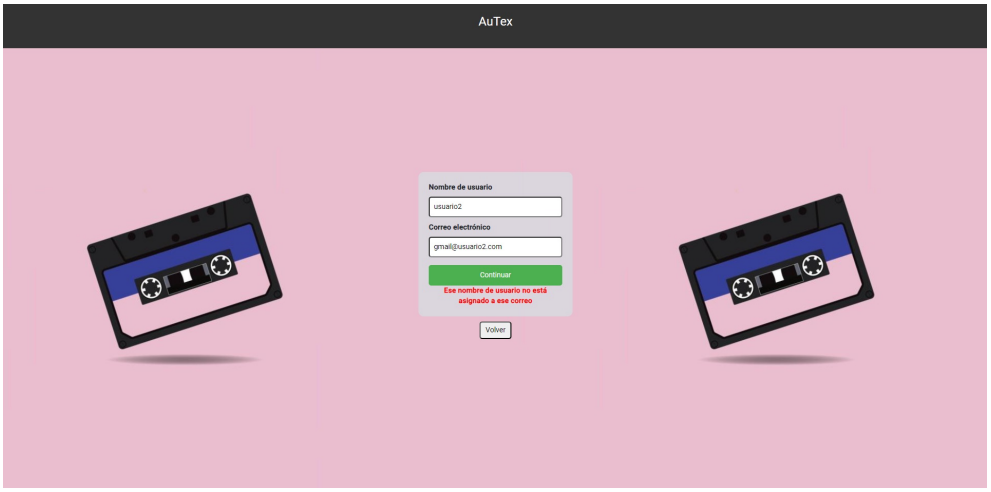

Figura 32: Recuperar contraseña. Error por campos incorrectos

Una vez se han introducido los datos correctamente, el usuario puede introducir una nueva contraseña que será almacenada en base de datos, como podemos observar en la imagen 33.

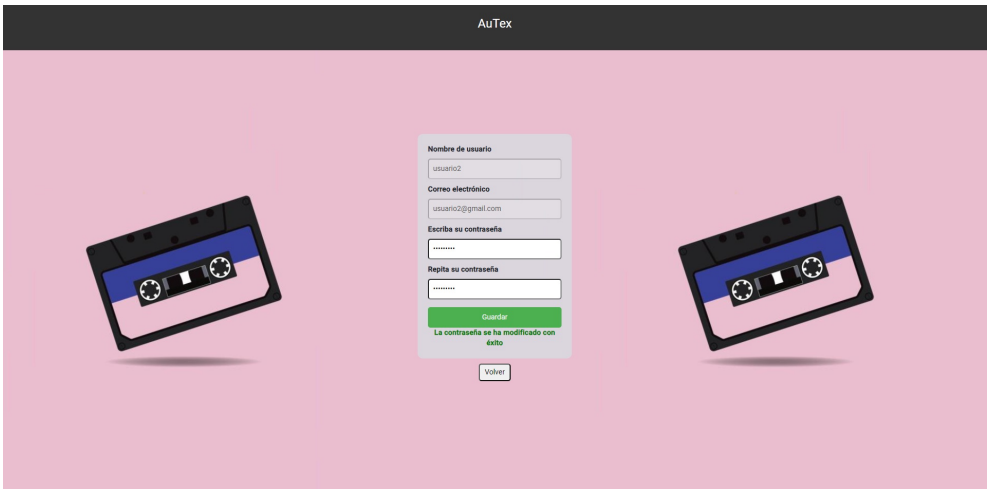

Figura 33: Recuperar contraseña correcto

En la imagen 34, podemos observar la interfaz de usuario para el caso de que se acceda sin iniciar sesión. En este caso únicamente podrá realizar transcripciones, y observamos que el menú superior solo contiene la opción para salir de la aplicación.

Figura 34: Página principal sin registro

Sin embargo, como vemos en la figura 35, si el usuario inicia sesión en el sistema, se le muestra la misma interfaz para realizar operaciones pero el menú superior ha habilitado el resto de opciones.

Figura 35: Página principal con registro

Una vez en este punto, el usuario puede subir varios archivos para transcribir, por lo que la interfaz se bloqueará mientras finaliza el proceso, y después mostrará los resultados en una tabla, permitiendo al usuario visualizarlos o descargarlos. Podemos observar el resultado en las imágenes 36 y 37.

Figura 36: Página principal. Transcripciones finalizadas

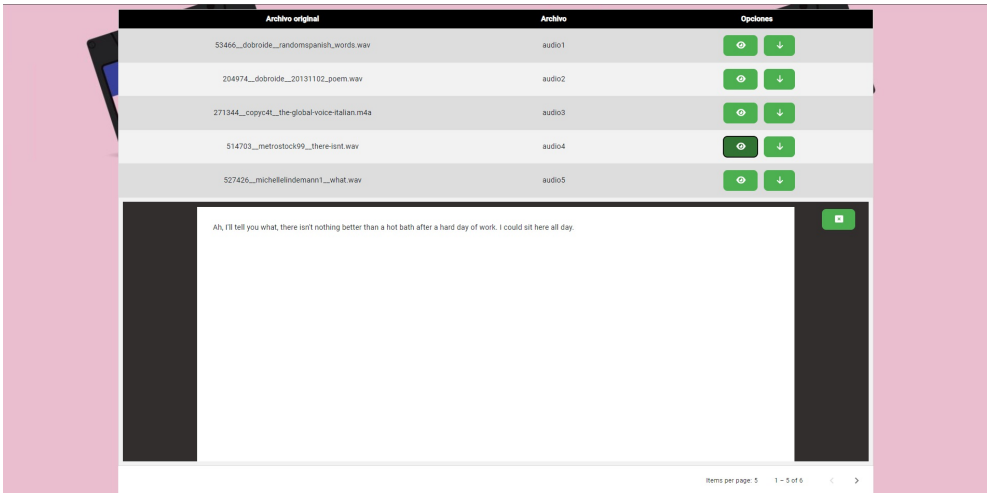

Figura 37: Página principal. Visualizar una transcripción

En la imagen 38 observamos la interfaz presentada al usuario para la consulta de las anteriores transcripciones realizadas. Esta interfaz muestra una tabla, con cinco registros por página, que aporta más datos sobre las transcripciones, y nos permite visualizarlas, descargar el fichero que contiene la transcripción y eliminarlas.

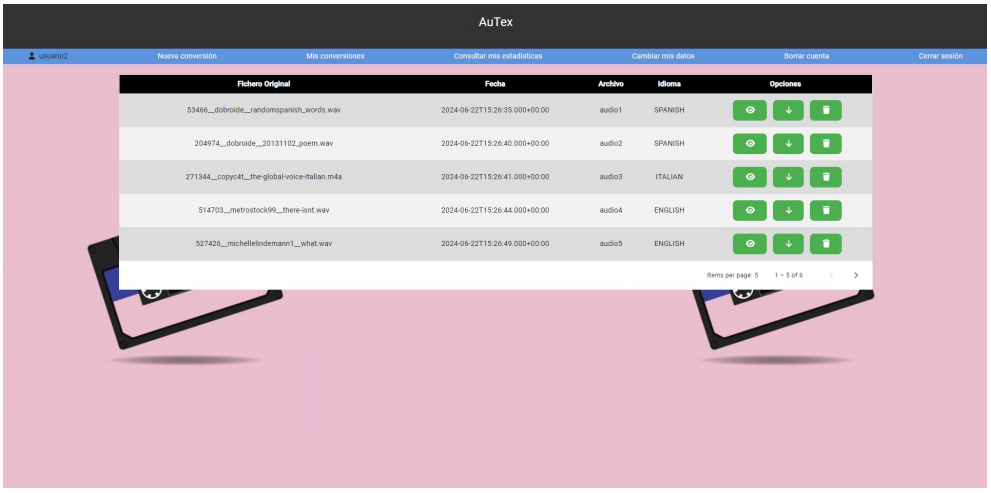

Figura 38: Interfaz visualizar transcripciones anteriores

En la imagen 39, se muestra la interfaz de las estadísticas del usuario. Esta interfaz es únicamente de consulta de datos, donde el usuario no puede realizar ninguna operación. Esta interfaz muestra estadísticas variadas como el número de transcripciones realizadas, el idioma más utilizado o una media de transcripciones diarias.

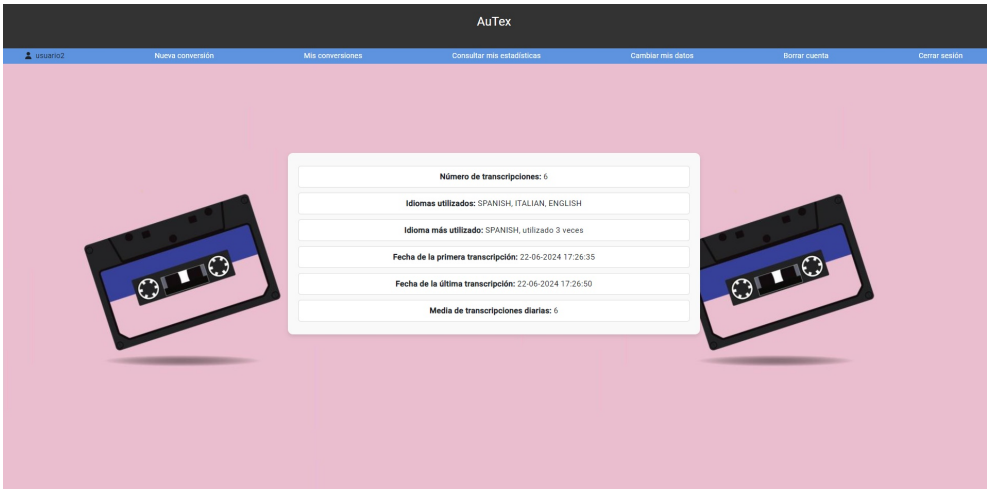

Figura 39: Interfaz de consulta de estadísticas del usuario

En las imágenes 40 y 41, podemos ver la interfaz diseñada para la modificación de datos del usuario. Los datos que se permite modificar son el nombre de usuario, el correo electrónico y la contraseña; y sigue las mismas restricciones que se solicitan para crear la cuenta, que el nombre y correo electrónico no estén en uso, que el correo tenga una estructura determinada y que la contraseña cumpla los parámetros de seguridad.

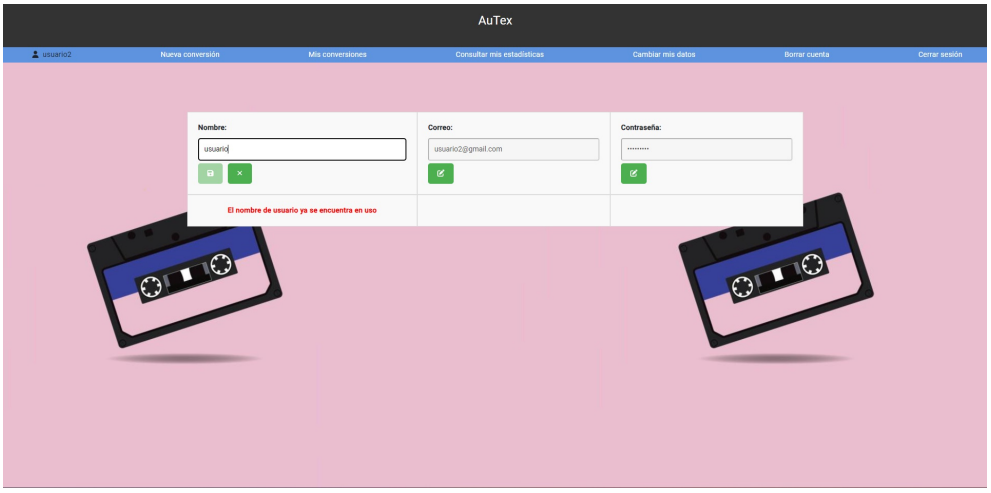

Figura 40: Interfaz de modificación de datos del usuario con error

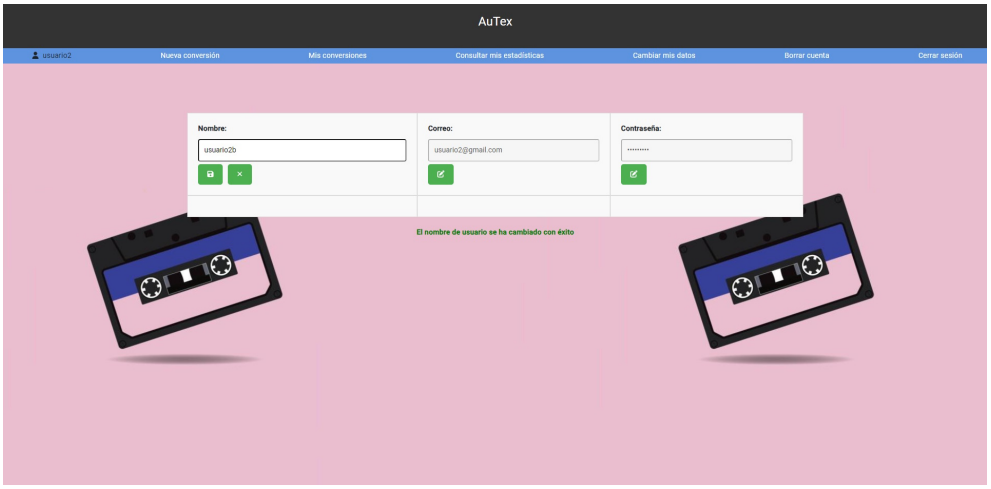

Figura 41: Interfaz de consulta de estadísticas del usuario

Por último, en la imagen 42, podemos observar la interfaz cuando seleccionamos la opción de borrar la cuenta. Esto nos abre un modal que ocupa toda la página, evitando tanto que se realice otra opción sin confirmar o modificar, como que la cuenta se borre por error. En caso de que decidamos borrarla, la aplicación redirecciona al usuario de nuevo a la ventana para iniciar sesión en la aplicación.

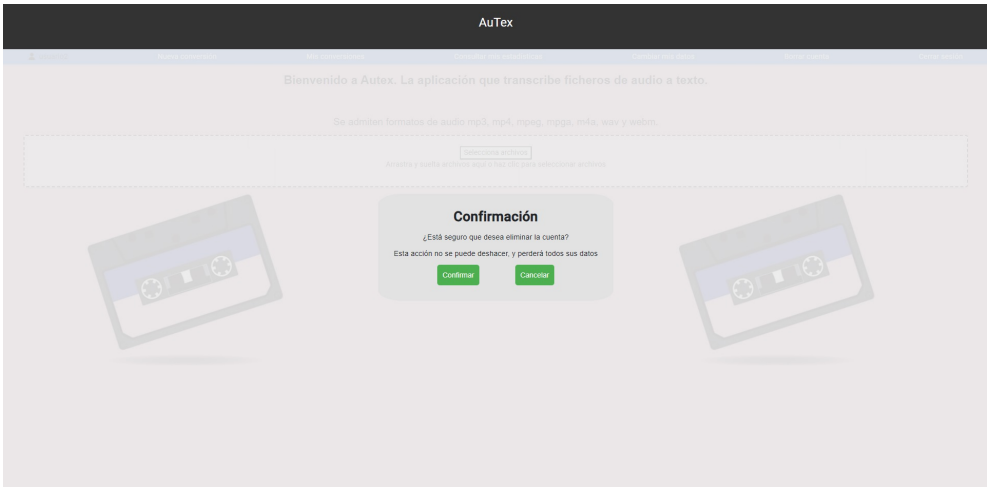

Figura 42: Modal de confirmación para eliminar la cuenta

9 Pruebas

El objetivo del desarrollo de unas pruebas completas y eficientes es asegurar el correcto funcionamiento de todos los componentes de la aplicación interaccionando entre sí.

Si bien durante el desarrollo se han ido realizando pruebas de funcionalidad, detalladas en los sprints, es necesario definir un conjunto de pruebas añadido encargado de probar en profundidad todas las opciones de la aplicación.

9.1. Pruebas manuales

El plan de pruebas desarrollado para el producto final ha sido desarrollado sin tener en cuenta el conocimiento de cómo está diseñado el código del programa. Una práctica común en los diseños y ejecución de baterías de pruebas, es que sean distintas personas las encargadas del desarrollo del sistema y de la realización de sus pruebas. Al ser un proyecto individual, no se puede lograr esta separación de roles, pero a pesar de ello es importante diseñar el plan de pruebas orientado a esta posibilidad.

Por lo tanto, esta batería de pruebas ha sido diseñada orientada únicamente a la interacción con la interfaz de usuario, añadiendo los pasos a realizar, los resultados esperados y los resultados obtenidos, de forma que pueda ser efectuado por cualquier persona ajena al desarrollo.

Pruebas sobre el Login			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID01	Sin introducir datos, pulsamos en "Iniciar sesión"	Se muestra un mensaje indicando que los datos no son correctos	Correcto
ID02	Introducimos datos erróneos y pulsamos en "Iniciar sesión"	Se muestra un mensaje indicando que los datos no son correctos	Correcto
ID03	Introducimos datos correctos y pulsamos en "Iniciar sesión"	Redirige a la página principal y el menú superior contiene todas las opciones	Correcto
ID04	Pulsamos en "He olvidado mi contraseña"	Redirige a la página que nos permite modificar la contraseña	Correcto
ID05	Pulsamos en "Iniciar sin registro"	Redirige a la página principal y el menú superior únicamente contiene la opción para salir	Correcto
ID06	Pulsamos en "Registrarse"	Redirige a la página de registro	Correcto

Cuadro 9.1: Pruebas sobre el Login

Pruebas sobre el formulario para recuperar contraseña			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID07	Sin introducir datos, pulsamos en "Continuar"	Se muestra un mensaje indicando que se rellenen los datos	Correcto
ID08	Introduciendo datos incorrectos, pulsamos en "Continuar"	Se muestra un mensaje indicando que los datos no son correctos	Correcto
ID09	Introduciendo datos correctos, pulsamos en "Continuar"	Se muestran dos nuevos input que permiten escribir la contraseña	Correcto
ID10	Introduciendo datos correctos, pulsamos en "Continuar" y escribimos una contraseña que no cumpla los criterios de seguridad	Se muestran un mensaje de error indicando que no es válida y no se habilita el botón	Correcto
ID11	Introduciendo datos correctos, pulsamos en "Continuar" y escribimos una contraseña válida	Se habilita el botón	Correcto
ID12	Introduciendo datos correctos, pulsamos en "Continuar", escribimos una contraseña válida y pulsamos en "Guardar"	Se muestra un mensaje indicando que se ha cambiado correctamente	Correcto
ID13	Pulsamos el botón de "Volver"	Volvemos a la pantalla de login	Correcto
ID14	Intentamos iniciar sesión con la contraseña que hemos introducido en la parte de la modificación	Se inicia sesión en la aplicación	Correcto

Cuadro 9.2: Pruebas sobre el formulario para recuperar contraseña

Pruebas sobre el formulario de registro			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID15	Pasamos el ratón por encima de los iconos de información	Muestra mensajes con información	Correcto
ID16	Introducimos un nombre de usuario en uso	Muestra un mensaje indicando que no es válido y no se habilita el botón	Correcto
ID17	Introducimos un nombre de usuario correcto y un correo con un formato no válido	Muestra un mensaje indicando que no es válido y no se habilita el botón	Correcto
ID18	Introducimos un nombre de usuario correcto y un correo en uso	Muestra un mensaje indicando que está en uso y no se habilita el botón	Correcto
ID19	Introducimos un nombre de usuario y correo correctos y una contraseña que no cumpla los criterios de seguridad	Muestra un mensaje indicando que no es válida y no se habilita el botón	Correcto
ID20	Introducimos un nombre de usuario y correo correctos y dos contraseñas distintas	Muestra un mensaje indicando que no coinciden y no se habilita el botón	Correcto
ID21	Introducimos un nombre de usuario, correo y contraseña correctos	Se habilita el botón	Correcto
ID22	Introducimos un nombre de usuario, correo y contraseña correctos, y pulsamos en "Registrarse"	Aparece un mensaje de éxito	Correcto
ID23	Pulsamos el botón de "Volver"	Volvemos a la pantalla de login	Correcto
ID24	Intentamos iniciar sesión con el nuevo usuario creado	Se inicia sesión en la aplicación	Correcto

Cuadro 9.3: Pruebas sobre el formulario de registro

Pruebas sobre la página principal sin haber iniciado sesión			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID25	Introducimos un fichero con una extensión no permitida	Muestra un mensaje de error	Correcto
ID26	Introducimos un fichero correcto	Realiza la transcripción y muestra una tabla con el resultado	Correcto
ID27	Pulsamos el botón de ver de la tabla	Despliega el resultado de la transcripción	Correcto
ID28	Pulsamos el botón de cerrar	Cierra el resultado de la transcripción	Correcto
ID29	Pulsamos el botón de descargar de la tabla	Muestra un mensaje de éxito y guarda el fichero en la carpeta de descargas del dispositivo	Correcto
ID30	Introducimos al menos 6 ficheros correctos y en la tabla de resultados, pasamos a la segunda página	Muestra la segunda página	Correcto
ID31	En la tabla de resultados, pasamos a la primera página	Muestra la primera página	Correcto
ID32	Pulsamos la opción de "Salir"del menú superior	Redirige a la pantalla del login	Correcto

Cuadro 9.4: Pruebas sobre la página principal sin haber iniciado sesión

Pruebas sobre la página principal I			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID33	Introducimos un fichero con una extensión no permitida	Muestra un mensaje de error	Correcto
ID34	Introducimos un fichero correcto	Realiza la transcripción y muestra una tabla con el resultado	Correcto
ID35	Pulsamos el botón de ver de la tabla	Despliega el resultado de la transcripción	Correcto
ID36	Pulsamos el botón de cerrar	Cierra el resultado de la transcripción	Correcto
ID37	Pulsamos el botón de descargar de la tabla	Muestra un mensaje de éxito y guarda el fichero en la carpeta de descargas del dispositivo	Correcto
ID38	Introducimos al menos 6 ficheros correctos y en la tabla de resultados, pasamos a la segunda página	Muestra la segunda página	Correcto
ID39	En la tabla de resultados, pasamos a la primera página	Muestra la primera página	Correcto
ID40	Pulsamos la opción de "Mis conversiones"del menú superior	Redirige a la pantalla de las conversiones realizadas	Correcto
ID41	Pulsamos la opción de "Consultar mis estadísticas"del menú superior	Redirige a la pantalla de las estadísticas del usuario	Correcto
ID42	Pulsamos la opción de "Cambiar mis datos"del menú superior	Redirige a la pantalla para modificar los datos del usuario	Correcto
ID43	Pulsamos la opción de "Cerrar sesión"del menú superior	Redirige a la pantalla del login y elimina los datos del Local Storage	Correcto
ID44	Iniciamos sesión con un usuario y pulsamos la opción de "Borrar cuenta", cuando aparezca el modal, pulsamos en "Cancelar"	Se cierra el modal sin realizar más operaciones	Correcto
ID45	Iniciamos sesión con un usuario y pulsamos la opción de "Borrar cuenta", cuando aparezca el modal, pulsamos en "Cancelar"	Se cierra el modal sin realizar más operaciones	Correcto
ID46	Iniciamos sesión con un usuario y pulsamos la opción de "Borrar cuenta", cuando aparezca el modal, pulsamos en "Confirmar"	El modal nos indica que se ha borrado la cuenta y nos redirige a la pantalla del login	Correcto

Pruebas sobre la página principal II			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID47	Intentamos iniciar sesión con el usuario que hemos eliminado	No se puede iniciar sesión porque no existe la cuenta	Correcto

Cuadro 9.5: Pruebas sobre la página principal

Pruebas sobre la página de conversiones realizadas			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID48	Sobre la tabla, pulsamos el botón para visualizar un archivo	Muestra el texto del fichero	Correcto
ID49	Pulsamos el botón para cerrar el texto	Cierra el texto del fichero	Correcto
ID50	Sobre la tabla, pulsamos el botón para descargar un archivo	Indica que el fichero ha sido descargado y aparece en la carpeta de descargas del dispositivo	Correcto
ID51	Sobre la tabla, pulsamos el botón para eliminar un archivo y, en el modal que aparece, pulsamos Çancelar"	Desaparece el modal y no se borra el archivo	Correcto
ID52	Sobre la tabla, pulsamos el botón para eliminar un archivo y, en el modal que aparece, pulsamos "Confirmar"	Nos indica que el fichero ha sido eliminado y se elimina también de la tabla	Correcto
ID53	Asegurándonos de que existen al menos 6 ficheros, pulsamos el botón para pasar a la segunda página de la tabla	Muestra la segunda página	Correcto
ID54	Pulsamos el botón para regresar a la primera página	Muestra la primera página	Correcto

Cuadro 9.6: Pruebas sobre la página de conversiones realizadas

Pruebas sobre la página de estadísticas de usuario			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID55	Accedemos a la página	Muestra todas las estadísticas	Correcto

Cuadro 9.7: Pruebas sobre la página de estadísticas de usuario

Pruebas sobre la página de modificación de datos de usuario			
ID	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID56	Pulsamos el botón para editar el nombre de usuario	Se habilita el cuadro de texto	Correcto
ID57	Pulsamos el botón de cancelar	Se deshabilita el cuadro de texto	Correcto
ID58	Introducimos un nombre de usuario que esté en uso	Se muestra un mensaje indicando que no es válido	Correcto
ID59	Introducimos un nombre de usuario correcto	Se habilita el botón de guardar	Correcto
ID60	Pulsamos el botón de guardar	Se muestra un mensaje indicando que se ha modificado correctamente	Correcto
ID61	Pulsamos el botón para editar el correo electrónico	Se habilita el cuadro de texto	Correcto
ID62	Pulsamos el botón de cancelar	Se deshabilita el cuadro de texto	Correcto
ID63	Introducimos un correo electrónico con un formato no permitido	Se muestra un mensaje indicando que no es válido	Correcto
ID64	Introducimos un correo electrónico que esté en uso	Se muestra un mensaje indicando que no es válido	Correcto
ID65	Introducimos un nombre de correo electrónico correcto	Se habilita el botón de guardar	Correcto
ID66	Pulsamos el botón de guardar	Se muestra un mensaje indicando que se ha modificado correctamente	Correcto
ID67	Pulsamos el botón para editar la contraseña	Se habilita el cuadro de texto	Correcto
ID68	Pulsamos el botón de cancelar	Se deshabilita el cuadro de texto	Correcto
ID69	Introducimos una contraseña que no cumpla los parámetros de seguridad	Se muestra un mensaje indicando que no es válida	Correcto
ID70	Introducimos una contraseña correcta	Se habilita el botón de guardar	Correcto
ID71	Pulsamos el botón de guardar	Se muestra un mensaje indicando que se ha modificado correctamente	Correcto
ID72	Cerramos sesión e intentamos iniciarla con los datos modificados	Se inicia sesión	Correcto

Cuadro 9.8: Pruebas sobre la página de modificación de datos de usuario

9.2. Pruebas automatizadas

Como se ha comentado en apartados anteriores, se ha añadido un plan de pruebas automatizadas para poder comprobar de forma rápida y sencilla el correcto funcionamiento de todos los endpoints de la aplicación.

Pruebas automatizadas para endpoints del módulo login				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID01	Login válido	Probamos el endpoint /login/getUser con valores existentes en la base de datos	La respuesta obtenida tiene estado 200, es JSON y contiene el id correspondiente a dichos valores	Correcto
ID02	Login erróneo	Probamos el endpoint /login/getUser con valores que no existan en la base de datos	La respuesta obtenida tiene estado 200 pero no contiene ningún id	Correcto
ID03	Login sin parámetro (username)	Probamos el endpoint /login/getUser con el parámetro username vacío	La respuesta obtenida tiene estado 200 pero no contiene ningún id	Correcto
ID04	Login sin parámetro (password)	Probamos el endpoint /login/getUser con el parámetro password vacío	La respuesta obtenida tiene estado 200 pero no contiene ningún id	Correcto

Cuadro 9.9: Pruebas automatizadas para endpoints del módulo login

Pruebas automatizadas para endpoints del módulo de cambiar contraseña				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID05	Comprobar usuario existente	Probamos el endpoint /resetPassword/checkUser con valores existentes en la base de datos	La respuesta obtenida tiene estado 200, es JSON y contiene el id correspondiente a dichos valores	Correcto
ID06	Comprobar usuario inexistente	Probamos el endpoint /resetPassword/checkUser con valores que no existen en la base de datos	La respuesta obtenida tiene estado 200 y está vacía	Correcto
ID07	Comprobar usuario sin parámetro (username)	Probamos el endpoint /resetPassword/checkUser con el parámetro username vacío	La respuesta obtenida tiene estado 200 y está vacía	Correcto
ID08	Comprobar usuario sin parámetro (email)	Probamos el endpoint /resetPassword/checkUser con el parámetro email vacío	La respuesta obtenida tiene estado 200 y está vacía	Correcto
ID09	Guardar contraseña	Probamos el endpoint /resetPassword/savePassword con valores válidos	La respuesta obtenida tiene estado 200, es JSON y devuelve el valor true	Correcto
ID10	Guardar contraseña con id 0	Probamos el endpoint /resetPassword/savePassword con una id que no puede existir para ningún usuario	La respuesta obtenida tiene estado 200, es JSON y devuelve el valor false	Correcto
ID11	Guardar contraseña sin parámetro (id)	Probamos el endpoint /resetPassword/savePassword con el parámetro id vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID12	Guardar contraseña sin parámetro (password)	Probamos el endpoint /resetPassword/savePassword con el parámetro password vacío	La respuesta obtenida tiene estado 200, es JSON y devuelve el valor false	Correcto

Cuadro 9.10: Pruebas automatizadas para endpoints del módulo de cambiar contraseña

Pruebas automatizadas para endpoints del módulo de cambiar contraseña I				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID13	Guardar usuario	Probamos el endpoint /register/saveUser con todos los parámetros	La respuesta obtenida tiene estado 200, es JSON, la respuesta es true y el nuevo usuario se almacena en base de datos	Correcto
ID14	Guardar usuario sin parámetro (username)	Probamos el endpoint /register/saveUser con el parámetro username vacío	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto
ID15	Guardar usuario sin parámetro (password)	Probamos el endpoint /register/saveUser con el parámetro password vacío	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto
ID16	Guardar usuario sin parámetro (email)	Probamos el endpoint /register/saveUser con el parámetro email vacío	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto
ID17	Comprobar usuario nuevo	Probamos el endpoint /register/checkUser con un nombre de usuario que no exista en base de datos	La respuesta obtenida tiene estado 200, es JSON y contiene el valor true	Correcto
ID18	Comprobar usuario existente	Probamos el endpoint /register/checkUser con un nombre de usuario que exista actualmente en base de datos	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto
ID19	Comprobar usuario sin parámetro (username)	Probamos el endpoint /register/-checkUser con el parámetro username vacío	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto

Pruebas automatizadas para endpoints del módulo de cambiar contraseña II				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID20	Comprobar email nuevo	Probamos el endpoint /register/checkUser con un email que no exista en base de datos	La respuesta obtenida tiene estado 200, es JSON y contiene el valor true	Correcto
ID21	Comprobar email existente	Probamos el endpoint /register/checkUser con un email que exista actualmente en base de datos	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto
ID22	Comprobar email sin parámetro (email)	Probamos el endpoint /register/-checkUser con el parámetro email vacío	La respuesta obtenida tiene estado 200, es JSON y contiene el valor false	Correcto

Cuadro 9.11: Pruebas automatizadas para endpoints del módulo de cambiar contraseña

Pruebas automatizadas para endpoints de la página principal I				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID23	Borrar usuario	Probamos el endpoint /home/deleteUser con un id existente en base de datos	La respuesta tiene estado 200, es JSON, contiene el valor true, y el usuario con id indicada y todas sus transcripciones se eliminan de base de datos y del sistema de archivos	Correcto
ID24	Borrar usuario con id 0	Probamos el endpoint /home/deleteUser con un valor de id que no puede existir en la base de datos	La respuesta tiene estado 200, es JSON, contiene el valor false	Correcto
ID25	Borrar usuario sin parámetro (id)	Probamos el endpoint /home/deleteUser con el parámetro id vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID26	Subir audio	Probamos el endpoint /home/uploadAudio con un fichero adecuado	La respuesta tiene estado 200, es JSON, contiene el valor true y el tiempo de respuesta es adecuado (menor a 1 segundo)	Correcto
ID27	Subir audio sin parámetro (files)	Probamos el endpoint /home/uploadAudio con el parámetro files vacío	La respuesta tiene un estado 500, el error es "Internal Server Error" y contiene el mensaje "Current request is not a multipart request"	Correcto

Pruebas automatizadas para endpoints de la página principal II				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID28	Transcribir	Probamos el endpoint /home/transcribe con parámetros correctos	La respuesta tiene un estado 200, contiene un objeto del tipo "Transcripciones" con todos sus campos, y estos campos no están vacíos	Correcto
ID29	Transcribir sin parámetro (idUser)	Probamos el endpoint /home/transcribe con el parámetro idUser vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'idUser' is not present."	Correcto
ID30	Transcribir sin parámetro (filename)	Probamos el endpoint /home/transcribe con el parámetro de nombre del fichero vacío	La respuesta tiene estado 200 pero su valor es null	Correcto
ID31	Ver PDF	Probamos el endpoint /home/seePDF con valores correctos y existentes en base de datos	La respuesta tiene estado 200 y contiene el texto adecuado al texto de la transcripción que tiene ese id en base de datos	Correcto
ID32	Ver PDF con id 0	Probamos el endpoint /home/seePDF con un valor de id que no puede existir en base de datos	La respuesta tiene un estado 200 pero no contiene texto	Correcto

Pruebas automatizadas para endpoints de la página principal III				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID33	Ver PDF sin parámetro (idFile)	Probamos el endpoint /home/seePDF con el parámetro idFile vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'idFile' is not present."	Correcto
ID34	Descargar fichero	Probamos el endpoint /home/downloadFile con un valor existente en base de datos	La respuesta tiene un estado 200, el contenido es un PDF y contiene el encabezado.	Correcto
ID35	Descargar fichero con id 0	Probamos el endpoint /home/downloadFile con un valor de id que no puede existir en base de datos	La respuesta tiene un estado 200 pero está vacía	Correcto
ID36	Descargar fichero sin parámetro (id)	Probamos el endpoint /home/downloadFile con el parámetro idFile vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'idFile' is not present."	Correcto

Cuadro 9.12: Pruebas automatizadas para endpoints de la página principal

Pruebas automatizadas para endpoints del módulo de conversiones I				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID37	Obtener transcripciones	Probamos el endpoint /userTranscriptions/getTranscriptions con un id asociado a un usuario que haya realizado transcripciones	La respuesta obtenida tiene estado 200, es JSON, devuelve un array, y este array tiene el mismo número de valores que transcripciones realizadas por el usuario	Correcto
ID38	Obtener transcripciones id inexistente	Probamos el endpoint /userTranscriptions/getTranscriptions con un id que no corresponda a ningún usuario	La respuesta obtenida tiene estado 200, es JSON y devuelve un array vacío	Correcto
ID39	Obtener transcripciones sin parámetro (id)	Probamos el endpoint /userTranscriptions/getTranscriptions	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID40	Ver PDF	Probamos el endpoint /userTranscriptions/seePDF con valores correctos y existentes en base de datos	La respuesta tiene estado 200 y contiene el texto adecuado al texto de la transcripción que tiene ese id en base de datos	Correcto
ID41	Ver PDF con id 0	Probamos el endpoint /userTranscriptions/seePDF con un valor de id que no puede existir en base de datos	La respuesta tiene un estado 200 pero no contiene texto	Correcto
ID42	Ver PDF sin parámetro (idFile)	Probamos el endpoint /userTranscriptions/seePDF con el parámetro idFile vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'idFile' is not present."	Correcto

Pruebas automatizadas para endpoints del módulo de conversiones II				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID43	Descargar fichero	Probamos el endpoint /userTranscriptions/downloadFile con un valor existente en base de datos	La respuesta tiene un estado 200, el contenido es un PDF y contiene el encabezado.	Correcto
ID44	Descargar fichero con id 0	Probamos el endpoint /userTranscriptions/downloadFile con un valor de id que no puede existir en base de datos	La respuesta tiene un estado 200 pero está vacía	Correcto
ID45	Descargar fichero sin parámetro (id)	Probamos el endpoint /userTranscriptions/downloadFile con el parámetro idFile vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'idFile' is not present."	Correcto
ID46	Borrar fichero	Probamos el endpoint /userTranscriptions/deleteFile con un id asociado a un fichero en base de datos	La respuesta tiene estado 200, es JSON, contiene el valor true, el registro ha sido eliminado de base de datos, y el fichero se ha eliminado del sistema de archivos	Correcto
ID47	Borrar fichero con id 0	Probamos el endpoint /userTranscriptions/deleteFile con un id que no puede estar asociada a ningún fichero en base de datos	La respuesta tiene estado 200, es JSON, contiene el valor false	Correcto

Pruebas automatizadas para endpoints del módulo de conversiones III				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID48	Borrar fichero sin parámetro (idFile)	Probamos el endpoint /userTranscriptions/deleteFile con el parámetro idFile vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'idFile' is not present."	Correcto

Cuadro 9.13: Pruebas automatizadas para endpoints del módulo de conversiones

Pruebas automatizadas para endpoints del módulo de estadísticas I				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID49	Obtener transcripciones	Probamos el endpoint /profile/getTranscriptions con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene estado 200, es JSON, devuelve un array que no está vacío, y todos los elementos del array tienen las propiedades esperadas y corresponden al mismo usuario indicado en el parámetro	Correcto
ID50	Obtener transcripciones id inexistente	Probamos el endpoint /profile/getTranscrip tions con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es JSON y devuelve un array vacío	Correcto
ID51	Obtener transcripciones sin parámetro (id)	Probamos el endpoint /profile/getTranscript ions con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID52	Consultar lenguajes	Probamos el endpoint /profile/getLanguages con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene estado 200, es un string y este string contiene los lenguajes esperados para ese usuario, que se corresponden con los distintos idiomas utilizados en sus transcripciones	Correcto

Pruebas automatizadas para endpoints del módulo de estadísticas II				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID53	Consultar lenguajes id inexistente	Probamos el endpoint /profile/-getLanguages con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es un string y ese string está vacío	Correcto
ID54	Consultar lenguajes sin parámetro (id)	Probamos el endpoint /profile/-getLanguages con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID55	Obtener Lenguaje máximo	Probamos el endpoint /profile/get-MaxLang uage con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene un estado 200, es un string y contiene el idioma esperado, correspondiente al idioma que más veces ha transcrito ese usuario	Correcto
ID56	Obtener Lenguaje máximo id inexistente	Probamos el endpoint /profile/get-MaxLang uage con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es un string y ese string está vacío	Correcto
ID57	Obtener Lenguaje máximo sin parámetro (id)	Probamos el endpoint /profile/get-MaxLang uage con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto

Pruebas automatizadas para endpoints del módulo de estadísticas III				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID58	Obtener usos	Probamos el endpoint /profile/getMaxUse con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene estado 200, es un string y ese string contiene el valor esperado, correspondiente al número de veces que ha sido utilizado el lenguaje más usado	Correcto
ID59	Obtener usos id inexistente	Probamos el endpoint /profile/getMaxUse con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es un string y ese string está vacío	Correcto
ID60	Obtener usos sin parámetro (id)	Probamos el endpoint /profile/getMaxUse con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID61	Obtener primera fecha	Probamos el endpoint /profile/getFirstDate con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene estado 200, es un string y contiene la fecha esperada, correspondiente a la fecha en la que el usuario indicado realizó la primera transcripción	Correcto
ID62	Obtener primera fecha id inexistente	Probamos el endpoint /profile/getFirstDate con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es un string y ese string está vacío	Correcto

Pruebas automatizadas para endpoints del módulo de estadísticas IV				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID63	Obtener primera fecha sin parámetro (id)	Probamos el endpoint /profile/getFirstDate con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID64	Obtener última fecha	Probamos el endpoint /profile/getLastDate con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene estado 200, es un string y contiene la fecha esperada, correspondiente a la fecha en la que el usuario indicado realizó la última transcripción	Correcto
ID65	Obtener última fecha id inexistente	Probamos el endpoint /profile/getLastDate con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es un string y ese string está vacío	Correcto
ID66	Obtener última fecha sin parámetro (id)	Probamos el endpoint /profile/getLastDate con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID67	Obtener media	Probamos el endpoint /profile/getAverage con un id de usuario que exista en base de datos y haya realizado alguna transcripción	La respuesta tiene estado 200, es un string y contiene el valor esperado, correspondiente a la media de transcripciones diarias del usuario	Correcto

Pruebas automatizadas para endpoints del módulo de estadísticas V				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID68	Obtener media id inexistente	Probamos el endpoint /profile/getAverage con un id que no esté asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es un string y ese string está vacío	Correcto
ID69	Obtener media sin parámetro (id)	Probamos el endpoint /profile/getAverage con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto

Cuadro 9.14: Pruebas automatizadas para endpoints del módulo de estadísticas

Pruebas automatizadas para endpoints del módulo de datos de usuario I				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID70	Consultar datos	Probamos el endpoint /data/getData con el id de un usuario existente en base de datos	La respuesta tiene estado 200 y es un array que contiene los datos del usuario esperados	Correcto
ID71	Consultar datos id inexistente	Probamos el endpoint /data/getData con un id que no esté asociado a ningún usuario	La respuesta tiene estado 200 y devuelve un array vacío	Correcto
ID72	Consultar datos sin parámetro (id)	Probamos el endpoint /data/getData con el parámetro id vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID73	Guardar nombre	Probamos el endpoint /data/saveName con un nombre distinto al actual y un id asociado a un usuario en base de datos	La respuesta tiene estado 200, es JSON, devuelve el valor true y el nombre de usuario se modifica en la base de datos	Correcto
ID74	Guardar nombre sin parámetro (username)	Probamos el endpoint /data/saveName con el parámetro username vacío	La respuesta tiene estado 200, es JSON y devuelve el valor false	Correcto
ID75	Guardar nombre sin parámetro (id)	Probamos el endpoint /data/saveName con el parámetro id vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto

Pruebas automatizadas para endpoints del módulo de datos de usuario II				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID76	Guardar nombre id 0	Probamos el endpoint /data/saveName con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es JSON y devuelve el valor false	Correcto
ID77	Guardar email	Probamos el endpoint /data/saveEmail con un email distinto al actual y un id asociado a un usuario en base de datos	La respuesta tiene estado 200, es JSON, devuelve el valor true y el email se modifica en la base de datos	Correcto
ID78	Guardar email sin parámetro (email)	Probamos el endpoint /data/saveEmail con el parámetro email vacío	La respuesta tiene estado 200, es JSON y devuelve el valor false	Correcto
ID79	Guardar email sin parámetro (id)	Probamos el endpoint /data/saveEmail con el parámetro id vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID80	Guardar email id 0	Probamos el endpoint /data/saveEmail con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es JSON y devuelve el valor false	Correcto
ID81	Guardar contraseña	Probamos el endpoint /data/savePassword con una contraseña distinta a la actual y un id asociado a un usuario en base de datos	La respuesta tiene estado 200, es JSON, devuelve el valor true y la contraseña se modifica en la base de datos	Correcto

Pruebas automatizadas para endpoints del módulo de datos de usuario III				
ID	**Nombre**	**Descripción**	**Resultado esperado**	**Resultado obtenido**
ID82	Guardar contraseña sin parámetro (password)	Probamos el endpoint /data/save-Password con el parámetro password vacío	La respuesta tiene estado 200, es JSON y devuelve el valor false	Correcto
ID83	Guardar contraseña sin parámetro (id)	Probamos el endpoint /data/save-Password con el parámetro id vacío	La respuesta obtenida tiene estado 400, es JSON, el error que muestra es "Bad request" y contiene el mensaje "Required parameter 'id' is not present."	Correcto
ID84	Guardar contraseña id 0	Probamos el endpoint /data/save-Password con un id que no puede estar asociado a ningún usuario en base de datos	La respuesta tiene estado 200, es JSON y devuelve el valor false	Correcto

Cuadro 9.15: Pruebas automatizadas para endpoints del módulo de datos de usuario

9.3. Ejecución de las pruebas automatizadas

Una vez se tienen las pruebas definidas y agrupadas en una misma colección, Postman permite la ejecución continua de todas ellas junto con parámetros de configuración de la ejecución.

En este caso en particular, se debe tener en cuenta, en caso de que se realicen ejecuciones continuadas, que hay pruebas diseñadas para eliminar valores en la base de datos a través de una id proporcionada, por lo tanto, para cada ejecución, se debe modificar dicha id para evitar que el registro ya no se encuentre presente por haber sido eliminado en la prueba anterior. Estas pruebas son la prueba ID23 y la prueba ID46.

Una vez realizada la ejecución, obtenemos una duración total de 11.558 segundos de ejecución total, con 272 test realizados, todos de forma satisfactoria.

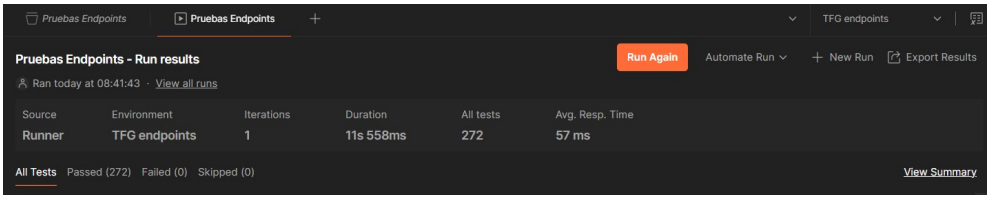

Figura 43: Resultados de la ejecución de las pruebas en Postman

Apéndice A.

Documentación OpenAPI

/v3/api-docs Explore

Aplicación de transcripción del audio de videos y podcast 0.0.1 OAS3

/v3/api-docs

Aplicación que permite la generación de ficheros PDF que contienen la transcripción de un fichero de audio

Servers

http://localhost:8080 - Generated server url

Login Gestiona el acceso de los usuarios a la aplicación.

GET `/login/getUser` Comprueba si existe en en la base de datos algún registro con el nombre de usuario y contraseña indicados.

Recibe desde el front como parámetros un nombre de usuario y contraseña, y busca en la tabla USUARIOS si existe un registro que contenga esos valores.En caso afirmativo, accede a la aplicación y almacena el nombre de usuario y su identificador en la sesión local

Parameters Try it out

Name	Description
username * required `string` *(query)*	Nombre de usuario introducido. username
password * required `string` *(query)*	Contraseña introducida. password

Responses

Code	Description	Links
200	La llamada ha sido existosa, puede devolver el idcorrespondiente o null en caso de no existir Media type: application/json Controls `Accept` header. Examples: Usuario encontrado **Example Value** `1` **Example Description** Los parámetros corresponden al usuario con id=1	*No links*

Cambiar contraseña

Gestiona la modificación de la contraseña de un usuario sin necesidad de iniciar sesión. Se realiza en dos fases; en la primera, el usuario debe introducir su nombre de usuario y su email; si estos son correctos, podrá modificar la contraseña

POST `/resetPassword/savePassword` Modifica la contraseña del usuario

Recibe el identificador de usuario y la nueva contraseña como parámetros, accede a la base de datos, obtiene el usuario asociado al identificador y sustituye la anterior contraseña por la recibida. Esta contraseña ya debe cumplir los parámetros de robustez, dado que se controla desde el front

Parameters

Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id
password * required `string`	Nueva contraseña indicada por el usuario

Name	Description
(query)	password

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede devolver true si se puede cambiar o false en caso contrario Media type: application/json Examples: Usuario encontrado Controls `Accept` header. **Example Value** `true` **Example Description** Encuentra al usuario en base de datos y se ha podido modificar la contraseña	*No links*
400	Identificador vacío Media type: application/json Examples: Error al modificar la contraseña c **Example Value** `"Required parameter 'id' is not present."` **Example Description** El parámetro id no se envía al back	*No links*

GET `/resetPassword/checkUser` Comprueba que el usuario es correcto

Recibe como parámetros el nombre de usuario y el email que se ha introducido, y accede a la base de datos de USUARIO para comprobar si existe un registro que coincida con ambos valores

Parameters Try it out

Name	Description
username * required `string` *(query)*	Nombre de usuario introducido username
email * required `string` *(query)*	Dirección de email introducida email

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria, y puede devolver el identificador del usuario con los datos introducidos, o null si no existe Media type: **application/json** Controls `Accept` header. Examples: **Usuario encontrado** **Example Value** `1` **Example Description** Los parámetros corresponden al usuario con id=1	*No links*

Datos de Usuario

Permite a los usuarios del sistema modificar sus datos: nombre de usuario, email y contraseña

POST `/data/savePassword` Gestiona la modificación de la contraseña del usuario

Recibe como parámetro el identificador del usuario y la nueva contraseña, accede a la tabla de USUARIOS y cambia la contrasñea actual por la recibida. La validez de la robustez de la contraseña se controla desde el front

Parameters Try it out

Name	Description
password * required `string` *(query)*	Nueva contrasela del usuario password
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido existosa y puede devolver un valor true si se ha podido modificar la contraseña, o false en caso contrario Media type: application/json Controls `Accept` header. Examples: Usuario encontrado **Example Value** `true` **Example Description**	*No links*

Code	Description	Links
	Se ha podido modificar la contraseña	
400	Identificador vacío Media type: application/json Examples: Error al cambiar la contraseña **Example Value** `"Required parameter 'id' is not present."` **Example Description** El parámetro id no se envía al back	*No links*

POST `/data/saveName` Gestiona la modificación del nombre de usuario

Recibe como parámetro el identificador del usuario y el nuevo nombre, accede a la tabla de USUARIOS y modifica el nombre de usuario poniendo el recibido por parámetro

Parameters Try it out

Name	Description
username * required `string` *(query)*	Nuevo nombre de usuario username
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido existosa y puede devolver un valor ture si ha podido cambiar el nombre, o false en caso contrario Media type: application/json Controls `Accept` header. Examples: Usuario encontrado **Example Value** `true` **Example Description** Se ha podido modificar el nombre de usuario	*No links*
400	Identificador vacío Media type: application/json Examples: Error al cambiar el nombre de us **Example Value** `"Required parameter 'id' is not present."` **Example Description** El parámetro id no se envía al back	*No links*

POST `/data/saveEmail` Gestiona la modificación del email

Recibe como parámetro el identificador del usuario y el nuevo email, accede a la tabla de USUARIOS y modifica el email, sustituyendo el anterior por el recibido por parámetro. La validez del formato del parámetro email se controla desde el front

Parameters Try it out

Name	Description
email * required `string` *(query)*	Nuevo email del usuario email
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido existosa y puede devolver un valor true si ha podido cambiar el email, o false en caso contrario Media type: application/json Examples: Usuario encontrado Controls `Accept` header. **Example Value** `true` **Example Description** Se ha podido modificar el email	*No links*
400	Identificador vacío Media type: application/json Examples: Error al cambiar el email **Example Value** `"Required parameter 'id' is not present."`	*No links*

Code	Description	Links

Example Description

El parámetro id no se envía al back

GET `/data/getData` Consulta los datos actuales

Se llama automáticamente al cargar la página de edición de datos. Accede a la tabla USUARIOS y devuelve los valores actuales de base de datos para ese identificador

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede devolver los datos encontrados Media type: application/json Examples: Usuario encontrado Controls `Accept` header. **Example Value** Schema `{ "id": 34, "name": "usuarioPrueba", "password": "QWErty12.@", "email": "prueba@gmail.com", "creation_date": "2024-10-10T18:44:01.000+00:00" }` **Example Description**	*No links*

Code	Description	Links
	Ha encontrado un usuario y devuelve sus datos	
400	Identificador vacío Media type: application/json Examples: Error al obtener los datos del usu **Example Value** `"Required parameter 'id' is not present."` **Example Description** El parámetro id no se envía al back	*No links*

Registro

Permite a los usuarios darse de alta en el sistema para poder obtener más funcionalidades.

POST `/register/saveUser` Inserta al nuevo usuario en base de datos

Recibe por parámetros el nombre de usuairo, email y contraseña del usuario que se va a dar de alta, crea un nuevo Objeto usuario, le asigna dichos valores, y lo almacena en un nuevo registro de la tabla USUARIO

Parameters Try it out

Name	Description
username * required `string` *(query)*	Nombre de usuario que se va a dar de alta username
email * required `string` *(query)*	Email del usuario que se va a dar de alta email
password * required	Contraseña del usuario que se va a dar de alta

Name	Description
string *(query)*	password

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria, puede devolver true o false en función de si se puede o no guardar el nuevo usuario Media type: application/json Controls `Accept` header. Examples: Usuario registrado **Example Value** `true` **Example Description** Se crea el nuevo usuario en base de datos	*No links*

GET `/register/checkUser` Comprueba que el nombre de usuario no exista

Para evitar que existan dos usuarios con el mismo nombre de usuario, accede a la base de datos y comprueba que el usuario introducido no existe acutalmente en la tabla USUARIO. Se llama a la función de forma recurrente cada vez que se escribe un nuevo caracter en el formulario de registro

Parameters Try it out

Name	Description
username * required string *(query)*	Nombre de usuario introducido username

Responses

Code	Description	Links
200	La llamada se realiza correctamente y se puede comprobar si ya existe el nombre de usuario en base de datos Media type: application/json Controls `Accept` header. Examples: Usuario existente **Example Value** `false` **Example Description** El nombre de usuario ya está en uso	*No links*

GET `/register/checkEmail` Comprueba que el email introducido no exista actualmente

Para evitar que un usuario se registre dos veces con el mismo correo, accede a la base de datos y comprueba que el email introducido no existe acutalmente en la tabla USUARIO. Se llama a la función de forma recurrente cada vez que se escribe un nuevo caracter en el formulario de registro

Parameters

Try it out

Name	Description
email * required `string` *(query)*	Email introducido en la aplicación email

Responses

Code	Description	Links
200	La llamada se realiza correctamente y se puede comprobar si ya existe el email en la base de datos Media type: application/json Controls `Accept` header. Examples: Email existente **Example Value** `false` **Example Description** El email ya está en uso	*No links*

Perfil de estadísticas de usuario

Permite al usuario visualizar diversas estadísticas de uso de la aplicación. El número total de transcripciones realizadas, todos los idiomas utilizados, el idioma más utilizado y su número de usos, las fechas de la primera y última transcripción, y una media de transcripciones

GET `/profile/getTranscriptions` Consulta las transcripciones realizadas por el usuario

Recibe como parámetro desde el front el identificador único de usuario, y consulta la base de datos de TRANSCRIPCIONES para obtener una lista con todos sus datos

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido exitosa. Puede devolver una lista con las transcripciones del usaurio Media type: application/json Controls `Accept` header. Examples: Usuario con transcripciones **Example Value** Schema [{ "id": 1, "id_user": 31, "date": "2024-06-22T15:26:35.000+00:00", "file": "audio1", "language": "SPANISH", "original_file": "53466__dobroide__randomspanish_words.wav", "text": "éxtasis, congénito, abanico, extirpar, embate, faccioso, radioscópico, vareo, éxtasis, bubónico, músculo, nieto, tampoco, hidrófobo, nuez, principio, invertebrado, meninge, populoso, comprada, éxtasis, tiznar, retama, vareo, tampoco, faccioso, éxtasis, extirpar, bubónico, hipnosis, invertebrado, músculo, congénito, meninge, hidrófobo, principio, nieto, populoso, comprada, nuez, éxtasis." }, { "id": 2, "id_user": 31, "date": "2024-06-22T15:26:40.000+00:00", "file": "audio2", "language": "SPANISH", "original_file": "204974__dobroide__20131102_poem.wav", "text": "Me gusta verlos pintarse, de sol y grana volar, sobre el cielo azul temblar, súbitamente y quebrarse. Nunca perseguí la gloria de dejar en la memoria de los hombres mi canción. Amo los mundos sutiles, ingrávidos y gentiles, como pompas de jabón." }, **Example Description** El usuario ha realizado una o varias transcripciones	*No links*
400	Identificador vacío Media type: application/json Examples: Error al obtener las transcripcion **Example Value** "Required parameter 'id' is not present"	*No links*

Code	Description	Links

Example Description

El parámetro id no se recibe correctamente

GET `/profile/getMaxUse` Consulta el número de usos del lenguaje más utilizado

Recibe como parámetro el identificador único del usuario y realiza una consulta sobre la tabla TRANSCRIPCIONES para obtener un listado de todos los distintos idiomas utilizados por el usuario junto con su número de usos y ordenados de más a menos usado; tras esto, obtiene el primer valor para mostrarlo como número de veces del idioma más utilizado

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede obtener el número de usos del lenguaje más recurrente del usuario Media type: application/json Controls `Accept` header. Examples: Número de usos encontrado **Example Value** 5	*No links*

Code	Description	Links
	Example Description El usuario ha realizado transcripciones y se puede obtener un lenguaje más utilizado y el número de veces	
400	Identificador vacío Media type: application/json Examples: Error al obtener el lenguaje más **Example Value** `"Required parameter 'id' is not present"` **Example Description** El parámetro id no se recibe correctamente	*No links*

GET `/profile/getMaxLanguage` Consulta el lenguaje más utilizado

Recibe como parámetro el identificador único del usuario y realiza una consulta sobre la tabla TRANSCRIPCIONES para obtener un listado de todos los distintos idiomas utilizados por el usuario, ordenados de más a menos usado; tras esto, obtiene el primer valor para mostrarlo como valor más utilizado

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede obtener el lenguaje Media type: application/json Controls `Accept` header. Examples: Lenguaje más utilizado encontra **Example Value** `"SPANISH"` **Example Description** El usuario ha utilizado uno o varios lenguajes	*No links*
400	Identificador vacío Media type: application/json Examples: Error al obtener el lenguaje más **Example Value** `"Required parameter 'id' is not present"` **Example Description** El parámetro id no se recibe correctamente	*No links*

GET `/profile/getLastDate` Consulta la última fecha en la que el usuario realizó una transcripción

Recibe como parámetro el identificador único del usuario y realiza una consulta sobre la tabla TRANSCRIPCIONES para obtener un listado de las fechas en las que el usuario ha realizado transcripciones. Ordena este listado en orden descendente y selecciona el primer valor, que se formatea para mostrarlo en un formato más visual de dd-mm-yyyy

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede obtener la última fecha en la que se realizó una transcripción Media type: **application/json** (Controls `Accept` header.) Examples: **Primera fecha encontrada** **Example Value** `"19-08-2024"` **Example Description** El usuario ha realizado transcripciones y se puede obtener una fecha válida	*No links*
400	Identificador vacío Media type: **application/json** Examples: **Error al obtener la última fecha d** **Example Value** `"Required parameter 'id' is not present"` **Example Description** El parámetro id no se recibe correctamente	*No links*

Code	Description	Links

GET `/profile/getLanguages` Consulta los lenguajes utilizados

Recibe como parámetro el identificador único del usuario y realiza una consulta sobre la tabla TRANSCRIPCIONES para obtener un listado de todos los distintos idiomas utilizados por el usuario; tras esto, formatea la respuesta para crear una cadena de texto con todos los lenguajes

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede obtener una lista de lenguajes Media type: application/json Examples: Lenguajes encontrados Controls `Accept` header. **Example Value** `"ENGLISH, SPANISH, ITALIAN"` **Example Description** El usuario ha utilizado uno o varios lenguajes	*No links*
400	Identificador vacío	*No links*

Code	Description	Links

Media type: application/json

Examples: Error al obtener los lenguajes

Example Value

```
"Required parameter 'id' is not present"
```

Example Description

El parámetro id no se recibe correctamente

GET /profile/getFirstDate

Consulta la primera fecha en la que el usuario realizó una transcripción

Recibe como parámetro el identificador único del usuario y realiza una consulta sobre la tabla TRANSCRIPCIONES para obtener un listado de las fechas en las que el usuario ha realizado transcripciones. Ordena este listado en orden ascendente y selecciona el primer valor, que se formatea para mostrarlo en un formato más visual de ddmmyyyy

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede obtener la primera fecha en la que se realizó una transcripción Media type: application/json Examples: Primera fecha encontrada	*No links*

Code	Description	Links

Controls `Accept` header.

Example Value

```
"22-06-2024"
```

Example Description

El usuario ha realizado transcripciones y se puede obtener una fecha válida

400

Identificador vacío

No links

Media type: **application/json**

Examples: **Error al obtener la primera fecha**

Example Value

```
"Required parameter 'id' is not present"
```

Example Description

El parámetro id no se recibe correctamente

GET `/profile/getAverage` Obtiene una media de transcripciones diarias realizadas por el usuario

Recibe como parámetro el identificador único del usuario y realiza una consulta sobre la tabla TRANSCRIPCIONES para obtener un listado de las fechas en las que el usuario ha realizado transcripciones. Ordena este listado en orden ascendente y selecciona el primer valor, y, posteriormente, ordena el listado de forma descendente y repite la operación, obteniendo la primera y última fecha en la que se realizaron transcripciones. Con estos valores, calcula la diferencia en milisegundos y posteriormente transforma esta diferencia en días. Una vez se tiene la diferencia en días, se consulta de nuevo la base de datos para obtener el número total de transcripciones realizadas. Por último, el número de transcripciones del usuario se divide entre la diferencia en días entre la primera y última transcripción, obteniendo así un valor de transcripciones realizadas por día. En caso de que ambas fechas sean la misma, para evitar que la diferencia sea 0 y sea imposible dividir, se devuelve como valor únicamente el número total de transcripciones

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido satisfactoria y puede obtener un valor Media type: **application/json** (Controls `Accept` header.) Examples: **Primera fecha encontrada** **Example Value** `0.21` **Example Description** `Se puede calcular un valor de transcripciones diarias	*No links*
400	Identificador vacío Media type: **application/json** Examples: **Error al obtener la última fecha d** **Example Value** `"Required parameter 'id' is not present"` **Example Description** El parámetro id no se recibe correctamente	*No links*

Home

Permite realizar transcripciones de ficheros de audio con formato mp3, mp4, mpeg, mpga, m4a, wav y webm; y soporta más de 50 idiomas. Con al menos una transcripción realizada, permite ver el PDF resultado y descargarlo. Si el usuario estáregistrado en el sistema, también permite borrar todos sus datos

POST `/home/uploadAudio` Sube los audios a transcribir a una carpeta temporal

Recibe como parámetro una lista de los audios a transcribir (la validación de la extensión se comprueba desde el front) y los mueve al directorio temporal por defecto del sistema operativo

Parameters

Try it out

Name	Description
files * required `array[string]` *(query)*	Objeto de tipo MultipartFile[] que contiene todos los ficheros que el usuario ha seleccionado para transcribir

Responses

Code	Description	Links
200	La llamada ha sido existosa y puede devolver el valor booleano correspondiente al resultado de la operación Media type: application/json Controls `Accept` header. Examples: Ficheros subidos correctamente **Example Value** `true` **Example Description** Los ficheros seleccionados por el usuario han sido procesados correctamente y se encuentran en el directorio temporal	*No links*
500	Parámetro vacío	*No links*

Code	Description	Links

Media type: application/json

Examples: Error al seleccionar los ficheros

Example Value

```
"Internal server error"
```

Example Description

Los ficheros no se han podido enviar correctamente y se recibe el parámetro vacío

POST /home/deleteUser Elimina el usuario y sus datos

Recibe como parámetro el identificador único del usuario y elimina ese registro de la base de datos de USUARIOS, todas los registros de la base de datos de TRANSCRIPCIONES asociadas al usuario, y esos mismos archivos del sistema de ficheros

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario almacenado en la sesión local id

Responses

Code	Description	Links
200	La llamada ha sido exitosa. Puede devolver true si se ha podido eliminar el usuario o false en caso contrario Media type: application/json Examples	*No links*

Code	Description	Links

Controls `Accept` header.

Usuario encontrado

Example Value

```
true
```

Example Description

El usuario existe en base de datos y se han podido eliminar sus datos

400

No links

Identificador vacío

Media type: application/json

Examples: Error al eliminar usuario

Example Value

```
"Required parameter 'id' is not present."
```

Example Description

El parámetro id no se envía al back

GET `/home/transcribe` Realiza la transcripción de los ficheros

Se llama automáticamente desde el front cuando se ha terminado de subir los archivos. Recibe como parámetros el identificador del usuario (o 0 si es un usuario que no ha iniciado sesión) y el nombre del fichero a transcribir. Busca el fichero recibido en la carpeta temporal del sistema y lo mueve a una carpeta temporal asociada a la ejecución. Tras esto, inicializa el servicio whisper de OpenAI y crea una petición de transcripción del fichero. Una vez finalizada la transcripción, crea un fichero PDF en la carpeta de resultados con el texto obtenido y elimina el fichero de la carpeta temporal de la ejecución. Por último, realiza una llamada a un detector de lenguaje y almacena los valores de la transcripción en base de datos

Parameters

Try it out

Name	Description
idUser * required `integer($int64)` *(query)*	Identificador del usuario que realiza la petición idUser
filename * required `string` *(query)*	Nombre del fichero del que se requiere la transcripción filename

Responses

Code	Description	Links
200	La llamada ha sido exitosa y puede devolver el objeto Transcripción obtenido Media type: application/json Examples: Trasnscipción realizada correcta Controls `Accept` header. **Example Value** Schema `{ "id": 4, "id_user": 31, "date": "2024-06-22 17:26:44", "file": "audio4", "language": "ENGLISH", "original_file": "514703__metrostock99__there-isnt.wav", "text": "Ah, I'll tell you what, there isn't nothing better than a hot bath after a hard day of work. I could sit here all day." }` **Example Description** Se ha realizado la transcripción y se han almacenado los valores en base de datos	*No links*
400	Parámetro idUser vacío Media type: application/json Examples: Error al realizar la transcripción	*No links*

Code	Description	Links

Example Value

```
"Required parameter 'idUser' is not present"
```

Example Description

El parámetro idUser no se ha recibido correctamente

GET `/home/seePDF` Permite visualizar el archivo de la transcripción

Recibe como parámetro el identificador del fichero de transcripción, obtiene el texto de la misma de base de datos y lo devuelve al front, que se encarga de mostrarlo visualmente con formato PDF

Parameters Try it out

Name	Description
idFile * required `integer($int64)` *(query)*	Identificador de la transcripción la cuál se quiere visualizar idFile

Responses

Code	Description	Links
200	La llamada ha sido exitosa y puede devolver el texto de la transcripción Media type: **application/json** Controls `Accept` header. Examples: **Transcripción encontrada** **Example Value** `"Ah, I'll tell you what, there isn't nothing better than a hot bath after a hard day of work. I could sit here all day"`	*No links*

Code	Description	Links
	Example Description Se ha encontrado la transcripción en base de datos y se ha podido obtener su texto	
400	Parámetro idFile vacío Media type: application/json Examples: Error al visualizar el texto de la tr **Example Value** `"Required parameter 'idFile' is not present"` **Example Description** El parámetro idFile no se ha recibido correctamente	*No links*

GET `/home/downloadFile` Permite descargar el PDF con la transcripción

Recibe como parámetro el identificador único de la transcripción a descargar, consulta la tabla de TRANSCRIPCIONES para obtener el nombre del fichero, y realiza una petición HTTP para descargarlo

Parameters Try it out

Name	Description
idFile * required `integer($int64)` *(query)*	Identificador de la transcripción que se quiere descargar idFile

Responses

Code	Description	Links
200	La llamada ha sido exitosa y se ha podido realizar la descarga Media type: **application/json** Controls `Accept` header. Examples: **Transcripción encontrada** **Example Value** ``` { "headers": { "Content-Disposition": "attachment; filename=audio19.pdf", "Content-Type": "application/pdf", "Content-Length": "204800" }, "body": "Archivo binario del PDF." } ``` **Example Description** El fichero indicado por el usuario se encuentra en el sistema de archivos y se puede realizar su descarga	*No links*
400	Parámetro idFile vacío Media type: **application/json** Examples: **Error al descargar la transcripció** **Example Value** ``` "Required parameter 'idFile' is not present" ``` **Example Description** El parámetro idFile no se ha recibido correctamente	*No links*

Transcripciones del usuario

Gestiona las transcripciones anteriores realizadas por el usuario. Permite acceder a todas las transcripciones anteriores, visualizarlas, descargarlas y eliminarlas.

POST `/userTranscriptions/deleteFile` Permite eliminar una transcripción

Recibe como parámetro el identificador de la transcripción que se desea eliminar. Se consulta la base de datos para obtener el nombre del fichero, y se elimina del sistema de archivos. Por último, también se elimina el registro de la tabla de TRANSCRIPCIONES

Parameters Try it out

Name	Description
idFile * required `integer($int64)` *(query)*	Identificador único de la transcripción idFile

Responses

Code	Description	Links
200	La llamada ha sido exitosa y se puede eliminar el fichero si existe Media type: application/json Controls `Accept` header. Examples: Fichero eliminado correctamente **Example Value** `true` **Example Description** El fichero ha sido encontrado y se ha eliminado	*No links*
400	Parámetro idFile vacío Media type: application/json Examples: Error al eliminar la transcripción **Example Value** `"Required parameter 'idFile' is not present"`	*No links*

Code	Description	Links

Example Description

El parámetro idFile no se ha recibido correctamente

GET `/userTranscriptions/seePDF` Permite visualizar el archivo de la transcripción

Recibe como parámetro el identificador único del fichero a visualizar, obtiene el texto de la tabla TRANSCRIPCIONES de base de datos y lo envía al front, que se encarga de mostrarlo visualmente con formato PDF

Parameters Try it out

Name	Description
idFile * required `integer($int64)` *(query)*	Identificador único de la transcripción que se quiere visualizar idFile

Responses

Code	Description	Links
200	La llamada ha sido exitosa y puede devolver el texto de la transcripción Media type: **application/json** (Controls `Accept` header.) Examples: **Transcripción encontrada** **Example Value** `"State ascoltando The Global Voice. Benvenuti alla celebrazione del nostro sesto compleanno."`	*No links*

Code	Description	Links
	Example Description Se ha encontrado la transcripción en base de datos y se ha podido obtener su texto	
400	Parámetro idFile vacío Media type: application/json Examples: Error al visualizar el texto de la tr **Example Value** `"Required parameter 'idFile' is not present"` **Example Description** El parámetro idFile no se ha recibido correctamente	*No links*

GET `/userTranscriptions/getTranscriptions` Consulta las transcripciones realizadas por el usuario

Recibe como parámetro desde el front el identificador único de usuario, y consulta la base de datos de TRANSCRIPCIONES para obtener una lista con todos sus datos

Parameters Try it out

Name	Description
id * required `integer($int64)` *(query)*	Identificador único del usuario id

Responses

Code	Description	Links
200	La llamada ha sido exitosa. Puede devolver una lista con las transcripciones del usaurio Media type: application/json Controls `Accept` header. Examples: Usuario con transcripciones **Example Value** Schema [{ "id": 1, "id_user": 31, "date": "2024-06-22T15:26:35.000+00:00", "file": "audio1", "language": "SPANISH", "original_file": "53466__dobroide__randomspanish_words.wav", "text": "éxtasis, congénito, abanico, extirpar, embate, faccioso, radioscópico, vareo, éxtasis, bubónico, músculo, nieto, tampoco, hidrófobo, nuez, principio, invertebrado, meninge, populoso, comprada, éxtasis, tiznar, retama, vareo, tampoco, faccioso, éxtasis, extirpar, bubónico, hipnosis, invertebrado, músculo, congénito, meninge, hidrófobo, principio, nieto, populoso, comprada, nuez, éxtasis." }, { "id": 2, "id_user": 31, "date": "2024-06-22T15:26:40.000+00:00", "file": "audio2", "language": "SPANISH", "original_file": "204974__dobroide__20131102_poem.wav", "text": "Me gusta verlos pintarse, de sol y grana volar, sobre el cielo azul temblar, súbitamente y quebrarse. Nunca perseguí la gloria de dejar en la memoria de los hombres mi canción. Amo los mundos sutiles, ingrávidos y gentiles, como pompas de jabón." }, **Example Description** El usuario ha realizado una o varias transcripciones	*No links*
400	Identificador vacío Media type: application/json Examples: Error al obtener las transcripcion **Example Value** "Required parameter 'id' is not present" **Example Description**	*No links*

Code	Description	Links
	El parámetro id no se recibe correctamente	

GET `/userTranscriptions/downloadFile` Permite descargar el PDF con la transcripción

Recibe como parámetro el identificador único de la transcripción que se desea descargar, consulta en base de datos la tabla de TRANSCRIPCIONES para obtener el nombre del fichero, y realiza una petición HTTP para descargarlo desde el sistema de archivos

Parameters Try it out

Name	Description
idFile * required `integer($int64)` *(query)*	Identificador único de la transcripción a descargar idFile

Responses

Code	Description	Links
200	La llamada ha sido exitosa y se ha podido realizar la descarga Media type: **application/json** Controls `Accept` header. Examples: **Transcripción encontrada** **Example Value** **Example Description** El fichero indicado por el usuario se encuentra en el sistema de archivos y se puede realizar su descarga	*No links*


```
{
  "headers": {
    "Content-Disposition": "attachment; filename=audio2.pdf",
    "Content-Type": "application/pdf",
    "Content-Length": "204800"
  },
  "body": "Archivo binario del PDF."
}
```

Code	Description	Links
400	Parámetro idFile vacío Media type: application/json Examples: Error al descargar la transcripció **Example Value** `"Required parameter 'idFile' is not present"` **Example Description** El parámetro idFile no se ha recibido correctamente	*No links*

Schemas

Transcripciones

```
{
  description:        Modelo que representa una transcripción en la base de datos
  id                  integer($int64)
                      Identificador único de la transcripción
  id_user             integer($int64)
                      Identificador único del usuario que ha realizado la
                      transcripción
  date                string($date-time)
                      Fecha de creación de la transcripción
  file                string
                      Nombre del fichero donde se ha almacenado la transcripción
  language            string
                      Lenguaje de la transcripción
  original_file       string
                      Nombre del fichero original de audio de la transcripción
  text                string
                      Texto de la transcripción
}
```

Usuarios {

description: Modelo que representa un usuario en la base de datos.

id **integer**($int64)
Identificador único del usuario

name **string**
Nombre de usuario utilizado en la aplicación

password **string**
Contraseña del usuario

email **string**
Email del usuario

creation_date **string**($date-time)
Fecha de creación de la transcripción

}

Bibliografía

[1] IBM. «Speech recognition.» (2024), dirección: https://www.ibm.com/es-es/topics/speech-recognition (visitado 22-06-2024).

[2] A. W. Services. «Speech to text.» (n.d.), dirección: https://aws.amazon.com/es/what-is/speech-to-text/ (visitado 22-06-2024).

[3] «Speech Recognition.» (2024), dirección: https://en.wikipedia.org/wiki/Speech_recognition.

[4] M. Rouse. «Natural Language Processing.» (2024), dirección: https://www.techopedia.com/definition/653/natural-language-processing-nlp (visitado 22-06-2024).

[5] A. Moreno. «Procesamiento del lenguaje natural ¿qué es?» (2018), dirección: https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/.

[6] K. Garrett. «La historia del reconocimiento de voz.» (2023), dirección: https://transcribe.com/es/blog/la-historia-del-reconocimiento-de-voz.

[7] F. Martinez, G. Portale, H. Klein y O. Olmos. «Reconocimiento de voz, apuntes de cátedra para Introducción a la Inteligencia Artificial.» (n.d.), dirección: https://www.frba.utn.edu.ar/wp-content/uploads/2021/02/IA1_IntroReconocimientoVoz.pdf.

[8] Alisys. «De Audrey a Siri: historia y uso de los asistentes virtuales.» (2017), dirección: https://alisys.net//es/blog/de-audrey-a-siri-historia-y-uso-de-los-asistentes-de-virtuales.

[9] Dictate.it. «Speech Recognition from Audrey to Alexa – A Brief History.» (2021), dirección: https://dictateit.com/speech-recognition-from-audrey-to-alexa-a-brief-history/.

[10] K. Moskvitch. «The machines that learned to listen.» (2017), dirección: https://www.bbc.com/future/article/20170214-the-machines-that-learned-to-listen.

[11] «Modelo oculto de Márkov.» (2024), dirección: https://es.wikipedia.org/wiki/Modelo_oculto_de_M%C3%A1rkov.

[12] L. Maldonado, «Los modelos ocultos de Markov, MOM,» Español, *Telos*, n.d. ISSN: 1317-0570. dirección: https://www.redalyc.org/articulo.oa?id=99324907003.

[13] L. N. Albarrán y L. J. R. Esparza, «Modelos ocultos de Markov: una aplicación de estimación bayesiana para series de tiempo financieras,» *Revista Metropolitana de Matemáticas*, vol. 14, n.º 1, págs. 77-99, n.d. dirección: http://mat.izt.uam.mx/mat/documentos/revistaMixbaal/Mixbaal2023-05.pdf.

[14] S. Figueiras. «¿Cómo funcionan las Redes Neuronales?» (2022), dirección: https://www.ceupe.mx/blog/como-funcionan-las-redes-neuronales.html.

[15] A. W. Services. «¿Qué es una red neuronal?» (n.d.), dirección: https://aws.amazon.com/es/what-is/neural-network/.

[16] J. Torres. «Redes Neuronales Recurrentes.» (2019), dirección: https://torres.ai/redes-neuronales-recurrentes/.

[17] «Redes neuronales recurrentes.» (2024), dirección: https://es.wikipedia.org/wiki/Redes_neuronales_recurrentes.

[18] R. Merritt. «¿Qué es un Modelo Transformer?» (2022), dirección: https://la.blogs.nvidia.com/blog/que-es-un-modelo-transformer/.

[19] M. Sotaquirá. «Redes Transformer (... o el fin de las Redes Recurrentes).» (2020), dirección: https://www.codificandobits.com/blog/redes-transformer/.

[20] C. Santander. «¿Qué son los Transformers?» (2021), dirección: https://www.linkedin.com/pulse/transformers-redes-neuronales-cristian-santander/.

[21] elastic. «¿Qué es un modelo de lenguaje grande (LLM)?» (2023), dirección: https://www.elastic.co/es/what-is/large-language-models.

[22] M. Sotaquirá. «Grandes Modelos de Lenguaje (Large Language Models).» (2023), dirección: https://www.codificandobits.com/blog/grandes-modelos-de-lenguaje/.

[23] «Modelo de lenguaje grande.» (2024), dirección: https://es.wikipedia.org/wiki/Modelo_de_lenguaje_grande.

[24] Picovoice. «Best Transcription Software (Free and Paid) to Convert Speech to Text.» (2022), dirección: https://picovoice.ai/blog/top-transcription-engines/.

[25] Toolify.ai. «Las mejores API y bibliotecas de código abierto gratuitas para la conversión de voz a texto.» (2024), dirección: https://www.toolify.ai/es/ai-news-es/las-mejores-api-y-bibliotecas-de-cdigo-abierto-gratuitas-para-la-conversin-de-voz-a-texto-1180624.

[26] OpenAI. «Introducing Whisper.» (2022), dirección: https://openai.com/index/whisper/.

[27] «TensorFlow.» (n.d.), dirección: https://www.tensorflow.org/.

[28] «PyTorch.» (n.d.), dirección: https://pytorch.org/.

[29] «DeepSpeech's documentation.» (2020), dirección: https://deepspeech.readthedocs.io/en/r0.9/#.

[30] «Google Cloud Speech-to-Text.» (n.d.), dirección: https://cloud.google.com/speech-to-text?hl=es.

[31] «Amazon Transcribe.» (n.d.), dirección: https://aws.amazon.com/es/transcribe/.

[32] «Microsoft Azure Speech-to-Text.» (n.d.), dirección: https://azure.microsoft.com/es-es/products/ai-services/speech-to-text.

[33] «AssembyAI.» (n.d.), dirección: https://www.assemblyai.com/.

[34] «Dragon NaturallySpeaking.» (n.d.), dirección: https://www.nuance.com/es-es/dragon.html.

[35] S. Laoyan. «Agile Manifesto: la guía para entender la metodología Agile.» (2024), dirección: https://asana.com/es/resources/agile-methodology.

[36] «Desarrollo ágil de software.» (2024), dirección: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software.

[37] K. Beck, M. Beedle, A. van Bennekum et al. «Manifiesto por el Desarrollo Ágil de Software.» (2001), dirección: https://agilemanifesto.org/iso/es/manifesto.html.

[38] «Scrum Org.» (n.d.), dirección: https://www.scrum.org/.

[39] «What is Scrum?» (n.d.), dirección: https://www.scrum.org/resources/what-scrum-module.

[40] J. Roche. «Scrum: roles y responsabilidades.» (n.d.), dirección: https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html.

[41] J. Roche. «Las 5 ceremonias Scrum: claves para la gestión de procesos.» (n.d.), dirección: https://www2.deloitte.com/es/es/pages/technology/articles/ceremonias-scrum.html.

[42] J. Roche. «Artefactos Scrum: las 3 herramientas clave de gestión.» (n.d.), dirección: https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html.

[43] U. de Valladolid. «Guía Docente del Trabajo de Fin de Grado. Mención Tecnologías de la Información.» (2023), dirección: https://apps.stic.uva.es/guias_docentes/uploads/2023/545/46977/1/Documento.pdf.

[44] W. API. «Word Error Rate (WER).» (n.d.), dirección: https://v17.angular.io/docs.

[45] J. M. Aguilar. «¿Qué es el patrón MVC en programación y por qué es útil?» (2019), dirección: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx.

[46] M. Gechev. «Introducing Angular v17.» (2023), dirección: https://blog.angular.dev/introducing-angular-v17-4d7033312e4b.

[47] A. University. «Angular Standalone Components: Complete Guide.» (2024), dirección: https://blog.angular-university.io/angular-standalone-components/.

[48] M. Rolfo. «Standalone components en Angular.» (2022), dirección: https://codigoencasa.com/standalone-components-en-angular/.

[49] «Spring Boot.» (n.d.), dirección: https://spring.io/projects/spring-boot.

[50] U. Ruelas. «¿Qué es la programación orientada a aspectos (POA)?» (2017), dirección: https://codingornot.com/que-es-la-programacion-orientada-a-aspectos-aop.

[51] «MySQL Workbench.» (n.d.), dirección: https://www.mysql.com/products/workbench/.

[52] «Apache Maven.» (n.d.), dirección: https://maven.apache.org/.

[53] H. Boutemy. «Introduction to the Build Lifecycle.» (2023), dirección: https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html.

[54] C. Á. Caules. «¿Que es un Maven LifeCycle y como funciona?» (2020), dirección: https://www.arquitecturajava.com/que-es-un-maven-lifecycle-y-como-funciona/.

[55] J. M. Alarcón. «Java: ¿Qué es Maven? ¿Qué es el archivo pom.xml?» (2022), dirección: https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx.

[56] «OpenAi-GPT3-Java.» (n.d.), dirección: https://github.com/TheoKanning/openai-java.

[57] «sWhisper Model Large-v2.» (n.d.), dirección: https://dataloop.ai/library/model/openai_whisper-large-v2/.

[58] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey e I. Sutskever. «Robust Speech Recognition via Large-Scale Weak Supervision.» (2022), dirección: https://arxiv.org/abs/2212.04356.

[59] S. Gandhi, M. Hollemans, M. Khalusova y V. Srivastav. «Introducción a los datos de audio.» (2023), dirección: https://huggingface.co/learn/audio-course/es/chapter1/audio_data.

[60] D. Hendrycks y K. Gimpel. «GAUSSIAN ERROR LINEAR UNITS (GELUS).» (2023), dirección: https://arxiv.org/abs/1606.08415.

[61] L. Vaughn. «Whisper.» (2024), dirección: https://github.com/openai/whisper/blob/main/README.md.

[62] «Angular.» (n.d.), dirección: https://whisperapi.com/word-error-rate-wer.

[63] «Common Voice 15.» (n.d.), dirección: https://commonvoice.mozilla.org/es.

[64] A. Conneau, M. Ma, S. Khanuja et al. «FLEURS: Few-shot Learning Evaluation of Universal Representations of Speech.» (2022), dirección: https://arxiv.org/abs/2205.12446.

[65] «Whisper.» (n.d.), dirección: https://github.com/openai/whisper/blob/main/README.md.

[66] «Eclipse Foundation.» (n.d.), dirección: https://www.eclipse.org/.

[67] «Visual Studio Code.» (n.d.), dirección: https://code.visualstudio.com/.

[68] «Overleaf.» (n.d.), dirección: https://es.overleaf.com/.

[69] «Astah.» (n.d.), dirección: https://astah.net/.

[70] «Postman.» (n.d.), dirección: https://www.postman.com/.

[71] «OpenAPI.» (n.d.), dirección: https://www.openapis.org/.

[72] P. M. Stahl. «Lingua.» (n.d.), dirección: https://github.com/pemistahl/lingua.

[73] J. A. V. Dávila. «Ambientes de Desarrollo de Software Basado en Componentes.» (2009), dirección: https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/399/1/ZACTE2.pdf.

[74] V. M. A. Angel. «Arquitectura de Componentes.» (2024), dirección: `https://vanessamarely.medium.com/arquitectura-de-componentes-e48816925d51`.