



Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE GRADO

Identificación de aves a partir del sonido

Realizado por
Jesús Vena Campos

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

Dirigido por
Francisco Luna Perejón

Realizado en el departamento de
Arquitectura y Tecnología de Computadores

Agradecimientos

En primer lugar me gustaría agradecer a mi familia, a mi padre, madre y hermano por todo lo que me han apoyado, todo lo que han hecho por mi y sus innumerables consejos, tanto a lo largo de mi vida como en estos últimos cuatro años de carrera, a mi madre por siempre estar ahí y a mi padre por ser quien me inició en este mundo de la informática desde chico hasta hoy.

Agradecer a D. Francisco Luna Perejón por su fenomenal tutoría a lo largo del desarrollo del proyecto y haber conocido el gran profesional que es.

A Encarna, por sus buenos consejos y ser para mi un ejemplo de superación en el día a día.

Y por supuesto, a mi Anita, mi pareja, por ser quien me ha aguantado en tantos momentos de agobio durante estos cuatro años, tantos momentos juntos en la biblioteca y por ser quien siempre ha estado conmigo tanto en bueno como en los malos momento.

Resumen

Este proyecto ha sido desarrollado con la finalidad de obtener un sistema que permita la identificación de aves a partir del canto de estas, este sistema está dirigido tanto para aquellos investigadores de los ecosistemas, ornitólogos como para todos aquellos curiosos de la naturaleza y en particular de las aves. Este sistema permitirá a todos estos tipos de usuarios mencionados anteriormente a identificar aves a partir del sonido que estas emiten en tiempo real y de una manera sencilla gracias a un modelo de inteligencia artificial.

Este sistema como acabamos de mencionar funciona gracias a la inteligencia artificial que es la encargada en este caso del procesado del audio de las aves y de identificarlas, además el sistema incorpora mecanismos automáticos para que se pueda grabar el audio del micrófono de la máquina del usuario y se pase de manera automática al programa de inteligencia artificial, además del posterior guardado de los datos para que el usuario acceda a estos cuando y donde quiera.

De este modo contamos con un sistema interno que tras accionar un botón por parte del usuario se encarga automáticamente y en segundo plano de capturar el audio del micrófono, pasárselo al programa donde la inteligencia artificial se encargará de identificar que avé es la que canta, y por último el procesado de esos datos y el guardado de ellos en una base de datos para hacer persistente esos datos y que no se pierdan.

Además el sistema cuenta con una interfaz gráfica vía aplicación web que permite que el usuario tenga una experiencia más agradable con todo el sistema y la visualización de los datos se haga de una manera clara y ordenada.

Los datos mostrados en el sistema nos permiten ver en formato tabla los análisis realizados por el programa de inteligencia artificial junto a la fecha y hora de estos, además contaremos con la opción de ver varias estadísticas y datos en formato de gráfica dinámica, las cuales cambian a medida que se realizan nuevos análisis y se obtienen nuevos datos.

En este documento se va a detallar todo el proceso en el que ha estado inmerso el proyecto, comenzando con una primera parte de introducción, donde conceptualizaremos el proyecto además de indicar los objetivos de este. Además se detallará el estado del arte y otros puntos importantes como la elicitation de requisitos, donde se muestran todos los requisitos previos que debían satisfacerse para completar el proyecto.

Tras eso encontraremos la sección de la ejecución del proyecto donde se encuentran desarrolladas todas las decisiones tanto del diseño como de la implementación del sistema. Finalmente se detalla la planificación del proyecto, las conclusiones y la bibliografía.

Abstract

In the following work, a system has been developed, which feeds on an artificial intelligence model, this artificial intelligence model, AI from now on, is a machine learning model through neural networks, this system will allow us to identify birds through their sound, first we will study how this model works and treats the data, later we will implement a way to transmit the sound perceived by a microphone of our machine to the AI model which will use that data for the subsequent identification of birds, after that, we will store the data in a Database, we will also provide the system with methods that allow us to execute the program in a loop for a certain time, So the user can launch the program without having to run it repeatedly. We also have a web application that will show us the system data and also allows us to run the program from the interface of the web application.

Keywords: AI = Artificial Intelligence

Índice general

| | |
|---|-----------|
| 1. Descripción del proyecto | 1 |
| 1.1. Introducción y motivación | 1 |
| 1.2. Conceptualización | 1 |
| 1.2.1. Antecedentes | 1 |
| 1.3. Objetivos del proyecto | 2 |
| 1.4. Estado del arte | 4 |
| 1.4.1. Introducción | 4 |
| 1.4.2. Primer estudio del problema | 4 |
| 1.4.3. Reconocimiento y clasificación de los sonidos. | 4 |
| 1.4.4. Inteligencia artificial y algoritmos | 5 |
| 1.4.5. Tipos de algoritmos | 6 |
| 1.4.6. Soluciones existentes y posibles. | 8 |
| 1.5. Elicitación de requisitos | 9 |
| 1.6. Alcance | 24 |
| 1.7. Metodología | 24 |
| 1.8. Presupuesto | 24 |
| 1.9. Acta de Constitución | 24 |
| 1.10. Conclusiones | 24 |
| 2. Ejecución del proyecto | 26 |
| 2.1. Introducción | 26 |
| 2.2. Diseño del sistema | 26 |
| 2.2.1. Aplicación web | 27 |
| 2.2.2. Base de datos | 29 |
| 2.2.3. Análisis y clasificación del audio | 29 |
| 2.2.4. Ejecución autónoma del sistema | 30 |
| 2.3. Implementación | 30 |
| 2.3.1. Introducción | 30 |
| 2.3.2. Tecnologías Aplicadas | 30 |
| 2.3.3. Desarrollo | 35 |
| 2.4. Pruebas del sistema | 59 |
| 3. Planificación del proyecto | 62 |
| 4. Conclusiones | 70 |
| 5. Referencias | 72 |

Índice de figuras

| | | |
|-------|---|----|
| 1.1. | Adjudicación del proyecto | 25 |
| 2.1. | Esquema general del diseño del sistema | 28 |
| 2.2. | Logotipo del editor Visual Studio Code | 31 |
| 2.3. | Logotipo del repositorio del modelo de inteligencia artificial. | 31 |
| 2.4. | Logotipo del lenguaje de programación python | 31 |
| 2.5. | Logotipo de la librería tensorflow para inteligencia artificial. | 32 |
| 2.6. | Logotipo del lenguaje de programación javascript | 32 |
| 2.7. | Logotipo del gestor de base de datos Sqlite3 | 33 |
| 2.8. | Logotipo de Node js | 33 |
| 2.9. | Logotipo del Framework Angular | 33 |
| 2.10. | Diagrama de la estructura de un proyecto Angular | 34 |
| 2.11. | Jerarquía de ficheros del modelo extraído de GitHub | 36 |
| 2.12. | Resultado de la ejecución del modelo de inteligencia artificial | 37 |
| 2.13. | Resultado del fichero tras la ejecución | 37 |
| 2.14. | Base de datos desde el visor de base de datos | 42 |
| 2.15. | Tabla de registros de los resultados | 42 |
| 2.16. | Diagrama del código que permite la ejecución iterativa del sistema | 44 |
| 2.17. | Diagrama de los módulos y del flujo del sistema | 45 |
| 2.18. | Modificaciones del package.json | 46 |
| 2.19. | Jerarquía inicial de carpetas de un proyecto angular | 49 |
| 2.20. | Pantalla principal de la aplicación | 50 |
| 2.21. | Vista móvil de una pantalla de a aplicación | 51 |
| 2.22. | Pantalla correspondiente al acceso a las tablas de datos. | 52 |
| 2.23. | Pantalla correspondiente a la tabla de aves del sistema. | 52 |
| 2.24. | Pantalla una vez iniciada la espera | 54 |
| 2.25. | Tarjeta emergente con la información del ave | 54 |
| 2.26. | Contenido mostrado por el primer enlace de la ventana emergente | 55 |
| 2.27. | Contenido mostrado por el segundo enlace de la ventana emergente | 55 |
| 2.28. | Página correspondiente a la tabla de registros del sistema | 56 |
| 2.29. | Ventana emergente con aviso tras pulsar el botón de eliminar | 56 |
| 2.30. | Pantalla de enlace a las gráficas | 57 |
| 2.31. | Pantalla con la gráfica resumen de los registros | 57 |
| 2.32. | Pantalla con las aves identificadas y el número total | 58 |
| 2.33. | Pantalla con los enlaces a la colección de postman y a la memoria del proyecto. | 58 |
| 3.1. | Días durante los cuales no se trabajará en el proyecto | 63 |
| 3.2. | Días durante los cuales no se trabajará en el proyecto 2 | 63 |
| 3.3. | Tareas del proyecto | 64 |
| 3.4. | Mes de ejemplo con los días que no se trabajará en el proyecto resaltados | 64 |
| 3.5. | Diagrama de gant con las tareas del proyecto | 65 |

| | |
|--|----|
| 3.6. Tareas del proyecto | 66 |
| 3.7. Tareas del proyecto 2 | 67 |
| 3.8. Tareas del proyecto 3 | 67 |
| 3.9. Estadísticas generales del proyecto antes de comenzar | 68 |
| 3.10. Diagrama y tareas con la realización del proyecto iniciada | 68 |
| 3.11. Estadísticas mientras se realiza el proyecto | 69 |

Índice de extractos de código

1. Descripción del proyecto

1.1. Introducción y motivación

Hoy en día vivimos en un mundo en el que tenemos acceso a gran cantidad de información en la palma de nuestra mano y de manera casi instantánea, aunque tengamos acceso a una gran cantidad de información, mucha de ella necesita un tratamiento y una posterior comprensión de la misma, la mayoría de esta información la obtenemos del día a día, en forma de texto, imágenes o sonidos, aunque mucha de esta información pasa desapercibida o simplemente no es suficiente la información como para dotar de valor por sí misma, no es menos el caso en el que estamos estudiando, trabajando o simplemente dando un paseo y escuchamos el canto de un ave, entonces, en ese caso, si quisiéramos saber que ave se identifica con ese canto, ¿nos dedicamos a buscar en bases de datos de miles de aves sus sonidos hasta encontrar alguno que se nos parezca al que habíamos escuchado?, pues evidentemente este trabajo sería muy costoso además de tener una probabilidad de encontrar una coincidencia de sonidos demasiado baja, contando que en el mundo se encuentran alrededor de unas dieciocho mil especies de aves diferentes y hasta unas 622 especies diferentes en España. De este modo, surgió este proyecto, el cuál fue propuesto por D. Francisco Luna Perejón, tutor del proyecto, donde encontramos la motivación de que la información que reciba un usuario, en este caso el canto de un ave, sea tratada y comprendida por una máquina, identificando así que ave corresponde con ese canto, facilitando así la tarea del usuario, ahorrándole tiempo y trabajo, así como por la parte que más me incumbe, el estudio de un modelo de inteligencia artificial que se encarga de la identificación de aves, e incorporar nuevas funcionalidades que hagan este modelo de detección de aves un sistema totalmente usable por un usuario final, aprendiendo así por mi parte, nuevas tecnologías y mejorando mis destrezas en distintos lenguajes de programación.

Este proyecto comenzó cuando a principios de este curso 2023-2024 tuve mi primera tutoría del trabajo con D. Francisco Luna Perejón, tutor del trabajo, que tras estar indagando en mis preferencias sobre el desarrollo y gustos en torno a la informática, decidió proponerme este tema sobre el que realizar mi proyecto, tras la comprensión por mi parte del proyecto, los objetivos de este y los aprendizajes que conllevaría el desarrollo del proyecto, no dudé en aceptar la propuesta de mi tutor y ponerme manos a la obra con el proyecto.

1.2. Conceptualización

1.2.1. Antecedentes

En este proyecto, los **clientes** serán todos aquellos usuarios a los cuales les inquieta algo la naturaleza, como sería el caso de los investigadores, o de los propios ornitólo-

gos, o del usuario que le gusta la naturaleza y disfruta de ella en sus ratos libres. Incluso aquellos usuarios curiosos por la tecnología. Como **patrocinadores** de este proyecto contamos con la propia Universidad de Sevilla, la Escuela Técnica Superior de Ingeniería informática, y el departamento de arquitectura y tecnología de computadores.

El jefe de proyecto será el alumno Jesús Vena Campos, encargado como tal tanto de la organización como del desarrollo en su plenitud de este trabajo, cuyo objetivo final será elaborar un sistema que permita la identificación de aves a partir del sonido.

Entre los interesados, encontramos desde el propio departamento, jefe de proyecto, patrocinadores, hasta los futuros usuarios de la aplicación.

Este proyecto será llevado a cabo con un mínimo de 300 horas, debido a que esto es lo equivalente a 12 créditos ECTS, este cómputo de horas ha sido establecido por la Universidad de Sevilla.

1.3. Objetivos del proyecto

El proyecto en sí se compone de dos grandes y diferenciados bloques, por una parte, encontramos el objetivo de la identificación de aves partiendo de un modelo de inteligencia artificial previamente entrenado, y toda la parte que a ello engloba como el envío de datos y el almacenamiento y tratamiento de los mismos. Por otra parte, nos encontramos con el objetivo de desarrollar una aplicación web que permita al usuario utilizar el sistema de una forma cómoda y agradable. A continuación, listaremos los objetivos:

- Estudio del modelo de inteligencia artificial.
 - Comprender como se produce el envío de datos para el análisis.
 - Comprender como realiza el tratamiento de los datos y como los muestra.
 - Extracción de los resultados de los análisis realizados por el modelo.
- Desarrollar un script que nos permita obtener el sonido de un micrófono del sistema y el posterior guardado de este sonido, de este modo podremos capturar los datos en tiempo real.
- Almacenamiento de los datos extraídos de los análisis del modelo en una base de datos.
 - Almacenar una tabla con los datos de los análisis e incluir la fecha y hora de estos.
 - Almacenar en una tabla los datos de las aves identificadas por el sistema a lo largo de toda la vida útil de este, sin incluir duplicaciones.
- Desarrollar un script que nos permita el guardado de los datos de los análisis en base de datos, este se encargará de las siguientes tareas:

- Lectura del fichero donde se encuentran los datos provenientes del análisis del audio capturado.
- Tratamiento de estos datos.
- Guardado de estos datos en base de datos.
- Eliminación del fichero que cuenta con los datos de los análisis para evitar la duplicación de datos en base de datos.
- Desarrollo de unos métodos que nos proporcionen la funcionalidad de poder ejecutar el sistema y los procesos que se incluyen en este, en segundo plano, y de forma iterativa. Indicando así un periodo de tiempo durante el que se encontraran en ejecución. Encontraremos tres procesos a ejecutar:
 - Ejecución del fichero que nos permita la grabación del audio y el guardado de este audio.
 - Ejecución del modelo, que identifique el audio capturado anteriormente.
 - Ejecución de un fichero que nos permita el guardado de los datos del análisis en base de datos.
- Creación de una aplicación web, que cuente con:
 - Un backEnd que cuente con una API que nos proporcione acceso a la base de datos y podamos obtener los datos de ésta.
 - Un frontEnd sencillo que nos muestre los datos de la base de datos.
 - La posibilidad de lanzar la ejecución del sistema desde la interfaz gráfica de modo que sea cómo y sencillo para el usuario usar el sistema.
- Realizar pruebas de todos los componentes del sistema de modo que garantizemos su correcto funcionamiento.
- Realizar una documentación exhaustiva del sistema.

A continuación listaremos los objetivos educacionales del proyecto:

- Estudio de las bases de modelos de Inteligencia Artificial para el reconocimiento del sonido.
- Estudio de las diferentes tecnologías a usar.
- Ampliación del conocimiento sobre los diferentes lenguajes de programación usados y resolución de nuevos retos de programación.
- Descubrimiento de nuevas tecnologías no usadas anteriormente, como el framework y el gestor de base de dato.

1.4. Estado del arte

1.4.1. Introducción

A continuación, se detallará los puntos principales abordados en el estudio del problema tras la elección del tema principal del trabajo, abordando desde el reconocimiento de sonidos, sistemas de detección de sonidos existentes, y finalizando con la exposición de nociones sobre inteligencia artificial que son de interés para poner en contexto el proyecto.

1.4.2. Primer estudio del problema

Tras la adjudicación del proyecto, las primeras tareas consistieron en indagar sobre la temática. En primer lugar, comencé realizando un estudio en profundidad sobre el tema planteado, posibles soluciones a abordar.

Este desarrollo comienza con un estudio sobre la identificación y clasificación de sonidos, mucho antes que la identificación de aves, ya que primero debíamos investigar las formas actuales mediante las cuales podemos identificar sonidos. Aunque el estudio de este proyecto no comenzaba desde cero, gracias a los conocimientos sobre diferentes tecnologías, algoritmos y el ámbito de la informática que he conseguido adquirir a lo largo de estos cuatro años de formación universitaria, gracias a asignaturas, profesores o investigación propia.

En primer lugar, comencé pensando cuál era el bloque principal que debía ser desarrollado en función a los requisitos del sistema, y era evidente que lo primero que el proyecto requería era el sistema que reconociese el audio del ave y nos dijera qué aves es.

De este modo el estudio comenzó con una búsqueda exhaustiva de cómo se podía identificar un ave a partir de su sonido, a parte de las ideas que previamente pudiera tener tras mis propios conocimientos.

1.4.3. Reconocimiento y clasificación de los sonidos.

Como vimos en el anterior apartado, nuestro primer problema con el que nos topamos es la identificación de los sonidos, de modo que realizaremos un estudio acerca de cómo se realiza esto hoy en día.

Por ello, para la identificación de sonidos pensé en varios modos, uno de ellos fue pensado con la idea de almacenar miles de sonidos de aves de modo que el sonido que más se pareciese al sonido que queremos detectar, sería el que el sistema nos indicaría como encontrado, pero mientras la misma idea se iba ocurriendo, era obvia la inefficiencia de esto y lo tedioso que resultaría el proceso de la búsqueda de miles de sonidos de aves y la previa clasificación de estos.

Por otra parte, tras una búsqueda exhaustiva obtuvimos el resultado de que en internet encontramos poca información sobre la identificación de sonidos, nada iba más lejos de explicar que la mayoría de software's de reconocimiento, tanto de imágenes, vídeo o sonidos están formados internamente por algoritmos de aprendizaje automático, redes neuronales y demás. Todo esto a lo que en el mercado se conoce bajo el término "Inteligencia Artificial". De este modo la investigación empezó a tomar una bifurcación desde el inicio hacia una búsqueda más centrada en el ámbito de la inteligencia artificial y sobre todo en el análisis de sonidos.

Hasta aquí podemos considerar nuestro primer estudio del problema, del cual obtenemos la conclusión de que para la detección de sonidos o identificación de éstos usaremos inteligencia artificial, pero no sabemos ni que tipos de algoritmos, si los desarrollaremos nosotros o podremos nutrirnos de algún modelo ya entrenado, si todo esto es viable, y muchísimas cuestiones que irán surgiendo a lo largo del estudio.

1.4.4. Inteligencia artificial y algoritmos

Tras llegar a la conclusión en el apartado anterior que el mejor modo de proceder a la identificación de sonidos era a partir de lo que hoy en día se conoce como inteligencia artificial, de modo que en esta sección procederemos a estudiar este concepto de inteligencia artificial y los diferentes algoritmos que pueden ser usados para obtener cierto resultado.

En primer lugar, comenzamos definiendo que es la inteligencia artificial en sí. Bajo mi punto de vista y basándome en mis conocimientos, la inteligencia artificial no es una máquina inteligente como podría pensar cualquiera, sino que es una selección lo más parecida posible a la que haría el cerebro humano pero basándose en técnicas y algoritmos de modo que la máquina pueda decidir en función de estos parámetros, por ejemplo, en el caso de que una máquina sea capaz de predecir que objeto es el que estamos apuntando con la cámara del móvil, pero realmente la diferencia con la propia inteligencia es que la máquina no entiende nada, ya que una máquina a diferencia de un ser humano no es consciente de lo que está realizando, de modo que nunca será capaz de aprender tal y como conocemos este concepto, si no que los resultados que nos devuelva puedan ser mejores a lo largo que el algoritmo es entrenando y este es eficiente.

Tras tener una idea base sobre esto, procedí a buscar en diversas fuentes en internet sobre inteligencia artificial, encontrando diversas definiciones de inteligencia artificial un resumen a muy grandes rasgos de la definición que podemos encontrar a continuación:

"La inteligencia artificial es la capacidad o habilidad de una máquina para desempeñar tareas parecidas a las realizadas por el cerebro humano como pueden ser identificar cosas, razonamientos, etc."

De modo que obtenemos que la inteligencia artificial, se desarrolla a partir de una serie de algoritmos, y a continuación definimos que es un algoritmo:

"Un algoritmo no es más que un conjunto de fases, etapas o procedimientos que hay que seguir para resolver un problema.", evidentemente, respecto a la complejidad de los algoritmos encontramos un abanico enorme, desde algoritmo muy sencillo que

todos conocemos como puede ser los pasos que realizamos para ducharnos, hasta algoritmos muy complejos como pueden ser aquellos que se encargan de calcular la trayectoria orbital de un cohete cuando sale de la tierra.

Además de tener complejidades diferentes, aunque los algoritmos tengan una misma finalidad, por ejemplo tomar un ducha, los pasos de estos algoritmos pueden cambiar, podemos verlo en el siguiente ejemplo:

Tenemos dos usuarios de la ducha, cuya finalidad es a misma, tomar una ducha, en el caso del usuario x, sigue los pasos:

- 1.Desnudarse.
- 2.Encender la ducha.
- 3.Se humedece la piel previamente con agua.
- 4.Enjabonado del cuerpo y cabello con jabón.
- 5.Aclarado.
- 6.Secado

Sin embargo, el algoritmo que usa el usuario y para tomar una ducha es el siguiente:

- 1.Encender el termo.
- 2.Comprobar que hay agua caliente.
- 3.Desnudarse.
- 4.Enciende la ducha.
- 5.Se humedece la piel previamente con agua.
- 6.Enjabonado de la piel con jabón.
- 7.Enjabonado del cabello con un champú especial.
- 8.Aclarado
- 9.Secado

Como podemos ver ambos algoritmos tienen la misma finalidad, tomar una ducha, y estos algoritmos tienen algunos pasos iguales, y otros diferentes, esto no indica que uno sea mejor que otro, sino que cada uno es bueno dependiendo de la situación en la que no encontrremos, los recursos de los que dispongamos y muchas otras variables que tenemos que tener en cuenta en función del problema que estemos abordando.

1.4.5. Tipos de algoritmos

Introducción

Al igual que otras disciplinas, existen numerosas clasificaciones distintas de los tipos de algoritmos que podemos encontrarnos en la actualidad, y citar todas estas clasificaciones de algoritmos sería un tanto engorroso debido a que muchas de las clasifica-

ciones existentes podemos encontrar mismos algoritmos con diferentes nombres y esto podría causarnos confusión, de este modo detallaremos una clasificación aprendida a lo largo de los cursos de la universidad.

Clasificación de algoritmos

A continuación listaremos los diferentes tipos de algoritmos y sus principales características:

- **Algoritmos iterativos:** Estos algoritmos son aquellos que se ejecutan en un bucle, es decir, se realizan varias ejecuciones de un mismo código, estos algoritmos suelen ser usados para realizar tareas repetitivas. El ejemplo más claro de un algoritmo iterativo es aquel que se encuentra por ejemplo, en un bucle hasta que alcanza un valor esperado y sale del bucle y termina el algoritmo.
- **Algoritmos recursivos:** Estos algoritmos a diferencia de los iterativos no se ejecutan en bucle hasta que una condición determina que debe de terminar la ejecución, si no que estos en función a unas condiciones realizar llamadas a ellos mismos comenzando de nuevo la ejecución de este algoritmo desde su mismo cuerpo.

Definimos entonces dos grandes grupos en los que podemos clasificar los diferentes tipos de algoritmos, queda por recalcar que la mayoría de los algoritmos iterativos pueden ser traducidos a algoritmos recursivos y viceversa.

Dentro de la inteligencia artificial encontramos gran cantidad de ramas, a continuación mencionaremos tres de los principales tipos:

- **Inteligencia artificial basada en reglas:** se definirán un conjunto de reglas a las que debe atender el programa y funcionar en base a estas, obteniendo un resultado a partir de la aplicación de las reglas.
- **Aprendizaje automático:** también conocido como **Machine Learning** se pretende que se mejore la ejecución y los resultados a medida que va pasando el tiempo y la recopilación de datos es mayor. Dentro de esta rama encontramos un grupo distinguido por su peso en el mercado actual, estas son las **Redes Neuronales** como su propio nombre indica esta basada en las redes neuronales, en el funcionamiento del cerebro humano, donde encontraremos varios nodos interconectados, que irán produciendo resultados, hasta completar el camino completo de nodos y obtener el "mejor" resultado.

Tras un estudio sobre el reconocimiento y clasificación de sonidos y los tipos de inteligencia artificial hemos obtenido como resultado que en el mercado las tecnologías que realizan estas tareas suelen implementar en su interior redes neuronales o aprendizaje automático, de este modo esta información nos ayudará nuevamente a encontrar una solución más adaptada hacia nuestros requisitos.

1.4.6. Soluciones existentes y posibles.

Tras un estudio de la inteligencia artificial, una pequeña pincelada sobre los algoritmos y las principales ramas de la IA, comenzaremos buscando soluciones de inteligencia artificial existentes en el mercado, ya que si la rueda ya está inventada, no a vamos a volver a inventar, y reciclar así de este modo el trabajo realizado por algún otro usuario de la comunidad. De este modo comenzamos con una búsqueda sobre las soluciones que pudieran servirnos para nuestro proyecto, soluciones que nos permitan la identificación de los sonidos que emiten las aves e identificar que aves son estas.

Tras una exhaustiva búsqueda, sorprendentemente hemos encontrado una cantidad considerable de páginas que nos aportaban información de cómo identificar aves a partir de su canto, en cambio, pese a parecer soluciones válidas, la gran mayoría no nos aportan la funcionalidad necesaria para montar nuestro sistema. A continuación detallamos las soluciones encontradas en el mercado, y los detalles de si usaremos dicha solución o no, y el por qué de esto.

- <https://losenlacesdelavida.fundaciondescubre.es/proyectos/una-nueva-app-para-identificar-y-aprender-los-cantos-de-las-aves/>
Como podemos apreciar si navegamos en el enlace anterior, es una app que nos permite identificar aves a partir del sonido, pero únicamente para móvil, de modo que no podremos integrarla en el sistema.
- <https://www.lavanguardia.com/tecnologia/20211017/7783287/merlin-bird-id-aplicacion-identifica-cantos-pajaro-shazam-pmv.html> Esta aplicación podemos ver que realiza todo aquello que necesitamos, y en esa página no encontramos toda la información para saber si podríamos usarla en nuestro proyecto de modo que continuaremos buscando información sobre la misma.

De modo que encontramos la página oficial <https://birdnet.cornell.edu/> y allí encontramos información bastante valiosa sobre este sistema como que es un sistema que se encarga de la detección e identificación de los cantos de las aves, mediante aprendizaje automático. Tras ello en la página encontramos que tienen un repositorio de Github con todo el código de la aplicación, por tanto evaluaremos este código y probaremos el sistema por si puede aportarnos la funcionalidad que requerimos para el desarrollo del proyecto.

Tras la instalación del sistema, y la prueba del mismo, llegamos a la conclusión de que realiza a la perfección las tareas que nosotros necesitamos para aportar esta funcionalidad de identificar el sonido de las aves, de modo que incorporaremos el modelo realizado por The Cornell Lab, un grupo perteneciente al departamento de ornitología de la Universidad de Cornell a nuestro proyecto final dotando así de la funcionalidad de identificar las aves a partir de su canto.

Podemos apreciar en el repositorio <https://github.com/kahst/BirdNET-Analyzer> como los módulos que se encargan del análisis de los datos (analyze.py) están desarrollados en lenguaje python y nos permitirán añadirle nuevas funcionalidades.

Recalcamos que antes de tomar la decisión de que el modelo podría servirnos para el desarrollo del proyecto se realizó un estudio a fondo sobre como este modelo trataba los datos, como recibía los archivos de audio, como se ejecutaba el modelo, como

procesaba la información, como analizaba y clasificaba esta información y cómo mostraba los resultados tras los análisis, además de comprender a fondo como realizaba todos los procesos anteriormente mencionados se realizaron pruebas pertinentes de si cabía la posibilidad de acoplar este modelo de inteligencia artificial con otros ficheros que nos permitan realizar tareas que dependen del funcionamiento y de los datos que nos aportan los análisis del modelo, una vez que todas las pruebas resultaron exitosas, dicho modelo fue tomado como válido para ser incorporado a nuestro sistema final.

Tras ello, comenzó la investigación sobre cómo podíamos capturar el sonido que el micrófono de un ordenador percibía aportando así un valor mayor al sistema permitiendo realizar capturas de audio en tiempo real, ya que tras el estudio del modelo de inteligencia artificial anterior no proporcionaba esta funcionalidad, tras una búsqueda exhaustiva en internet encontramos varias librerías de python que nos permitían la captura del audio del micrófono, el troceado del sonido para que fuera más eficiente y el guardado del audio para su posterior uso, encontramos varias librerías como **speech-recognition**, **pyaudio**, **sounddevice**, aparte de estas librerías para captura de audio, van ligadas a estas otras librerías que se encargan del guardado en ficheros, la lectura de estos, etc. Encontramos también varias librerías, algunas de ellas son **wavio**, **wavio**, **wave**, entre muchas otras.

1.5. Elicitación de requisitos

A partir de los objetivos mencionados anteriormente en la sección 1.3, obtenemos los requisitos necesarios para dotar al sistema de la funcionalidad requerida, comenzaremos citando los tipos de requisitos que identificaremos, junto con las características principales de estos tipos de requisitos.

- Requisitos funcionales: describen qué debe hacer el software y como hacerlo. Los etiquetaremos con el acrónimo “RF”.
- Requisitos no funcionales: describen características del software, cómo debe este funcionar. Los etiquetaremos con el acrónimo “RNF”.
- Requisitos de datos: describen como debe de general, almacenar y tratar los datos el software. Los etiquetaremos con el acrónimo “RD”.
- Requisitos de interfaz: describen la interacción del software con terceros, ya bien sean usuarios u otros software's. Los etiquetaremos con el acrónimo “RI”.
- Reglas de negocio: describen características del negocio que influyen sobre el software. Los etiquetaremos con el acrónimo “RN”.

Una vez tenemos claro los diferentes tipos de requisitos, comenzaremos a clasificarlos a continuación, cada requisito contará con un identificador único, algunos de estos requisitos podrán desarrollarse y completarse independientemente de los demás, sin embargo alguno de ellos depende de otros requisitos, si un requisito depende de otro anterior, este anterior lo etiquetaremos como “bloqueante”, cabe destacar que si tenemos un requisito, por ejemplo el requisito 017, cuyo desarrollo depende de la reali-

zación del requisito 012, y el requisito 022 depende del requisito 017, el 022 depende implícitamente del requisito 012. Estableceremos también una prioridad a los diferentes requisitos en función de la importancia con la que cuenten esto. A continuación encontramos los requisitos funcionales:

| | | | |
|---------------------|---|-------|----------|
| RF-001 | Uso de un modelo de inteligencia artificial que detecte aves a partir del sonido | | |
| Descripción | Debemos usar para el sistema un modelo de inteligencia artificial, que nos permita la detección de aves a partir del sonido. El lenguaje en el que dicho modelo esté desarrollado es indiferente, mientras que sea compatible con el mayor número de dispositivos actuales. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | – | | |

| | | | |
|---------------------|--|-------|----------|
| RF-002 | Capturar el audio del micrófono de la máquina. | | |
| Descripción | Realizar un script el cuál dote al sistema de la capacidad de grabar el audio de un micrófono del sistema y realizar el guardado de este fichero de audio en formato .wav. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|-------|----------|
| RF-003 | Análisis del modelo del audio grabado por el micrófono | | |
| Descripción | El modelo de inteligencia artificial deberá realizar el análisis del audio que el programa de capturar el audio del sistema guardó posteriormente | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-002, RF-001 | | |

| | | | |
|---------------------|---|-------|----------|
| RF-004 | Guardado de los datos de los análisis en un fichero de texto. | | |
| Descripción | El modelo de inteligencia artificial deberá realizar el guardado de los datos de los análisis en un fichero de texto. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-001 | | |

| | | | |
|---------------------|---|-------|----------|
| RF-005 | Lectura de los datos obtenidos del modelo de inteligencia artificial. | | |
| Descripción | Realizar un script que permita al sistema realizar una lectura de los datos obtenidos de los análisis del modelo de inteligencia artificial, para posterior tratamiento de estos datos. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-004, RF-001 | | |

| | | | |
|----------------------|---|---------|----------|
| RF-006 | Base de datos | | |
| Descripción | El sistema deberá contar con una base de datos, de modo que podamos almacenar en ella los datos obtenidos de los análisis del modelo de inteligencia artificial. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-004 | | |
| Observaciones | Las dependencias encontradas son debido a que es necesario saber que datos serán los que tendremos disponibles, ya en función de eso decidiremos las tablas a formar. | | |

| | | | |
|---------------------|--|---------|----------|
| RF-007 | Creación de la base de datos. | | |
| Descripción | El sistema deberá de crear la base de datos de una forma autónoma. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|---------|----------|
| RF-008 | Guardado de datos en la base de datos | | |
| Descripción | El sistema deberá escribir los datos en la base de datos de forma autónoma. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|---------|----------|
| RF-009 | Borrado automático de los datos ya guardados | | |
| Descripción | El sistema deberá borrar los ficheros de datos de los análisis de forma automática una vez estos ya sean persistentes en la base de datos | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-007 | | |

| | | | |
|---------------------|---|---------|----------|
| RF-010 | El sistema deberá realizar todos los procesos de manera automática. | | |
| Descripción | El sistema deberá realizar automáticamente el siguiente flujo: 1.Captura del audio del micrófono y guardado de este. 2.Análisis del audio capturado 3.Lectura de los datos y tratamiento de estos 4.Creación de la base de datos y de las tablas (Si no existen) 5.Guardado de los datos del análisis en la base de datos 6.Borrado de los ficheros que contienen los datos de los análisis para no duplicarlos en la base de datos. El proceso deberá ser iterativo, de modo que se ejecute este flujo durante un tiempo determinado, y acabe una vez pase el tiempo indicado. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|----------------------|--|-------|----------|
| RF-011 | El sistema deberá contar con una aplicación web | | |
| Descripción | El sistema debe contar con una aplicación web completa, que permita la visualización de los datos obtenidos por el sistema. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | — | | |
| Observaciones | Este requisito se divide en requisitos más pequeños de modo que sea más fácil abordarlos. No contamos con dependencias debido a que la creación de la base de la aplicación web es independiente de los requisitos anteriores, aunque el contenido de este, sí que dependerá de los requisitos anteriores. | | |

| | | | |
|---------------------|---|-------|----------|
| RF-012 | La aplicación web deberá obtener los datos de la base de datos | | |
| Descripción | La aplicación web debe acceder a la base de datos obteniendo los datos que contenga esta para su posterior uso. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|---|-------|----------|
| RF-013 | La aplicación web deberá mostrar los datos de la base de datos | | |
| Descripción | La aplicación web debe mostrar los datos de la base de datos en alguna pantalla, en forma de tabla o gráfica. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-012, RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|---|---------|----------|
| RF-014 | La aplicación web deberá ejecutar el sistema desde la interfaz gráfica. | | |
| Descripción | La aplicación web debe permitirnos la ejecución de todo el sistema en segundo plano, de modo que este esté capturando los datos y guardándolos en base de datos y no interfiera en el uso de la aplicación. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | NO | | |
| Dependencias | RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|--|---------|----------|
| RF-015 | La aplicación web deberá mostrar información sobre los análisis realizados por el modelo | | |
| Descripción | La aplicación web debe mostrar los datos de los análisis que tenemos en base de datos, en una tabla. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-013, RF-012, RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|--|-------|------------|
| RF-016 | La aplicación web deberá permitir el borrado de los análisis | | |
| Descripción | La aplicación web debe permitir la opción de borrar cada análisis de forma individualmente. | | |
| Prioridad | Máxima | Media | Reducida ✓ |
| Bloqueante | NO | | |
| Dependencias | RF-014, RF-013, RF-012, RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|--|---------|----------|
| RF-017 | La aplicación web deberá mostrar una tabla con las aves identificadas. | | |
| Descripción | La aplicación web debe mostrar una tabla únicamente de las aves que han sido identificadas por el sistema, sin duplicado de datos. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-014, RF-013, RF-012, RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|--|---------|----------|
| RF-018 | La aplicación web deberá mostrar la información de cada ave del sistema. | | |
| Descripción | La aplicación web debe buscar información del ave que le indiquemos. | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | SI | | |
| Dependencias | RF-016, RF-014, RF-013, RF-012, RF-011, RF-010, RF-009, RF-008, RF-007, RF-006, RF-005, RF-004, RF-003, RF-002, RF-001 | | |

| | | | |
|---------------------|---|-------|----------|
| RNF-001 | Funcionamiento autónomo del sistema | | |
| Descripción | El sistema deberá realizar todos los flujos del sistema de manera totalmente autónoma, requiriendo una única interacción del usuario para lanzar la ejecución del sistema, aportando así sencillez y un grado de usabilidad mayor | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|--|-------|----------|
| RNF-002 | Ausencia de errores | | |
| Descripción | El sistema en su plenitud no debe mostrar ningún tipo de error, controlando así todas las posibles excepciones que puedan producirse | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|-------|----------|
| RNF-003 | Compatibilidad de los módulos | | |
| Descripción | Todos los módulos por desarrollar para la aplicación deben de ser totalmente compatibles, de modo que no haya conflictos entre ellos. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|--|-------|----------|
| RNF-004 | Eficiencia de los módulos | | |
| Descripción | Todos los módulos del sistema deberán ser eficientes, en la medida que el número de operaciones bloqueantes sea el mínimo posible, aportando así valor a la usabilidad y eficiencia. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|-------|----------|
| RNF-005 | Ejecución del sistema | | |
| Descripción | La ejecución del sistema deberá realizarse de forma iterativa, por un tiempo máximo de 60 segundos, de modo que una vez que concluyan se para el proceso. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|-------|----------|
| RNF-006 | Usabilidad de la aplicación web | | |
| Descripción | La aplicación web debe de contar con una interfaz web usable y sencilla para cualquier usuario, de modo que la experiencia de este sea totalmente gratificante y le incite a pasar más tiempo en la aplicación. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|-------|----------|
| RNF-007 | Accesibilidad de la aplicación web | | |
| Descripción | La accesibilidad de la aplicación web debe ser elevada, facilitando así a los usuarios a la navegación entre las diferentes páginas de manera sencilla e intuitiva. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |

| | | | |
|---------------------|---|-------|----------|
| RD-001 | Datos extraídos de los análisis del modelo de inteligencia artificial. | | |
| Descripción | El sistema debe poder acceder a los datos que el modelo devuelve tras analizar el fichero de audio. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | — | | |
| Objetivos | Poder realizar posteriormente el tratamiento de estos datos. | | |

| | | | |
|---------------------|--|-------|----------|
| RD-002 | Lectura y tratamiento de los datos extraídos de los análisis, | | |
| Descripción | El sistema debe poder realizar un tratamiento de los datos obtenidos de los análisis, de modo que obtengamos los datos más relevantes. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RD-001 | | |
| Objetivos | Poder realizar posteriormente el almacenamiento de estos datos. | | |

| | | | |
|---------------------|--|-------|----------|
| RD-003 | Almacenamiento de los datos en base de datos. | | |
| Descripción | El sistema debe poder guardar los datos ya tratados en base de datos | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RD-002 | | |
| Objetivos | Poder acceder a ellos en base de datos. | | |

| | | | |
|---------------------|---|-------|----------|
| RD-004 | Tabla de datos | | |
| Descripción | El sistema debe poder guardar en la base de datos una tabla que cuente con los datos de los análisis realizados con las siguientes columnas: 1.Identificador único del registro. 2.Tiempo de inicio del análisis. 3.Tiempo de finalización del análisis. 4.Código de especie. 5.Nombre común del ave. 6.Confianza. 7.Fecha y hora del análisis. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RD-002 | | |
| Objetivos | Poder acceder a ellos en base de datos. | | |

| | | | |
|---------------------|--|-------|----------|
| RD-005 | Tabla de aves | | |
| Descripción | El sistema debe poder guardar una tabla en base de datos con las diferentes aves que hemos detectado a lo largo de la vida del sistema, sin aves duplicadas. La tabla debe contar con las siguientes columnas: 1.Identificador único del registro. 2.Nombre del ave. 3.Código de la especie. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RD-002 | | |
| Objetivos | Poder acceder a ellos en base de datos. | | |

| | | | |
|---------------------|---|-------|----------|
| RD-006 | Acceso a los datos desde la aplicación web. | | |
| Descripción | La aplicación web debe poder acceder a la base de datos para obtener los datos de ambas tablas. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RD-005, RF-004, RF-002 | | |
| Objetivos | Poder acceder a la base de datos y obtener los datos para un posterior uso de los mismos. | | |

| | | | |
|---------------------|---|-------|----------|
| RD-007 | Mostrar datos de la base de datos en la aplicación web. | | |
| Descripción | Las tablas almacenadas en la base de datos deben mostrarse en la interfaz gráfica como mínimo en formato de tabla. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | RD-006 | | |
| Objetivos | Poder acceder a los datos en base de datos y mostrarlos al usuario con un formato más agradable y sencillo que las tablas de una base de datos. | | |

| | | | |
|---------------------|---|-------|----------|
| RI-001 | Interfaz sencilla y fácil de usar. | | |
| Descripción | La interfaz de la aplicación web debe ser sencilla y fácil de usar, aportándole valor al usuario y comodidad durante su uso, reduciendo así además la frustración del propio usuario frente interfaces complejas. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Alcanzar la satisfacción de uso por parte del usuario final. | | |

| | | | |
|---------------------|---|-------|----------|
| RI-002 | Navegación mediante barra de navegación. | | |
| Descripción | Se deberá dotar a la interfaz con una barra de navegación que facilite la movilidad del usuario entre las diferentes páginas de la aplicación, de forma que sea intuitiva para el usuario y cómoda. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Facilitar la navegación del usuario en la aplicación. | | |

| | | | |
|---------------------|---|-------|----------|
| RI-003 | Información básica de la aplicación | | |
| Descripción | Se deberá mostrar información básica de la aplicación en la página principal, de modo que el usuario vea el propósito de la página. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Acercar al usuario e informarlo sobre la finalidad y el objetivo de esta página. | | |

| | | | |
|---------------------|---|-------|----------|
| RI-004 | Tablas | | |
| Descripción | Debemos mostrar en la interfaz una página que nos indique las tablas de los datos que podremos visualizar, en este caso tendremos dos enlaces: 1.Tabla de todos los registros. 2.Tabla de aves. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Hacer cómo la navegación del usuario, ofreciendo siempre información sobre que va a ver a continuación. | | |

| | | | |
|---------------------|---|-------|----------|
| RI-005 | Tabla de aves | | |
| Descripción | Además, incluiremos un botón que nos permita obtener información de esa ave en concreto. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Mostrar los datos de la base de datos de forma clara y organizada para facilitar así la lectura de estos datos por parte del usuario. | | |

| | | | |
|---------------------|--|-------|----------|
| RI-006 | Tabla de datos | | |
| Descripción | Mostraremos una tabla con todas las identificaciones que han sido obtenidas del modelo de inteligencia artificial a lo largo del tiempo, añadiendo la opción de poder eliminar una fila en concreto de la tabla. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Mostrar los datos de la base de datos de forma clara y organizada para facilitar así la lectura de estos datos por parte del usuario. | | |

| | | | |
|---------------------|---|---------|----------|
| RI-007 | Gráficas dinámicas | | |
| Descripción | Deberemos mostrar alguna gráfica dinámica, que cambie a medida que los datos cambian, y que muestre algunos datos interesantes de los obtenidos a partir de los análisis del modelo | | |
| Prioridad | Máxima | Media ✓ | Reducida |
| Bloqueante | NO | | |
| Dependencias | – | | |
| Objetivos | Mostrar los datos de una manera diferente, más amigable y mucho más sencilla de interpretar para el usuario. | | |

| | | | |
|---------------------|--|-------|------------|
| RI-008 | Ejecución del sistema. | | |
| Descripción | Incorporar un botón en alguna parte de la interfaz que nos permita la ejecución de todo el sistema desde la interfaz. | | |
| Prioridad | Máxima | Media | Reducida ✓ |
| Bloqueante | NO | | |
| Dependencias | — | | |
| Objetivos | Permitir que el usuario lance los procesos en segundo plano y todo el sistema realice el flujo de manera autónoma mediante el usuario navega por la aplicación o hace otra cosa, de modo que se guardaran los datos automáticamente en la base de datos y al refrescar la página por ejemplo de todos los registros estos datos se cargarán automáticamente en la tabla. | | |

| | | | |
|---------------------|---|-------|------------|
| RI-009 | Documentación | | |
| Descripción | Mostrar en alguna página la documentación realizada para este proyecto, incluyendo las diversas pruebas que hayan sido realizadas | | |
| Prioridad | Máxima | Media | Reducida ✓ |
| Bloqueante | NO | | |
| Dependencias | — | | |
| Objetivos | Mostrar el trabajo que hay detrás de un proyecto como este al usuario. | | |

| | | | |
|---------------------|--|-------|----------|
| RN-001 | Lenguaje de programación para los módulos | | |
| Descripción | El lenguaje de programación a usar en los diferentes módulos que incorporaremos al modelo de inteligencia artificial será realizado en lenguaje Python dada su alta compatibilidad con la mayoría de las máquinas, su eficiencia y el conocimiento del programador de este lenguaje. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | Es esencial. | | |
| Objetivos | Realizar los módulos (como el de grabar el audio, por ejemplo), en un lenguaje conocido por el programador, débilmente tipado y que cuenta con gran número de librerías para incorporar miles de funcionalidades. | | |

| | | | |
|---------------------|---|-------|----------|
| RN-002 | Base de datos | | |
| Descripción | La base de datos debe ser de tipo relacional. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | Es esencial. | | |

| | | | |
|---------------------|---|-------|----------|
| RN-003 | Aplicación web | | |
| Descripción | La aplicación web debe realizarse con el framework Angular | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | Es esencial. | | |
| Objetivos | Usar un framework actual, eficiente, escalable y con un gran soporte tras él. | | |

| | | | |
|---------------------|---|-------|----------|
| RN-004 | API REST para la aplicación | | |
| Descripción | La aplicación web debe contar con su propia api desarrollada con lenguaje JavaScript, usando node.js y express. | | |
| Prioridad | Máxima ✓ | Media | Reducida |
| Bloqueante | SI | | |
| Dependencias | Es esencial. | | |
| Objetivos | Uso de tecnologías actuales bien documentadas y con un buen soporte. | | |

1.6. Alcance

El alcance del proyecto vendrá determinado por los siguientes apartados:

- Interacción del usuario con el sistema de manera cómoda y simple.
- Completar todas las funcionalidades requeridas en los posteriores requisitos.
- Sistema totalmente funcional y probado.
- Desarrollo de una memoria del proyecto altamente documentada y justificada.
- Haber completado y justificado un mínimo de 300 horas de trabajo.

1.7. Metodología

Este proyecto será llevado a cabo con una metodología en cascada, o modelo de cascada, dicha metodología se trata de un proceso en el que realizaremos el trabajo de una manera escalonada, en la que contaremos con distintas fases. En esta metodología, el comienzo de una nueva fase está sujeto a la finalización de la fase anterior, de modo que se establecerá una fecha límite para la finalización del desarrollo de la fase en curso; de este modo, podremos tener una planificación mejor elaborada del proyecto, pudiendo ser modificada en función de las necesidades del proyecto si se diera el caso.

1.8. Presupuesto

Para el presupuesto de este proyecto contaremos con los recursos ofrecidos por el departamento de Arquitectura y Tecnología de Computadores.

1.9. Acta de Constitución

En la figura 1.1 podemos ver la adjudicación del proyecto :

1.10. Conclusiones

Finalmente, una vez han sido los objetivos del proyecto detallados, es crucial establecer una metodología de trabajo y una planificación para que el proyecto sea llevado a cabo con éxito, y una vez hemos detallado todo esto, podemos continuar con la siguiente fase del proyecto.



E.T.S. INGENIERÍA INFORMÁTICA

Asignatura Trabajo Fin de Grado

ADJUDICACIÓN

Datos del trabajo:

| | |
|--------------|---|
| Título: | Identificación de aves a partir de audio |
| Descripción: | Se propone la creación de un sistema que, a partir de un algoritmo de clasificación basado en Inteligencia Artificial, permita la identificación de distintas aves a partir del sonido que emiten. Complementariamente, el sistema puede estar dotado de la capacidad de actuar sobre el entorno tras identificar el ave. |
| Tutor/a: | Luna Perejón, Francisco |
| Cotutor/a: | Muñoz Saavedra, Luis |

Estudiantes:

| Apellidos, Nombre | Identificación | Titulación |
|--------------------|----------------|--|
| VENA CAMPOS, JESUS | 49528515Q | Grado en Ingeniería Informática - Tecnologías Informáticas |

Fecha: 01-10-2022

Figura 1.1: Adjudicación del proyecto

2. Ejecución del proyecto

2.1. Introducción

En este capítulo detallaremos en un primer lugar como ha sido el diseño del sistema y los diferentes módulos que componen a este, en segundo lugar se detallará toda la implementación del sistema donde se detallarán las tecnologías usadas y el desarrollo del sistema entre otras cosas.

2.2. Diseño del sistema

Este sistema podemos estructurarlo en dos grandes bloques, a continuación indicaremos los dos bloques que componen el sistema y listaremos a su vez los elementos con los que cuenta cada bloque:

- Bloque 1: Sistema de reconocimiento de aves. Este bloque se compone principalmente por el modelo de inteligencia artificial y todo lo que a ello engloba para reconocer aves y registrarlas en el sistema:
 - Modelo de inteligencia artificial
 - Script que ejecuta de manera iterativa y en segundo plano todo el proceso.
 - Script que se encarga de la captura del audio de un micrófono y el guardado de este audio.
 - Script que se encarga de la lectura de los datos resultantes de los análisis y el guardado de estos datos en base de datos, además de la creación de la base de datos y la inicialización de las tablas si algo no existiera.
 - Una base de datos relacional, diseñada en sqlite3.
- Bloque 2: Aplicación Web. En este bloque encontramos todo lo relativo a la aplicación web, que se nutre de la información recopilada por el bloque 1, para dar servicio al usuario en la consulta de información sobre las aves identificadas.
 - BackEnd de la aplicación, esta parte nos permite levantar un servidor. Este servidor cuenta con una api REST que nos permite poder realizar peticiones a esta desde el FrontEnd.
 - Script que se encarga de crear la conexión con la base de datos, para poder obtener los datos de la base de datos
 - FrontEnd, el cual contiene todo el contenido relativo a la parte visible de la aplicación web.

A continuación mostramos un diagrama sencillo donde podemos ver un flujo sencillo de los datos que tratará el sistema además de los principales componentes que usará.

En la figura 2.1 podemos apreciar ambos bloques mencionados en la enumeración del apartado 2.2, por una parte el modelo de inteligencia artificial y todo lo que a este engloba, y por otra parte la aplicación web, además de ver a muy grandes rasgos por las flechas del diagrama como son los principales flujos de la aplicación.

2.2.1. Aplicación web

La aplicación web en si, consta de dos partes. En primer lugar tenemos el front-end, donde se encuentra toda la parte visual de la aplicación web y donde encontraremos diversas pestañas entre las que encontraremos una página de inicio, otra con las estadísticas, otra con tablas de datos y una última de documentación. Todo esto es la parte con la que el usuario interaccionará y podrá visualizar así todos los datos que plasmemos desde el sistema.

Por otra parte, encontramos el otro bloque de la aplicación web, que es el back-end, que cuenta una API (Application Programming Interface) que nos permite realizar las conexiones pertinentes con la base de datos de modo que este servidor puede hacer peticiones a la base de datos y obtener los datos de esta, realmente la api es el intermediario entre la base de datos y el front-end. El front-end realizará peticiones http solicitando x dato de la base de datos a la api, y la api se encarga de realizar la petición correspondiente a la base de datos y enviarle estos al front-end.

En el caso de la mayoría de las funcionalidades de la aplicación, en el caso de estar trabajando en local, no requieren de conexión a la red, excepto una funcionalidad que requiere de conexión para hacer consultas a través de la red, esta es la búsqueda de información de un ave en concreto. También se ha dispuesto el sistema de manera que si se quisiera ampliar en el futuro podríamos alojar este servidor back-end en un servidor remoto de igual manera que la base de datos y de este modo no fuera necesario tener que tener una máquina que se encargue de albergar tanto base de datos como de levantar el servidor, y con un par de cambios albergar el sistema en un servidor remoto desde el que acceder por ejemplo con el teléfono móvil.

En grandes rasgos la aplicación tendrá unas 4 pestañas, en cada una de ellas aportando una funcionalidad distinta.

En primer lugar, la pestaña de inicio nos mostrará información básica sobre el sistema, además contaremos con enlaces hacia las demás páginas, una funcionalidad a recalcar de esta página inicio es la de poder ejecutar todo el sistema desde la aplicación web, haciendo clic en un simple botón.

Otra página de la aplicación es la de las tablas, aquí encontramos el acceso a dos tablas. La primera consiste en la tabla de pájaros almacenados en el sistema, que ilustra todas las aves identificadas a lo largo del tiempo, permitiendo además buscar información de cada ave en concreto. La otra tabla muestra la información de todos los análisis del sistema, aportando además la posibilidad de borrar cualquier entrada de esta tabla, haciendo este cambio persistente y borrándolo del sistema por completo.

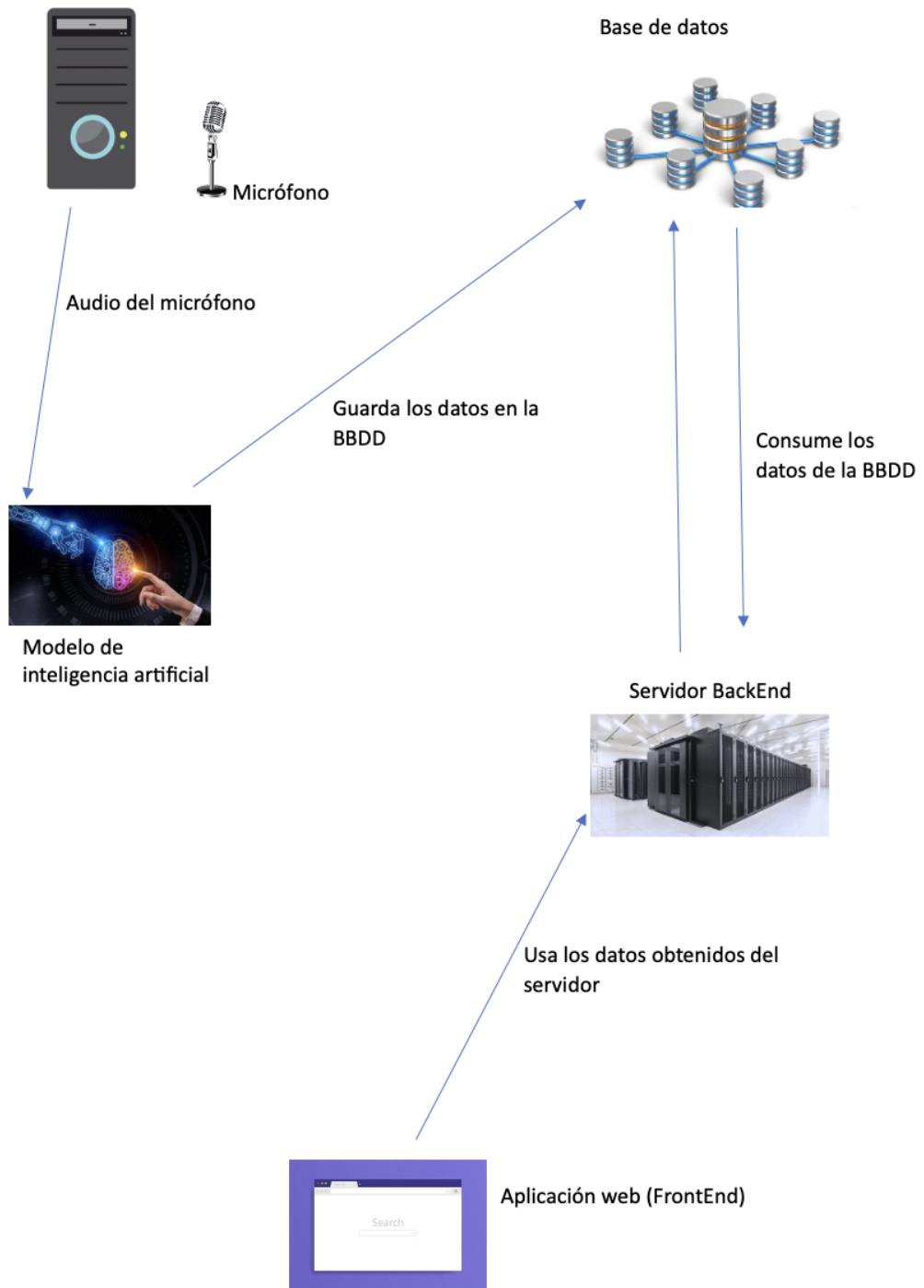


Figura 2.1: Esquema general del diseño del sistema

En la tercera página nos encontraremos con los accesos a varias estadísticas que obtendremos a partir de los datos que nos ofrece el sistema, en primer lugar ofreceremos la estadística de todos los análisis realizados por el sistema en las diferentes fechas, y por otro lado encontraremos la estadística de las aves identificadas en el sistema junto con el número total de aves identificadas.

2.2.2. Base de datos

Para este sistema usaremos una base de datos que nos permita el almacenamiento de la información que recoja el sistema de reconocimiento de aves. En rasgos generales la base de datos albergará los datos previamente tratados que vuelque en ella el sistema de identificación de aves, mientras que la aplicación web consultará u eliminará determinada información en función de las peticiones del usuario.

Requiere haber realizado un estudio previo y adaptación a los requisitos de datos contemplados a la hora de almacenar diferentes datos, la información se debe estructurar de una manera clara, de modo que el acceso a los datos se realice de la manera más sencilla posible.

Se requiere además que se puedan acceder a estos datos desde el back-end de la aplicación de modo que podamos usar los datos para la aplicación web.

La base de datos será una base de datos de tipo relacional, concretamente una base de datos en SQL, para poder visualizar la información de la base de datos usaremos un gestor de base de datos sencillo como es Sqlite3.

Identificamos dos principales requisitos para la base de datos, y esta contará con 2 tablas, una en la que se guardarán los datos correspondientes a los diferentes análisis de sonidos y la predicción de estos, y otra tabla en la que guardaremos un registro de las aves que el sistema ha identificado.

Además se deberá tanto inicializar las tablas de manera autónoma como tratar todas las excepciones, es decir, que si las tablas o la base de datos no está creada, que se cree sola (sin necesidad de que el usuario la cree) tanto la base como las tablas.

2.2.3. Análisis y clasificación del audio

Incorporaremos al sistema el modelo de inteligencia artificial que mencionamos en anteriores apartados, de modo que nos permitirá la identificación de aves a partir de su canto. Se deberá poder acceder al micrófono del ordenador de modo que podamos realizar análisis de los sonidos que esta capturando nuestro micrófono, además los resultados de los análisis deben de ser guardados en la base de datos mencionada en la anterior sección.

Además se debe permitir realizar una ejecución iterativa del sistema, es decir, que al ejecutar el sistema, todo funcione de manera autónoma un numero de iteraciones o un determinado tiempo, por ejemplo, que el sistema funcione durante 60 segundos, y en ese tiempo que realice el siguiente flujo:

- 1. Grabar el sonido del micrófono
- 2. Ejecutar el modelo que detecte que ave se oye en el audio anterior
- 3. Guardar los análisis en base de datos
- 4. Comenzar desde el paso 1.

2.2.4. Ejecución autónoma del sistema

Se desarrollará para el sistema un modo de ejecución autónoma, es decir, que no sea necesario ejecutar cada módulo del sistema de manera individual, sino que por ejemplo, al accionar un botón todo se ejecute de manera secuencial y autónoma. El caso en que el usuario tuviese que accionar manualmente cada proceso del sistema era impensable debido a que tanto el numero de pasos como el flujo que el usuario pudiera realizar los pasos puede afectar gravemente tanto a la usabilidad como a la proyección de la aplicación, de este modo se automatizaron los pasos quedando el proceso de la siguiente manera:

- Ejecución del fichero que captura, procesa y almacena el audio del micrófono.
- Ejecución del fichero que se encarga de la ejecución del sistema de identificación de las aves.
- Ejecución del fichero que se encarga de la lectura de los resultados de los análisis, procesa y guarda los datos en la base de datos.

2.3. Implementación

2.3.1. Introducción

Procedemos a detallar las herramientas y tecnologías seleccionadas para llevar a cabo el proyecto, así como los pasos realizados y los puntos clave en la implementación de cada bloque del sistema completo.

2.3.2. Tecnologías Aplicadas

Editor

Para el desarrollo de todos los módulos de los proyectos hemos usado el editor Visual Studio Code, gratuito y desarrollado por Microsoft.

Bird-NET Analyzer

El modelo de inteligencia artificial que se encarga del análisis y la clasificación del audio lo encontramos ya desarrollado en lenguaje python [2.4](#), el cuál obtenía los au-

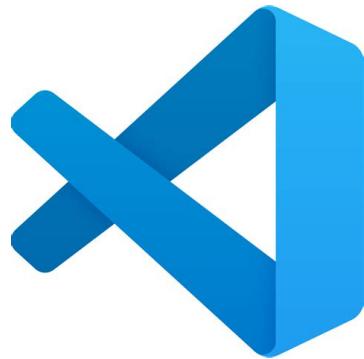


Figura 2.2: Logotipo del editor Visual Studio Code

dios de un fichero .wav y almacena los datos en un fichero .txt, por lo cual tenemos que tener en cuenta estos datos de cara al desarrollo de los demás módulos.

De este modo requerimos tener instalado python en nuestra máquina, además de tensorflow 2.5, en MacOs tensorflow da problema, por lo que tuve que crear un entorno de conda que nos permitiese usar tensorflow para MacOs, de modo que todo aquello que use esta parte, deberemos inicializar el entorno previamente con “conda activate base”.

BirdNET-Analyzer

Automated scientific audio data processing and bird ID.



Figura 2.3: Logotipo del repositorio del modelo de inteligencia artificial.



Figura 2.4: Logotipo del lenguaje de programación python



Figura 2.5: Logotipo de la librería tensorflow para inteligencia artificial.

Captura del audio del micrófono

Para la captura del audio del micrófono conseguimos encontrar una librería python que nos permite la captura del audio del micrófono del ordenador, de este modo en este módulo usaremos como tecnología el lenguaje python.

Desarrollo de la aplicación web

Para el desarrollo de la aplicación web hemos usado varias tecnologías, las cuales las veremos en detalle a continuación.

JavaScript

Para el desarrollo del back-end de la aplicación web decidimos usar el lenguaje JavaScript, dada su amplia documentación en internet, además de haber realizado previamente otros proyectos de desarrollo web, api's y demás con javascript, de este modo consideramos este el mejor lenguaje para el desarrollo del back-end además de ser totalmente compatible con los demás módulos.



Figura 2.6: Logotipo del lenguaje de programación javascript

Base de datos Para el desarrollo de la base de datos optamos por las bases de datos de tipo relacionales, en este caso desarrollaremos una base de datos SQL, como

gestor de base de datos para poder visualizar los datos, las tablas y demás usaremos Sqlite3, desarrollado en lenguaje C, sencillo, rápido y eficiente.



Figura 2.7: Logotipo del gestor de base de datos Sqlite3

Node js

Node js es un entorno en tiempo de ejecución disponible para MacOs, Linux y Windows, principalmente está creado como entorno de ejecución de JavaScript orientado a eventos asíncronos, para la creación de aplicaciones networks eficientes y escalables, además permite realizar múltiples conexiones simultáneamente. En nuestro caso lo usaremos para la creación del servidor del back-end. La versión usada de node es la 19.6.0



Figura 2.8: Logotipo de Node js

Angular

Para el desarrollo del front-end hemos decidido usar el framework Angular, un framework bastante potente, que nos permite modularizar bastante los componentes de la aplicación ordenando bastante cada parte y proporcionando una eficiencia bastante grande, además cuenta con un gran soporte de cara a la escalabilidad y la sostenibilidad del software. La versión de angular usada para este proyecto es la 15.0.5



Figura 2.9: Logotipo del Framework Angular

Angular puede resultar un framework bastante tedioso sobre todo al principio, ya que tiene una estructura de todos los componentes un tanto peculiar, aunque como

todo, una vez uno indaga y entra a fondo en la materia, se ve todo más sencillo, este framework modulariza mucho las partes, y cada pantalla de la aplicación es un “componente”, de este modo tendremos tantos componentes como pantallas tengamos en la aplicación, también encontramos los servicios, estos nos aportan funcionalidades como por ejemplo consumir los datos de una api. A continuación les mostramos un esquema de la estructura de un proyecto angular (figura 2.10), donde encontraremos dos bloques divididos, la parte del back-end, que no forma parte del proyecto angular en si pero usaremos los datos de esta parte, por otro lado tenemos nuestra parte del front-end y todo lo que engloba el proyecto angular, los servicios, los componentes y dentro de estos los ficheros html, js, css y los de configuración necesarios para que el proyecto funcione correctamente.

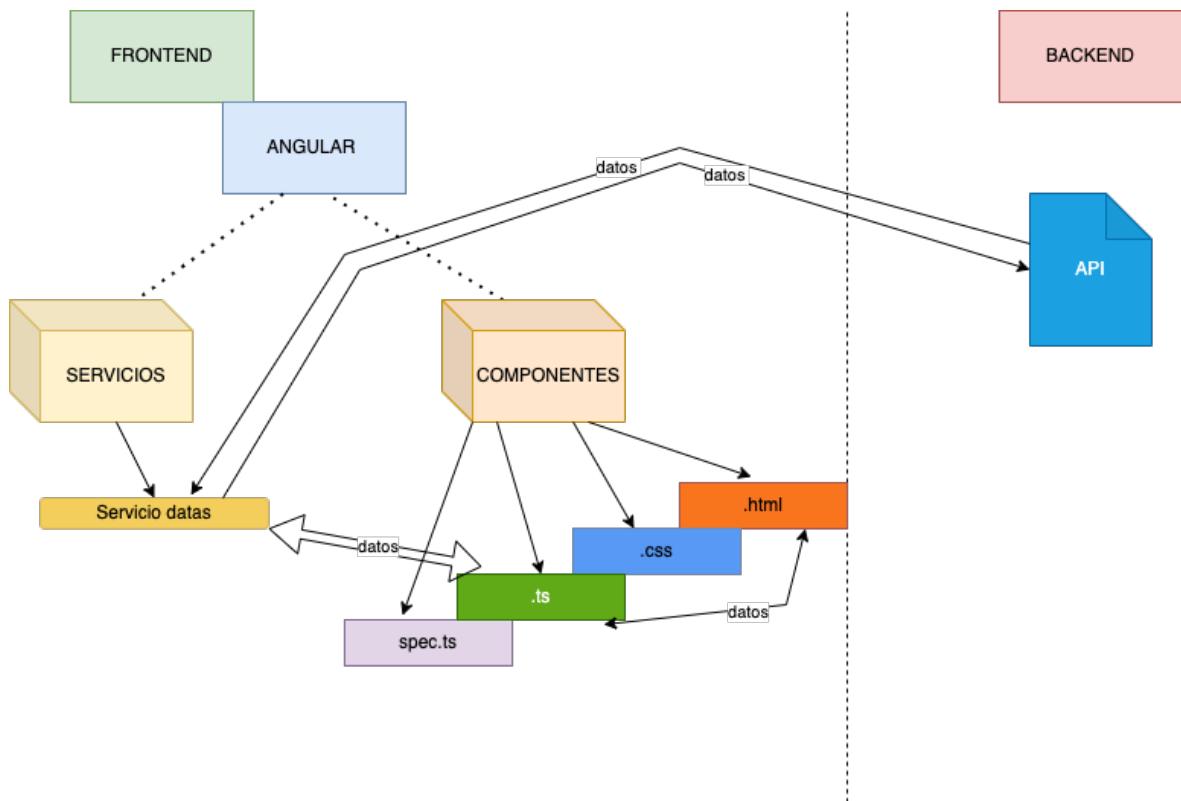


Figura 2.10: Diagrama de la estructura de un proyecto Angular

2.3.3. Desarrollo

A continuación detallaremos los puntos principales del desarrollo del proyecto.

Como anteriormente mencionamos, el proyecto comienza con el estudio del problema, realizando un análisis de los requisitos, la viabilidad de los mismos, tras ellos algunos requisitos necesitaban un estudio previo, como es el caso del modelo de inteligencia artificial, que como bien detallamos en el estado del arte, encontramos tras una exhaustiva búsqueda un modelo que nos proporcionaba la funcionalidad que requeríamos. Hasta saber que dicho modelo nos proporcionaba la funcionalidad que necesitábamos fueron realizadas diversas pruebas, de este modo, el desarrollo como tal comenzó en esta fase.

Estudio y pruebas sobre el modelo de inteligencia artificial

Tras descubrir la existencia de un modelo de inteligencia artificial que podría satisfacer las necesidades de una parte del proyecto antes de incorporar dicho modelo sin verificación de que realmente cumpliese estos requisitos, se realizaron tanto un estudio como unas pruebas.

En primer lugar se procedió al estudio de cómo estaba estructurado dicho modelo. A continuación se muestra la figura 2.11 con la jerarquía de carpetas que se encontraban una vez era importado el proyecto. Cabe recalcar que para la ejecución del mismo se ofrecía una guía por parte de los desarrolladores, en el mismo repositorio del que fue descargado, indicándose las herramientas necesarias para poder usar dicho modelo, además de unos pasos iniciales, los cuales seguí para ver como ponerlo en funcionamiento.

Finalmente, tras un gran estudio sobre cada fichero mostrado en la figura 2.11, un análisis de las tareas que los involucraban y el papel que tomaban dentro del programa como tal, se concluyó cuáles eran los principales para el desarrollo de este proyecto.

Tal como se ilustra en la captura con la jerarquía de los ficheros y carpetas, el principal fichero que encontramos es el fichero “analyze.py”, el cual como ya sabemos está escrito en lenguaje python, y este fichero es el encargado, a grandes rasgos, de analizar los archivos de audio que le indiquemos. Dentro de este fichero encontramos el código correspondiente al algoritmo que se encarga de la predicción de las aves a partir de los sonidos. Se encuentra también algunas funciones de troceo del audio, aunque es el fichero “audio.py” donde se realiza a la lectura del fichero de audio. Además, en el fichero “analyze.py” se encuentran funciones encargadas de manejar y estructurar los parámetros que le pasamos al programa a la hora de ejecutarlo para que se obtengan los datos correctamente y se proceda a una ejecución exitosa. Recalcar que para la ejecución del programa basta con escribir un comando en la terminal de comandos, de modo que este ejecute el fichero analyze.py, indicándole dónde se encuentra el archivo de audio, y dónde queremos que devuelva los resultados de los análisis, todo ello es opcional ya que posee unos parámetros por defecto, un audio de ejemplo y una carpeta (ruta) donde albergar los resultados del análisis en un fichero de texto, esta ruta por defecto se encuentra en la carpeta example, donde tenemos un audio de ejemplo y ahí se guardarán los resultados. Esta ruta por defecto para el guardado no

| | |
|--|---|
|  checkpoints | Model V2.3 update |
|  example | Move to Gradio |
|  extra-hooks | Move to Gradio |
|  gui | added link to contact |
|  labels/V2.3 | Model V2.3 update |
|  .gitignore | accordions |
|  Dockerfile | Update Dockerfile |
|  LICENSE | License update |
|  README.md | Readme update |
|  analyze.py | German Umlaute are now written correctly |
|  audio.py | Write full error to err_log.txt |
|  client.py | Added API server and client |
|  config.py | Model V2.3 update |
|  eBird_taxonomy_codes_2021E.json | Initial commit |
|  embeddings.py | Fix output path handling |
|  gui.py | Replaced old GUI and updated README |
|  model.py | Fixed prediction for protobuf model |
|  segments.py | Merge pull request #77 from AFairbairn/main |
|  server.py | Added utf-8 encoding for label files #49 |
|  species.py | Removed unused default values |
|  translate.py | Adjusted paths requested in #3 |

Figura 2.11: Jerarquía de ficheros del modelo extraído de GitHub

la hemos modificado de modo que cuando realicemos nuestros análisis automáticos con la aplicación se guardarán ahí, únicamente hemos cambiado el nombre del fichero de texto que se escribe. De este modo, teniendo todas las herramientas necesarias indicadas por los desarrolladores en el repositorio instaladas, podemos proceder a la ejecución del programa, en la siguiente captura podemos ver como se ejecuta el comando “python analyze.py” para ejecutar con python el fichero, no le indicamos parámetros como anteriormente mencionamos ya que tiene unos valores establecidos por defecto.

```

○ (base) jesusvenacampos@MacBook-Air-de-Jesus:~/model% 
○ (base) jesusvenacampos@MacBook-Air-de-Jesus:~/model% 
● (base) jesusvenacampos@MacBook-Air-de-Jesus:~/model% python analyze.py
y
Species list contains 3337 species
Analyzing /Users/jesusvenacampos/Universidad/ CUARTO/TFG/Repositorio/TFG/model/example/output.wav
Finished /Users/jesusvenacampos/Universidad/ CUARTO/TFG/Repositorio/TFG/model/example/output.wav in 2.
54 seconds
○ (base) jesusvenacampos@MacBook-Air-de-Jesus:~/model% 

```

Figura 2.12: Resultado de la ejecución del modelo de inteligencia artificial

| Selection | View | Channel | Begin Time (s) | End Time (s) | Low Freq (Hz) | High Freq (Hz) | Species | Code | Common Name | Confidence |
|-----------|-------------|---------|----------------|--------------|---------------|----------------|---------|------|------------------------|------------|
| 1 | Spectrogram | 1 | 0 | 3.0 | 150 | 12000 | bkcchi | | Black-capped Chickadee | 0.6618 |
| 2 | Spectrogram | 1 | 9.0 | 12.0 | 150 | 12000 | houfin | | House Finch | 0.5067 |
| 3 | Spectrogram | 1 | 12.0 | 15.0 | 150 | 12000 | houfin | | House Finch | 0.1234 |
| 4 | Spectrogram | 1 | 12.0 | 15.0 | 150 | 12000 | purfin | | Purple Finch | 0.1152 |
| 5 | Spectrogram | 1 | 15.0 | 18.0 | 150 | 12000 | blujay | | Blue Jay | 0.2294 |
| 6 | Spectrogram | 1 | 18.0 | 21.0 | 150 | 12000 | blujay | | Blue Jay | 0.2049 |
| 7 | Spectrogram | 1 | 21.0 | 24.0 | 150 | 12000 | blujay | | Blue Jay | 0.3831 |
| 8 | Spectrogram | 1 | 21.0 | 24.0 | 150 | 12000 | shshaw | | Sharp-shinned Hawk | 0.1194 |
| 9 | Spectrogram | 1 | 21.0 | 24.0 | 150 | 12000 | baleag | | Bald Eagle | 0.1003 |
| 10 | Spectrogram | 1 | 27.0 | 30.0 | 150 | 12000 | daejun | | Dark-eyed Junco | 0.1200 |
| 11 | Spectrogram | 1 | 33.0 | 36.0 | 150 | 12000 | daejun | | Dark-eyed Junco | 0.3555 |
| 12 | Spectrogram | 1 | 33.0 | 36.0 | 150 | 12000 | amtspa | | American Tree Sparrow | 0.1675 |
| 13 | Spectrogram | 1 | 36.0 | 39.0 | 150 | 12000 | merlin | | Merlin | 0.1338 |
| 14 | Spectrogram | 1 | 36.0 | 39.0 | 150 | 12000 | daejun | | Dark-eyed Junco | 0.1159 |
| 15 | Spectrogram | 1 | 42.0 | 45.0 | 150 | 12000 | daejun | | Dark-eyed Junco | 0.7101 |
| 16 | Spectrogram | 1 | 42.0 | 45.0 | 150 | 12000 | merlin | | Merlin | 0.3600 |
| 17 | Spectrogram | 1 | 48.0 | 51.0 | 150 | 12000 | merlin | | Merlin | 0.1608 |

Figura 2.13: Resultado del fichero tras la ejecución

Como podemos ver en la figura 2.12 nos muestra un pequeño texto indicando el número de especies con las que cuenta para identificar, luego la ruta del fichero que va a analizar, tras esto nos indica la ruta del fichero en el que se guardaran los resultados pertinentes del análisis y por último nos indicará el tiempo en el que se ha realizado el análisis.

En la figura 2.13 podemos ver a la izquierda al igual que en la figura donde mostrábamos la jerarquía desde el repositorio, la carpeta “example”, donde se guardan por defecto tantos los archivos de audio para analizar como los ficheros de texto donde se escriben los resultados. En la parte central de la figura podemos apreciar un fichero de resultados tras la ejecución del sistema, donde nos indican los siguientes datos: un identificador del numero de fila, la vista del archivo de audio de donde el fragmento de audio que identifica el sonido con el ave ha sido obtenido, el canal de audio (por defecto a 1), el tiempo de inicio de análisis, el tiempo de fin del análisis, la frecuencia baja del audio, la frecuencia alta del audio, el código de la especie, el nombre común y el nivel de confianza (en tanto por 1) que estima el modelo sobre la identificación del ave indicada.

Tras probar el programa y ver los resultados obtenidos, le sigue plantear si realmente nos es de utilidad para el desarrollo de nuestro proyecto y lo podemos incorporar. En cuanto a precio y copyright no es problema ya que este es totalmente gratuito y open source, por la parte más engorrosa que es acoplarlo con la nueva funcionalidad que deseamos, no parece ser que vaya a haber conflictos ya que únicamente necesitamos poder ejecutarlo de forma autónoma desde un script, poder enviarle los archivos de audio que capturemos desde el micrófono en tiempo real, y leer el fichero de texto con los resultados que nos devuelva tras un análisis, como estas tareas no deben de entrar en conflicto con el modelo, se decidió utilizar este sistema ya entrenado para la detección de aves a partir del sonido, de este modo concluimos con esta primera parte del desarrollo.

A continuación, el desarrollo prosiguió con las tareas pertinentes a la captura del audio que el micrófono de nuestra máquina puede percibir.

Captura del audio de un micrófono

El objetivo a alcanzar era poder obtener el audio en tiempo real del micrófono de la máquina en la que estuviéramos ejecutando el programa, para posteriormente poder usar ese audio para realizar los análisis con el modelo de inteligencia artificial.

El desarrollo de este objetivo comenzó con pocas ideas sobre cómo abordar el problema debido a que nunca antes había afrontado un objetivo como éste, y en primera instancia resultó algo novedoso.

Tras una exhaustiva búsqueda de cómo podía ser resuelto dicho problema, resultó haber varias librerías de python que nos permitía realizar dicha tarea. Tras ver varias librerías, me decanté por pyaudio, una librería que cuenta con un gran número de funciones para grabación y tratamiento del audio. Podemos decidir el tiempo que queremos estar capturando el audio, para este proyecto hemos decidido que lo más adecuado es grabar audio de 5 segundos ya que la posterior ejecución iterativa nos permitirá realizar varias grabaciones. Una vez se ha grabado el audio, procedemos al guardado de este en la ruta que será leída por el programa de reconocimiento de aves

para la identificación de las aves en el audio.

Guardado en base de datos

Una vez tenemos el modelo de inteligencia artificial que se encarga de los análisis, y la funcionalidad de poder grabar el audio del micrófono y que el modelo pueda usar este audio para los análisis, procederemos a comenzar con el desarrollo de todo lo relacionado con la base de datos.

En primer lugar recordamos que la base de datos es una base de datos relacional Sql, estará formada por dos tablas, una en la que se almacenará toda la información de los análisis y en la otra un registro de todas las aves que vamos a identificar a lo largo de la vida del sistema, sin incluir valores duplicados.

Otro requisito era que la base de datos debe de poder crearse desde cero de forma autónoma, en caso de que no esté creada que se cree automáticamente y en caso de que no existan las tablas que se creen de igual manera, además que los datos pertinentes de los análisis se añadan de igual manera automáticamente.

De esto modo la solución que se ha pensado para este objetivo es realizar un script donde podamos crear una conexión con la base de datos, comprobemos si esta creada o no, y crearla en caso de que no, de igual manera con las tablas, y luego añadir los datos de los análisis.

Comenzamos obteniendo información sobre cómo podemos realizar una conexión con una base de datos con python, encontramos para esta funcionalidad una librería de python que nos permite realizar conexiones con sqlite3, de este modo nos permitiría tanto la creación como la modificación de la base de datos.

Esta librería es “sqlite3”, nuestro script que se encargará tanto de la creación de la base de datos y las tablas de esta como de la inserción de nuevos datos en la tabla se estructurará en tres funciones, una función principal, la función main que se ejecuta al lanzar el script, esta función únicamente contendrá la llamada a otras dos funciones, una llamada a la función lectura de datos y almacenará el resultado de esta en dos variables que usará en la siguiente llamada, y otra llamada a la función database pasándole como parámetro las variables con los datos pertinentes que devolvió la llamada de la función anterior. A continuación pasaremos a explicar las dos funciones en detalle.

En primer lugar la primera función que se ejecuta desde la función principal como ya mencionamos anteriormente es la función “lecturaDatos”. Esta función no recibe ningún parámetro de entrada, el cometido de esta función es la lectura del fichero en el que tenemos los resultados obtenidos de los análisis de los ficheros de audio, tras ello se realizará un pequeño tratamiento de estos datos, eliminando aquellos que no vayan a ser de utilidad, de este modo una vez finalice la ejecución de esta función, se devolverán dos listas, con todos los datos necesarios y requeridos para insertarlos en la tabla pertinente en la base de datos.

Llegados a este punto había que decidir qué datos íbamos a insertar en las tablas de la base de datos, para quedarnos únicamente con estos, finalmente los datos que se guardaron de cara a ser insertados son los siguientes.

En la tabla de los análisis se guardarán los siguientes datos:

- Identificador único por cada registro tras el análisis (el número de fila de la tabla)
- El tiempo de inicio del análisis
- El tiempo de fin del análisis
- El código de especie del ave
- El nombre común del ave.
- La confianza de que el sonido del ave sea o no sea el indicado.
- El día y la fecha del análisis.

De este modo y con los datos que estipulamos que debían ser insertados en las tablas, comenzamos con el desarrollo del código de la función. Lo principal era realizar la lectura del fichero de texto donde se encontraban los resultados y tras ello la implementación de un bucle que recorriera todas las líneas del fichero, y ya por cada línea realizábamos el tratamiento de los datos pertinentes para eliminar aquellos que no eran necesarios, eliminar algunos caracteres que daban conflictos y añadir la fecha y hora ya que esta no se encontraba entre los datos que nos devolvía el modelo.

Se obtuvo la fecha y hora del sistema con la ayuda de la librería “time”, gracias a esta librería obtuvimos a la perfección la fecha y hora del sistema y se añadió a cada fila de los datos. Queda recalcar que obtenemos la fecha y hora actual del sistema ya que el propósito es que se realice un proceso automático que capture el audio, se lo pase al modelo y lance la ejecución, obtenga los datos resultantes, y los guarde en base de datos, todo ello de manera secuencial y autónoma.

En el caso de la tabla de las aves identificadas por el sistema se guardará los siguientes datos:

- Identificador único por cada registro.
- Nombre del ave
- Código de especie del ave

Estos datos igualmente los obtendremos del mismo fichero resultante de los análisis, efectivamente podríamos tener una única tabla con todos los datos y extraer esto de la tabla anterior, pero debido a las funcionalidades futuras a implementar como por ejemplo habilitar la posibilidad de borrar filas de la tabla de análisis. Decidimos de esta manera crear una nueva tabla donde albergásemos la información de las aves que hemos ido identificando con el sistema, sin que éstas fueran borradas.

La restricción de que estos datos no sean duplicados se tratará en la función de la base de datos, de modo que a continuación cuando detallaremos el funcionamiento y el código de la función se indicará donde se cumple dicha restricción.

Finalmente se almacenaban los datos en una lista de listas, donde cada sublista de la lista principal es una línea del fichero, es decir, una entrada a la tabla de la base de datos, y se devolverán dos variables, una por cada lista en la que añadiremos datos.

Este módulo realiza las siguientes acciones:

Se abre el fichero donde se devuelven los datos tras los análisis, se realiza el bucle mencionado anteriormente para recorrer cada fila del archivo y se eliminan los caracteres de salto de linea, algunos datos que no nos sirven y se añade la fecha con la llamada a la función strftime de la librería time. En el mismo bucle se añaden a las variables data y birds los correspondientes datos, y luego fuera del bucle, en el caso de la lista de birds que es una lista de tuplas, creamos una lista de listas donde cada sublistas de la lista principal estará formado por los elementos de la tupla que existía anteriormente, tras ello se devuelven las dos listas de listas.

En este momento se procede a detallar la función encargada de la creación/modificación de la base de datos. Esta función recibe dos listas de listas como parámetros, donde se almacenan los datos que debemos añadir a las tablas correspondientes. En primer lugar creamos una conexión con la base de datos, esta conexión nos aporta la funcionalidad de que en caso que no exista la base de datos, la crea, tras ello comenzamos a la creación de las tablas de la base de datos. En primer lugar se crea la tabla de aves, con las columnas que indicamos en la anterior sección al indicar los datos que guardamos en la función de lectura del fichero de resultados, en caso de que la tabla ya esté creada debemos controlarlo y que no nos devuelva ningún error. Tras esto creamos de igual manera la tabla con los resultados de los análisis.

Una vez estamos seguros que las tablas han sido creadas, deberemos añadir los datos en éstas, de este modo comenzamos con la inserción de los datos en la tabla de aves, aquí controlaremos la restricción comentada anteriormente, comprobando si ese ave ya se ha añadido, en el caso que ya se haya añadido no se añade, y en caso contrario lo añadiremos. Posteriormente pasamos a la inserción de los datos en la tabla de registros de los resultados, donde no es necesario controlar ninguna restricción. Finalmente guardamos los cambios en la base de datos, cerramos la conexión con la base de datos, y muy importante, borramos los datos del fichero de resultados, de este modo evitamos que en posteriores lecturas y guardados de estos datos se produzcan duplicaciones.

Tras la creación de la base de datos se crea un fichero donde se encuentran los metadatos de la base de datos, abriendo este con el visor podemos ver de una forma visual la base de datos.

A continuación vamos a mostrar cómo se verían la base de datos tras la creación de esta y la inserción de los datos en las tablas, usando un visor para la base de datos llamado “DB BROWSER FOR SQLITE”:

| Nombre | Tipo | Esquema |
|------------------|-------------|---|
| Tablas (3) | | |
| birds | | CREATE TABLE birds(id_bird integer PRIMARY KEY, "name" TEXT UNIQUE, speciesCode VARCHAR(30)) |
| id_bird | integer | "id_bird" Integer |
| name | TEXT | "name" TEXT UNIQUE |
| speciesCode | VARCHAR(30) | "speciesCode" VARCHAR(30) |
| datas | | CREATE TABLE datas(id_data integer PRIMARY KEY, beginTime VARCHAR(30), endTime VARCHAR(30), speciesCode VARCHAR(30), commonName VARCHAR(30), confidence VARCHAR(30), datetime DATETIME) |
| id_data | integer | "id_data" Integer |
| beginTime | VARCHAR(30) | "beginTime" VARCHAR(30) |
| endTime | VARCHAR(30) | "endTime" VARCHAR(30) |
| speciesCode | VARCHAR(30) | "speciesCode" VARCHAR(30) |
| commonName | VARCHAR(30) | "commonName" VARCHAR(30) |
| confidence | VARCHAR(30) | "confidence" VARCHAR(30) |
| datetime | DATETIME | "datetime" DATETIME |
| sqlite_sequence | | CREATE TABLE sqlite_sequence(name, last_value) |
| Índices (0) | | |
| Vistas (0) | | |
| Disparadores (0) | | |

Figura 2.14: Base de datos desde el visor de base de datos

En la siguiente figura podemos ver la tabla correspondiente a los resultados, en la sección hoja de datos del visor.

| | Id_data | beginTime | endTime | speciesCode | commonName | confidence | datetime |
|---|---------|-----------|---------|-------------|--------------------|------------|----------------------|
| | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro |
| 1 | 3 | 3.0 | 6.0 | humvoc | Human vocal | 0.1048 | 03/09/2023, 15:53:52 |
| 2 | 4 | 0 | 3.0 | picpig2 | Picazuro Pigeon | 0.1691 | 03/09/2023, 15:54:05 |
| 3 | 5 | 0 | 3.0 | rocpig | Rock Pigeon | 0.1140 | 03/09/2023, 15:54:05 |
| 4 | 6 | 3.0 | 6.0 | cowplig1 | Common Wood-Pigeon | 0.1771 | 03/09/2023, 15:54:05 |
| 5 | 7 | 3.0 | 6.0 | wepdov1 | West Peruvian Dove | 0.1619 | 03/09/2023, 15:54:05 |
| 6 | 8 | 3.0 | 6.0 | whtdov | White-tipped Dove | 0.1039 | 03/09/2023, 15:54:05 |

Figura 2.15: Tabla de registros de los resultados

LLegados a este punto contamos con lo siguiente:

- Modelo de inteligencia artificial.
- Script para la captura del audio del micrófono.
- Script para la creación de la base de datos y la inserción de estos.

Hasta este momento contamos con esas funcionalidades para el sistema, de modo que para ver esto en funcionamiento deberíamos de seguir el siguiente flujo:

- Ejecutar el script que captura el audio del micrófono.(Esperar que termine)
- Ejecutar el modelo de inteligencia artificial que identificará las aves.(Esperar que termine)
- Ejecutar el script de base de datos para guardar los resultados de los análisis.

De este modo el usuario que ejecutase el sistema debería de ejecutar ficheros y esperar hasta que estos terminen, de este modo podrían inducirse problemas que no deberían aparecer, por ejemplo, en el caso que el usuario intente lanzar el modelo que analice el audio antes de que este audio haya sido grabado puede darnos problemas tanto de inconsistencia de datos como excepciones en la ejecución, de este modo se incorpora al sistema la funcionalidad de que todo esto se ejecute automáticamente de forma secuencial y en segundo plano, es decir, el usuario final únicamente ejecutará un fichero, el cual se encargará de realizar el flujo que mencionamos anteriormente en la última tabla, ahorrando así al usuario final pasos innecesarios, simplificando el sistema de cara al usuario y ahorrándonos errores innecesarios. Además de la razón de peso anterior para automatizar el sistema es que dejar el sistema sin automatizar puede restar valor al sistema ya que limitaríamos en gran medida el sistema de reconocimiento de aves ya que para que este funcione tendría que estar el usuario en todo momento pendiente del sistema y de este modo automatizándolo nos aporta el valor de recoger una mayor cantidad de datos de manera más cómoda de cara a las estadísticas y el histórico de análisis.

Ejecución del sistema automática y en segundo plano

A continuación se detallaran los pasos y la lógica seguida durante el desarrollo de fichero que nos permite añadir esta funcionalidad al sistema.

En primer lugar se procedió a realizar un estudio sobre cómo se podían lanzar ejecuciones de ficheros python en segundo plano desde un fichero python. Tras una búsqueda en profundidad de las herramientas que permitían realizar estas tareas, encontramos en primer lugar la librería threading que nos permitía la ejecución de funciones en un hilo secundario. La librería threading nos permite activar o desactivar hilos de ejecución dentro de la máquina. Estos hilos son los que nos permitirán ejecutar tareas mientras en otros hilos se ejecutan otras tareas distintas. Tras ello deberemos ejecutar la función deseada en el hilo, una vez la ejecución haya comenzado pondremos un temporizador, para que una vez haya pasado el tiempo deseado, la ejecución pare.

Finalmente una vez haya acabado el tiempo de ejecución, para parar las tareas debemos de cerrar el hilo y de este modo habría acabado la ejecución de nuestro pro-

grama. En este caso, en el punto en el que indicamos la función a ejecutar dentro del hilo, ahí en esa función es donde va el código correspondiente a ejecutar los diferentes módulos del sistema. Ahí es donde encontramos la otra herramienta que necesitaríamos para completar esta funcionalidad, y es la librería subprocess, la cual nos permite lanzar la ejecución de scripts desde un script python. Esta función la ejecutaremos mientras el contador/temporizador que mencionamos anteriormente no este a cero, entonces, ahí ejecutaremos de manera secuencial el fichero de captura del audio, el del análisis de los audios y el de el guardado en base de datos, tras ello se volverá a comprobar si el temporizador sigue activo, en ese caso volvemos a realizar el mismo proceso, en caso contrario, finalizamos la ejecución.

A continuación mostramos un diagrama que nos muestra el flujo que sigue el código que nos permite incorporar esta funcionalidad en el sistema.

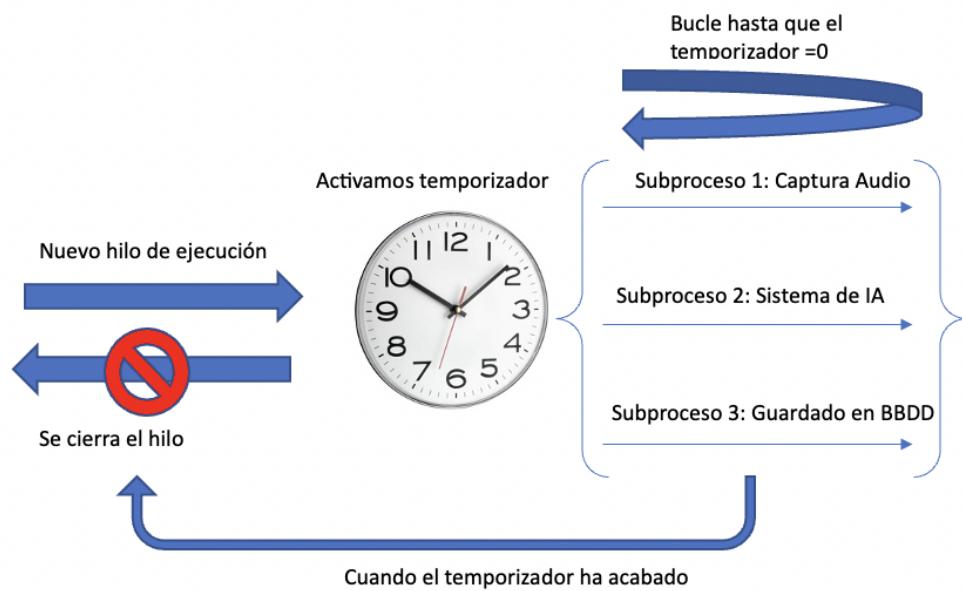


Figura 2.16: Diagrama del código que permite la ejecución iterativa del sistema

Una vez hemos completado esta funcionalidad ya tenemos todo lo necesario para que el sistema funcione, aportando la funcionalidad requerida, como podemos apreciar tras todo lo explicado anteriormente, hemos llegado a un punto en el que tenemos un flujo de datos completamente autónomo, sin necesidad que el usuario mueva ficheros ni los ejecute continuamente, a continuación vamos a revisar el flujo del sistema una vez hemos realizado todos los módulos necesarios:

- El usuario ejecuta el sistema, dandole al botón de ejecutar por ejemplo del Vs-Code.
- Se lanza el fichero que ejecuta de forma iterativa todos los módulos secuencialmente.
- Se lanza el primer módulo indicado en el fichero anterior, en este caso, se procede a la captura del audio del micrófono.
- Se procede al análisis del audio que acabamos de grabar.
- Se procede al guardado de los resultados del anterior análisis.
- Se comprueban parámetros y se sigue o se termina la ejecución del sistema.

A continuación se mostrará un diagrama en el que podemos apreciar con claridad el flujo que acabamos de listar, así como los módulos existentes:

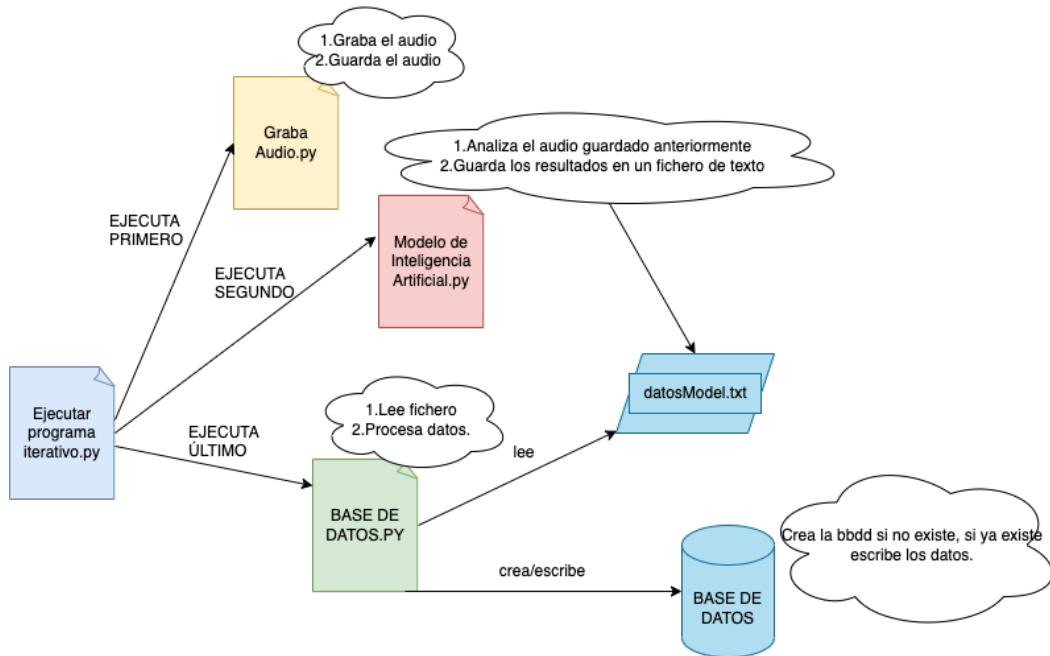
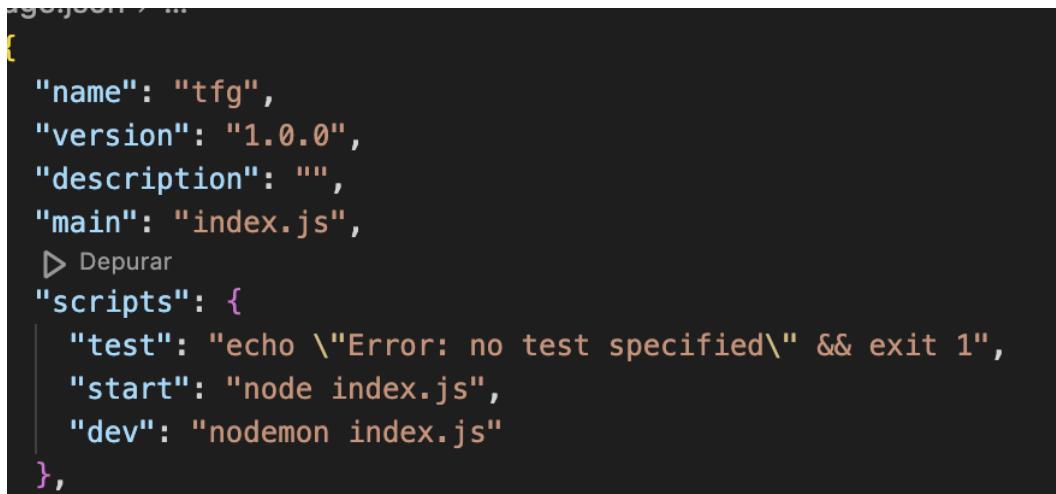


Figura 2.17: Diagrama de los módulos y del flujo del sistema

Llegados a este momento pasamos a un bloque fundamental del sistema, el bloque de la aplicación web, esta es imprescindible en el sistema. Es cierto que un usuario experimentado y con amplios conocimientos informáticos podría desempeñarse con la terminal y accediendo al gestor de base de datos para visualizar estos, pero un usuario menos experimentado podría afrontar dificultades innecesarias y el uso del sistema podría llegar a ser algo bastante tedioso. De este modo, tanto para usuarios experimentados como para los que no, desarrollamos una aplicación web, que para ambos permite una interacción con el sistema bastante más intuitiva además de mostrar los datos de una manera mas clara y sencilla. De igual manera la aplicación web también fomenta el uso del sistema pensando en todos los tipos de usuarios.

Aplicación web El desarrollo de la aplicación web la vamos a dividir en dos grandes bloques, por una parte detallaremos el desarrollo del back-end, donde encontraremos la api del sistema que nos permite el acceso a la base de datos y el uso de estos dato posteriormente por el otro bloque, y por otra parte encontramos el bloque del front-end, donde se encuentra todo lo relacionado con la parte visual de la aplicación, las pantallas de estas, los datos que se muestran, etc.

Back-end El desarrollo del back-end comienza con la instalación de las herramientas necesarias, como editor usaremos igual que hasta ahora el Visual Studio Code, y necesitaremos instalar nodejs el cual instalamos desde la página oficial de node y dejando los pasos de la instalación por defecto. En mi caso no instale node al ya tenerlo en mi máquina. El primer paso es crearnos el directorio, la carpeta donde albergaremos el proyecto, una vez estemos dentro de la carpeta, ejecutamos el comando “npm init” para inicializar el proyecto, de este modo nos creará un archivo llamado “package.json” donde se encuentra toda la configuración del proyecto. Este fichero lo modificamos levemente, mostramos en la siguiente figura los cambios:



```
package.json
{
  "name": "tfg",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Depurar
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1",
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
}
```

Figura 2.18: Modificaciones del package.json

En este caso modificamos los puntos start y dev de la parte de scripts, en el caso del start, indicamos que cuando hagamos “npm start”, se ejecute el archivo index.js, y en el caso de cuando ejecutemos “npm run dev”, ejecute nodemon index.js, nodemon es una herramienta de node js que detecta los cambios que se realizan mientras la aplicación se esta ejecutando y reinicia este automáticamente, ahorrando así mucho tiempo al desarrollador. Tras esta iniciación en nodejs, continuaremos creando-

nos un fichero index.js en la carpeta donde hemos inicializado el proyecto node, en este fichero se encontrará el código que se encarga de levantar nuestro servidor del backend. Instalaremos los paquetes necesarios con el comando “npm i nombre del paquete”, en mi caso por ejemplo tuve que instalar el paquete express, con el comando “npm i express”, express nos proporciona una infraestructura web rápida y sencilla para nodejs. En el fichero index.js nos encargaremos de requerir las dependencias necesarias para el servidor backend, donde recalcamos la dependencia de un fichero llamado db, este se encargará de realizar la conexión con la base de datos sqlite3 de modo que cuando desde el frontend se requieran datos de la base de datos podamos hacer peticiones desde aquí a la base de datos.

Otra dependencia importante que tenemos es el archivo indexApi contiene todos los métodos necesarios para consumir los datos de la base de datos, en este caso esta api no cuenta con métodos PUT ni POST, debido a que no consideramos que fuera necesario para la funcionalidad esperada de la aplicación, de este modo encontraremos únicamente métodos GET y DELETE.

Se han desarrollado varios métodos get, uno para el acceso a la tabla de resultados de análisis y otro método para el acceso a la tabla de registros de aves. Encontraremos una función y otra petición dentro del fichero, pero estas las explicaremos más adelante.

Finalmente encontraremos otro método más en la api, este método no es una petición cualquiera, en primer lugar encontramos el problema de que si queríamos mostrar al usuario información específica sobre un ave, deberíamos de guardar miles de datos de aves y su información, cosa que podría resultar un proceso bastante tedioso, de este modo se me ocurrió la idea de integrar la api del actualmente famoso Open-AI, exacto, la api de Chat-GPT-4 sería el encargado de buscar los datos correspondientes al ave que le indiquemos, de esta forma aprendemos a cómo integrar esa api con la nuestra, y nos ahorraremos el proceso de recopilar tales cantidades de información, aportando así al usuario una nueva funcionalidad en la aplicación, queda recalcar que la api es de pago, pero tras el estudio del precio de esta api es bastante reducido de modo que para los meses de duración del trabajo, el presupuesto presente cubrirá los gastos sin problema.

Una vez hemos desarrollado todo este código mencionado, comenzaremos con el desarrollo de la parte del front-end.

Front-end El front-end es todo aquello que el usuario vera o interactuará de alguna manera, para el desarrollo del front-end y toda la lógica que a este engloba se usará el framework Angular, un framework potente, con una amplia documentación, y muy escalable por si el día de mañana se decide ampliar la aplicación. Lo primero que debemos realizar es la instalación de angular en nuestra máquina y las configuraciones previas, en este caso yo seguí la guía que se encuentra en la página oficial de angular. Una vez que disponemos de todo lo necesario para realizar el proyecto, comenzamos inicializando este. Para inicializar un proyecto angular basta con escribir el comando “ng new nombre del proyecto” ya se creará nuestro proyecto, en el caso de angular en primer momento si nunca se ha usado puede resultar algo lioso, ya que cuenta con aspectos diferentes a lo que puede ser un proyecto base de desarrollo web con simple código hmtl, css y js. De este modo, detallaremos que angular modulariza mucho las partes de la aplicación, donde tendremos dos elementos principales, los componentes y los servicios, en resumidas cuentas y a muy grandes rasgos, los componentes podemos decir que son las pantallas de la aplicación, cada pantalla, tiene su propio componente, y cada componente tiene una carpeta en la que tiene un archivo .html donde obviamente escribimos el código html, un archivo .css donde se detallará el código de estilos css, un archivo .ts donde podremos crear funciones que use el front-end, detallar en este punto que Angular no usa javascript como tal, sino que usa un lenguaje llamado TypeScript, bastante similar a js, retomando los ficheros que se encuentran en la carpeta de componentes tenemos uno mas, el .spec.ts, el cual no he tocado en ningún momento a lo largo del proyecto, estos son pruebas unitarias, las cuales suelen venir definidas por defecto, y como en mi caso no hemos usado este tipo de pruebas, no lo hemos modificado. En el caso de los servicios aportan funcionalidades a la aplicación, funcionalidades bien definidas, por ejemplo un servicio puede ser el de obtener los datos de una base de datos.

Tras estas leves pinceladas sobre angular, mostramos a continuación como sería la estructura base cuando creamos un nuevo proyecto:

En cuando a carpetas señaladas tenemos nodemodules, donde se irán añadiendo los módulos que instalemos con node necesarios para el proyecto, en la carpeta src tendremos todo lo relacionado con el código en sí del proyecto, ahí irán los componentes y servicios que mencionamos anteriormente, la carpeta assets en la que añadiremos algunas imágenes, en cuanto a archivos a recalcar tenemos el index.html donde detallaremos las importaciones necesarias para el html,css o js, luego recalcamos el package-lock.json donde se meten las dependencias del proyecto. Los ficheros que comienzan por app son los compartidos por todos los demás de la aplicación, es decir, en el fichero app.component.html incluiremos el código correspondiente a la barra de navegación y esta automáticamente se mostrará en todas las demás pantallas.

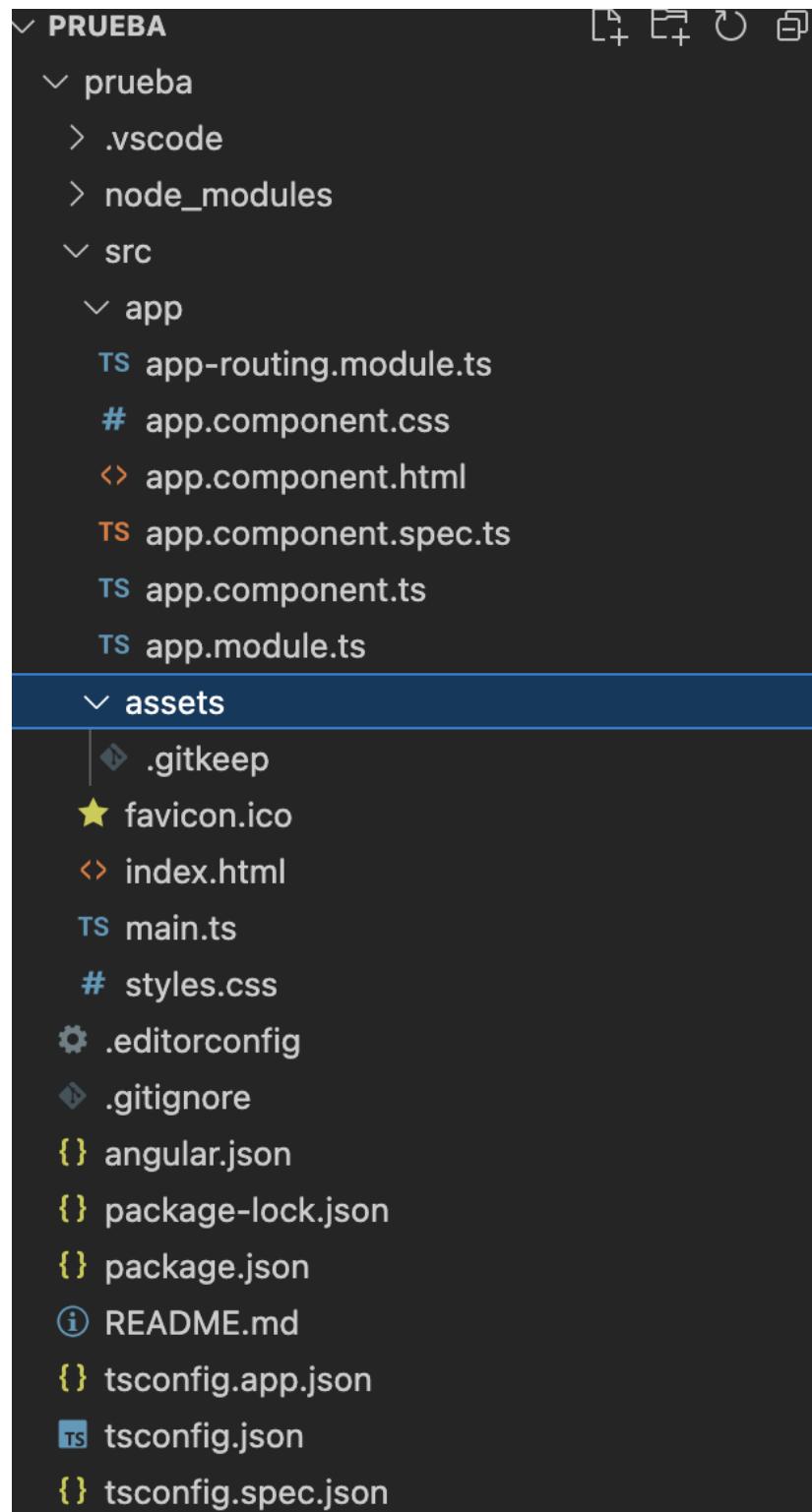


Figura 2.19: Jerarquía inicial de carpetas de un proyecto angular

Una vez está clara la organización de los elementos dentro de un proyecto angular, el desarrollo comenzó creando los componentes y los servicios necesarios, con los comandos “ng g c nombre del componente z “ng g s nombre del servicio respectivamente, finalmente contamos con nueve componentes, uno por cada pantalla de la aplicación, y tres servicios, uno para consumir la api, otro para la renderización de un spinner de carga, y otro un interceptor para habilitar el uso de peticiones http y https.

Mostramos a continuación las pantallas de la aplicación y las funcionalidades que hay en cada pantalla:



Figura 2.20: Pantalla principal de la aplicación

Podemos apreciar en la parte superior de la pantalla la barra de navegación, esta la encontraremos en todas las pantallas de la aplicación, con acceso a las páginas de acceso a las tablas, acceso a las gráficas y a la de documentos. En primer lugar encontraremos la información principal sobre el proyecto y las bases de este y detallaremos algunas de las tecnologías usadas. Tras esto nos encontramos un botón en el que pone “Ejecutar la aplicación”, este botón nos permite ejecutar todo el sistema interno que explicamos anteriormente desde la interfaz gráfica, de este modo evitamos que el usuario final tenga que usar terminales de comandos ni editores de código desde el que lanzar el fichero que ejecuta todo el sistema, este botón aporta un gran valor al sistema ya que con él podemos encapsular todo lo necesario para el usuario únicamente en la interfaz, facilitándole a este el uso del sistema y nos evitamos que problemas como por ejemplo al tener que ejecutar desde la terminal, que elimine cualquier archivo sin querer o multitudes de posibles problemas. A continuación mostramos la pantalla en vista móvil, esta vista es importante considerarla por si algún día se habilita la aplicación para móviles, además de comprobar que es responsive la aplicación a modo de calidad y buenas prácticas.



Tablas de los registros de la aplicación



A continuación veremos una tabla con los registros de todas las aves identificadas por la aplicación, además de información sobre estas mediante el acceso a la Api de OpenAI

[Tabla de registros de aves](#)



Procederemos a ver en forma de tabla todos los registros de los análisis realizados por la aplicación, con la posibilidad de eliminar alguno si se desea.

[Tabla de registros de la aplicación](#)

Figura 2.21: Vista móvil de una pantalla de la aplicación

A continuación mostraremos la pantalla de acceso a las tablas, podemos apreciar en ella a la izquierda una caja con un texto de información y un botón que nos llevará a la tabla de registros de aves, de manera análoga, a la derecha, accederíamos a la tabla de registros de la aplicación.

Como podemos apreciar en la vista de ordenador encontramos al igual que en la pantalla de inicio la barra de navegación, y en la vista móvil se encuentra colapsada hasta que le demos al botón de desplegarla, y encontramos las dos cajas en las que tenemos dos botones a modo de enlace, a continuación se mostrará en la siguiente figura la página donde se encuentra la tabla de aves, que se accede clicando desde el enlace situado más a la izquierda en la vista de ordenador y el primer enlace que encontraríamos si estuviéramos en la vista de móvil.

Tablas de los registros de la aplicación



A continuación veremos una tabla con los registros de todas las aves identificadas por la aplicación, además de información sobre estas mediante el acceso a la API de OpenAI

[Tabla de registros de aves](#)



Procederemos a ver en forma de tabla todos los registros de los análisis realizados por la aplicación, con la posibilidad de eliminar alguno si se desea.

[Tabla de registros de la aplicación](#)

[Volver](#)

Figura 2.22: Pantalla correspondiente al acceso a las tablas de datos.

Tabla de registros de aves

| Entrada | Nombre Común Ave | Código de especie | Más información |
|---------|--------------------|-------------------|---------------------------------|
| 1 | Common Name | Species Code | Más información |
| 2 | Power tools | powtoo | Más información |
| 3 | Engine | engine | Más información |
| 4 | Human vocal | humvoc | Más información |
| 5 | Common Wood-Pigeon | cowpig1 | Más información |
| 6 | White-tipped Dove | whtdov | Más información |
| 7 | West Peruvian Dove | wepdov1 | Más información |
| 8 | Rock Pigeon | rocpig | Más información |
| 9 | Picazuro Pigeon | picpig2 | Más información |
| 10 | Human non-vocal | humnov | Más información |

« Anterior 1 2 [Siguiente »](#)[Volver](#)

Figura 2.23: Pantalla correspondiente a la tabla de aves del sistema.

En la figura 2.23 nos encontramos con una tabla donde se muestran los registros de aves del sistema, donde se indica el número de entrada, el nombre común del ave, el código de especie, y se muestra un botón en el que podemos acceder a más información del ave. Podemos apreciar en la parte inferior izquierda de la pantalla la paginación de los datos de la tabla, limitando la muestra de estos a diez filas máximo, pudiendo movernos entre las diferentes pestañas de la tabla con los controles de la paginación, también encontramos un botón de volver con el que volveríamos a la página anterior, también podríamos volver a la página de inicio clicando sobre “TFG.” en la barra de navegación.

La funcionalidad del botón de “Más información.” es un tanto curiosa, como queríamos mostrar la información de un ave en concreto, comenzó el estudio de cómo podríamos

implementar esto, en principio la idea más clara era realizar una búsqueda de miles de datos de aves y almacenarlos en una base de datos a la que pudiéramos hacerle peticiones en el momento, bueno, pues evidentemente como vemos esto es un proceso muy tedioso a la par que bastante ineficiente debido a la carga de horas que debían ser dedicadas al proyecto, de este modo, como estaba tan de moda Chat-GPT, y acaban de sacar la api al mercado de forma usable y estable, me decidí a probarla, hasta que finalmente tras unas pruebas caí en que podía ser incorporada al proyecto de modo que viéramos una funcionalidad bastante atractiva y aporte además valor a lo que es la solución, aportar como dato que nos ofrecen un mes de prueba gratuito, pero pasados este momento es necesario disponer de una versión de pago, pero el precio para lo que buscamos realizar ronda entre unos 0.03 y 0.06 centavos de dolar cada unas mil palabras aproximadamente, de este modo es asequible por parte del proyecto.

Para integrar esta funcionalidad bastan con crearse una cuenta en OpenAI, tras ello buscaremos el apartado donde ponga Api, y solicitaremos nuestra api key, que será la que nos permita el acceso a realizar peticiones a la api de OpenAi, de este modo con esto, realizaremos el código pertinente en el back-end, en nuestra api, estas son las funciones que mencionamos que detallaríamos posteriormente en la parte del back-end, pues llegados a este momento, desde la parte de angular en el momento que lanzamos la llamada a la api, primer realizamos en pantalla una espera activa, de modo que se quede la pantalla esperando la respuesta, eso en primer lugar, ya que al ser un producto relativamente nuevo, y no ser una simple búsqueda en base de datos de un dato, tardará entre unos 30-50 segundos la búsqueda, una vez la espera ha sido iniciada, se llama a la función de nuestra api que conecta con la api de OpenAI, en primer lugar se realiza la conexión con esta, aportando nuestra clave de acceso, tras ello simplemente se escriben los comandos necesarios, indicándole como parámetro el mensaje que le queremos preguntar, nuestro caso lo que le solicitamos es que busque cierta información del ave “nombre del ave”, pasándole desde el front-end el nombre del ave a buscar. Una vez hemos obtenido la respuesta con los datos, los tratamos y los mostramos en un pop-up.

A continuación podemos ver la pantalla de espera una vez hemos realizado la petición de los datos en la figura 2.24 y la ventana que emerge en la pantalla con la información recibida del ave en la figura.

En la figura 2.26 mostramos a donde nos lleva el primer enlace de la figura 2.25 y en la figura 2.28 mostramos el segundo enlace de la ventana emergente, que nos mostrará cierta información del ave en una página bastante completa sobre aves.

2.25

TFG Tablas Gráficas Documentación

Tabla de registros de aves

| Entrada | Nombre Común Ave | Código de especie | Más información |
|---------|--|-------------------|-----------------|
| 1 | Common Name | Species Code | Más información |
| 2 | Power tools | powtoo | Más información |
| 3 | Engine | engine | Más información |
| 4 | Human vocal | humvoc | Más información |
| 5 | Common Wood-Pigeon | cowpig1 | Más información |
| 6 | White-tipped Dove | whtdov | Más información |
| 7 | Estamos buscando algo de información de este ave, espera unos segundos por favor | | Más información |
| 8 | Picazuro Pigeon | picpig2 | Más información |
| 9 | Human non-vocal | humnov | Más información |
| 10 | | | Más información |

< Anterior 1 2 Siguiente >

Volver

Figura 2.24: Pantalla una vez iniciada la espera

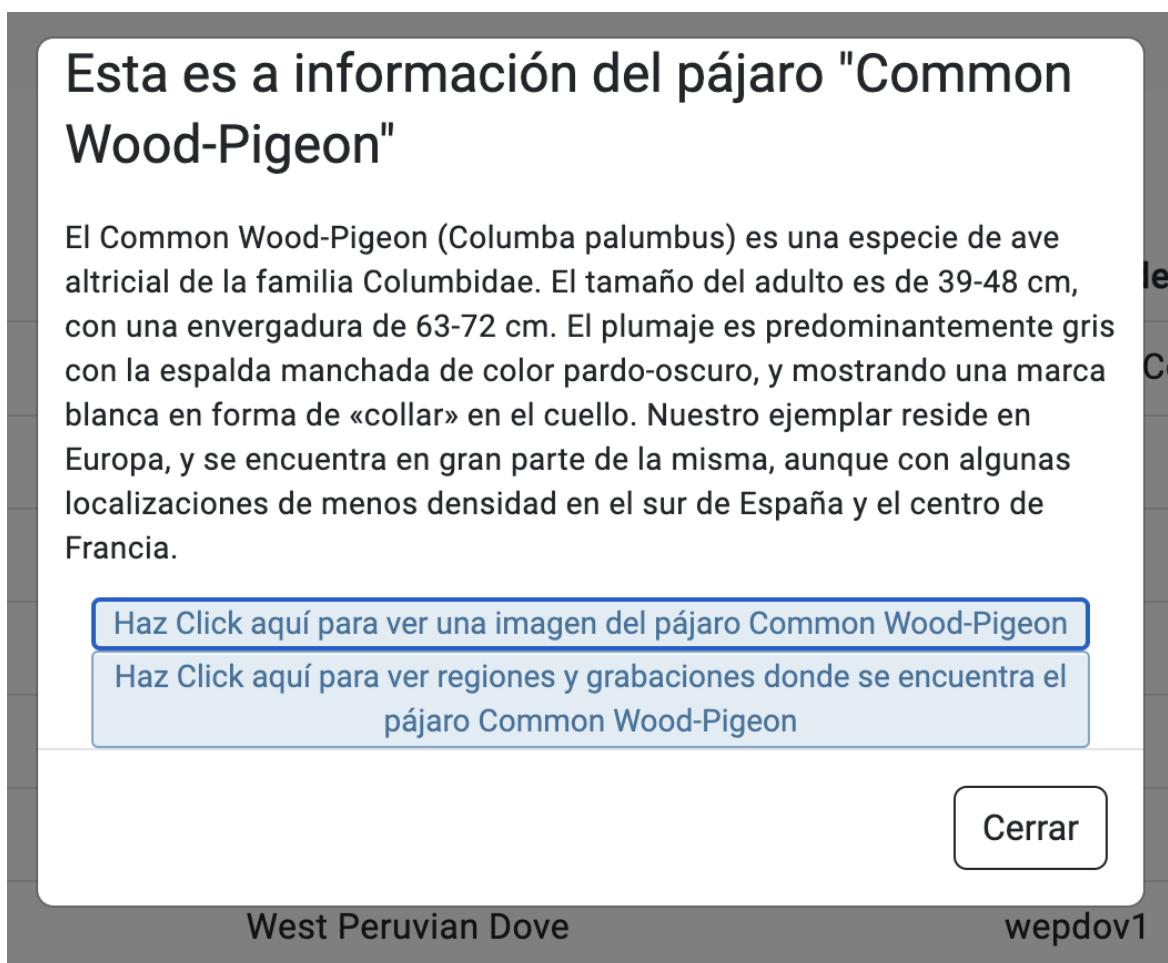


Figura 2.25: Tarjeta emergente con la información del ave



Figura 2.26: Contenido mostrado por el primer enlace de la ventana emergente



Figura 2.27: Contenido mostrado por el segundo enlace de la ventana emergente

Tras mostrar todo el contenido de la página donde se encuentra la tabla de aves, pasamos a mostrar el otro enlace de la figura 2.23, este enlace nos lleva a la pantalla en la que encontraremos la tabla de identificaciones del sistema.

| Entrada | Nombre Común Ave | Confianza | Fecha y Hora | Eliminar |
|---------|--------------------|-----------|----------------------|---------------------------|
| 3 | Human vocal | 0.1048 | 03/09/2023, 15:53:52 | <button>Eliminar</button> |
| 4 | Picazuro Pigeon | 0.1691 | 03/09/2023, 15:54:05 | <button>Eliminar</button> |
| 5 | Rock Pigeon | 0.1140 | 03/09/2023, 15:54:05 | <button>Eliminar</button> |
| 6 | Common Wood-Pigeon | 0.1771 | 03/09/2023, 15:54:05 | <button>Eliminar</button> |
| 7 | West Peruvian Dove | 0.1619 | 03/09/2023, 15:54:05 | <button>Eliminar</button> |
| 8 | White-tipped Dove | 0.1039 | 03/09/2023, 15:54:05 | <button>Eliminar</button> |
| 9 | Engine | 0.5527 | 03/09/2023, 15:54:18 | <button>Eliminar</button> |
| 10 | Rock Pigeon | 0.2380 | 03/09/2023, 15:54:18 | <button>Eliminar</button> |
| 11 | Human vocal | 0.1668 | 03/09/2023, 15:54:18 | <button>Eliminar</button> |
| 12 | Rock Pigeon | 0.4042 | 03/09/2023, 15:54:18 | <button>Eliminar</button> |

« Anterior 1 2 3 Siguiente »

Figura 2.28: Página correspondiente a la tabla de registros del sistema

En ella apreciamos al igual que en la anterior página la presencia de la tabla con los registros, esta no tiene la columna de código de especie y tiene dos nuevas como son la confianza y la fecha y hora, además de tener el botón de eliminar, además igual que en la anterior pantalla tenemos también los elementos de paginación y el botón de volver que en este caso está más abajo. El botón de borrar nos proporciona como su nombre indica la funcionalidad de borrar un registro de la base de datos, y hacerlo de manera persistente, una vez pulsamos este botón se nos abrirá una ventana emergente en la pantalla de aviso, para confirmar si queremos o no borrar el registro, podemos verla en la figura 2.29

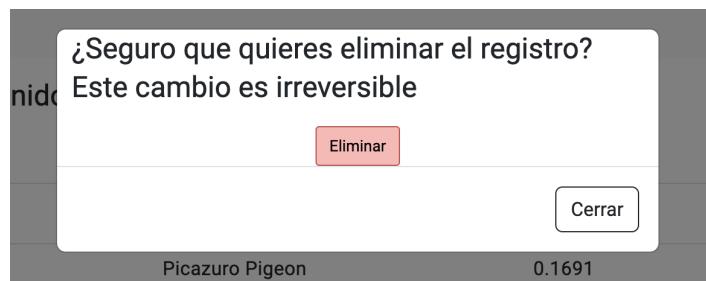


Figura 2.29: Ventana emergente con aviso tras pulsar el botón de eliminar

Tras mostrar estas dos pantallas, pasamos a otra funcionalidad de la aplicación, clicando en el enlace de Gráficas en la barra de navegación, nos llevará a la pantalla de manera análoga a la pantalla de las tablas, esta nos mostrará los diferentes enlaces para navegar hacia las diferentes gráficas de la aplicación. Estas gráficas son gráficas dinámicas, es decir, en el caso de que cambien los datos no es necesario actualizar los datos de la gráfica ni nada por el estilo, sino que estos se actualizan automáticamente, por ejemplo, en el caso que eliminemos un registro con el botón de eliminar que vimos anteriormente, y si luego visitamos la gráfica que muestra datos sobre los registros, el dato correspondiente al registro eliminado ya no aparecerá, de modo que siempre guardamos una consistencia con los datos que tenemos y mostramos. Para el desarrollo de las gráficas se ha usado canva.js una herramienta para poder realizar gráficas dinámicas en angular y muchos más frameworks.

A continuación mostramos una gráfica de linea donde podemos ver los análisis realizados por la aplicación que se obtuvieron en distintas fechas y horas

Procedemos a la muestra una gráfica de tipo tarta con los registros de las aves del sistema.

[Análisis de la aplicación](#)

[Aves encontradas](#)

[Volver](#)

Figura 2.30: Pantalla de enlace a las gráficas

Podemos apreciar en la figura 2.31 una gráfica de tipo líneas donde se muestran el número de registros en función del día y la hora.

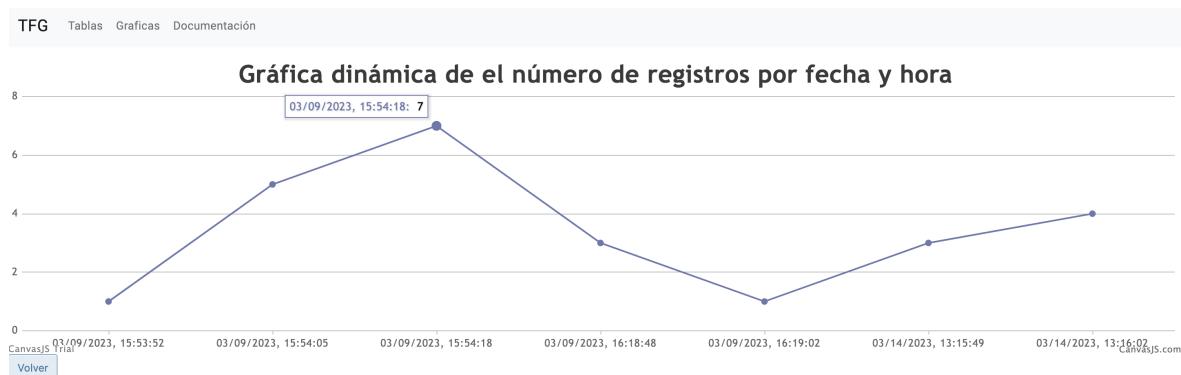


Figura 2.31: Pantalla con la gráfica resumen de los registros

Podemos apreciar en la figura 2.32 una gráfica de tipo tarta, donde se muestran el tipo de ave que se ha identificado y el número de estas, en el parte inferior se muestra el total de aves identificadas, aportamos las gráficas a la aplicación para que se pueda ver un resumen general de los datos de la aplicación de una manera visual y sencilla, consiguiendo esto gracias a las gráficas dinámicas. A medida que movemos el cursor por las diferentes porciones del gráfico nos muestra la venta aque vemos con el nombre del ave y el número exacto de veces que ha sido identificado.

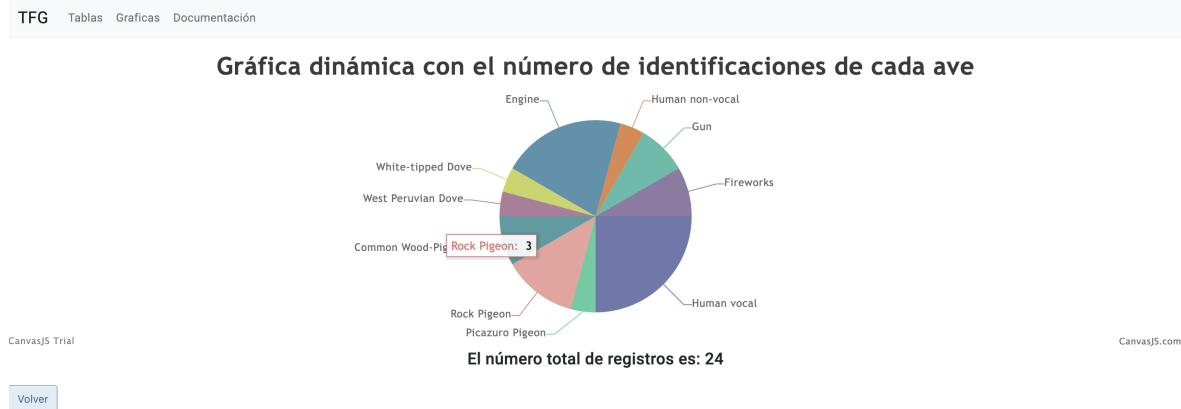


Figura 2.32: Pantalla con las aves identificadas y el número total

Por último tenemos la pantalla de Documentación, donde encontraremos un enlace a una colección de postman donde se muestran las llamadas que podemos realizar a la api y se muestra el correcto funcionamiento de estas, encontraremos además un enlace que nos permite acceder a la memoria del proyecto.



Figura 2.33: Pantalla con los enlaces a la colección de postman y a la memoria del proyecto.

2.4. Pruebas del sistema

Para comprobar el correcto funcionamiento del sistema se realizaron diversas pruebas en función al módulo a probar.

Modelo de inteligencia artificial Las primeras pruebas que se realizaron en el sistema fueron con el modelo de inteligencia artificial, el cual una vez descargado e instalado las herramientas necesarias se probó que funcionara. En primer lugar se probó el funcionamiento de este con el fichero de audio de prueba, ejecutando el comando ‘python analyze.py’ para ejecutar el fichero principal podíamos ver como en la carpeta example escribía los resultados de manera correcta en el fichero de texto.

Script para la captura del audio Una vez que este script estaba desarrollado completamente, se procedió a la ejecución del mismo, las pruebas eran bastante sencillas ya que bastaba con realizar dos comprobaciones, una que nos confirmase que se había grabado el audio que se había producido, por ejemplo decir hola mientras se graba el audio y escucharlo posteriormente comprobando que realmente se había dicho hola, tras ello la otra comprobación era ver si el fichero había sido guardado en el sitio correcto, debido a que de esto dependía que pudiera ser leído por el modelo de inteligencia artificial.

Tras ello se procedió a ejecutar el modelo de inteligencia artificial tras modificar la ruta del fichero de audio que debía leer este, tras observar como leía correctamente este archivo de audio, ya la prueba tenía resultado exitoso.

Base de datos

Para comprobar el correcto funcionamiento del script de base de datos realizamos las siguientes pruebas:

- Comprobar que se crea la base de datos si no existía previamente.
- Comprobar que se crean las tablas si no existen.
- Comprobar que no dan errores si existen las tablas o la base de datos al intentar crearlas
- Comprobar que se escriben los datos en las tablas correctamente.
- Comprobar que las columnas de las tablas son las esperadas.
- Comprobar que no se producen duplicaciones innecesarias.
- Comprobar la persistencia de los datos en la base de datos

Para las dos primeras pruebas nos bastó con comprobar en el caso de que no existiera la base de datos, que se creaba correctamente y sin errores y lo mismo para las tablas. Para comprobar que no se daban errores al intentar crear las tablas si existía realizamos pruebas intentando crear las tablas existiendo, tras no obtener ningún error en ese caso, se dio la prueba por válida, además fue simple, el motivo por el que no obteníamos error era debido a la sentencia usada en el caso de crear las tablas que eran que se crearan si no existían. Para comprobar que se escribían los datos correctamente bastaba con ver los datos del fichero de resultados de la carpeta de

example y tras ejecutar el fichero de base de datos ver si se escribía como debían. Finalmente y tras comprobar que todo se realizaba correctamente se le dio el visto bueno al módulo y se pasó al siguiente punto.

Ejecución iterativa del sistema Para comprobar la ejecución iterativa del sistema bastaba con comprobar que realmente se ejecutaban los tres scripts anteriores de manera secuencial e iterativa, hasta que el tiempo del temporizador se agotaba.

Back-end Para comprobar el correcto funcionamiento del back-end se realizaron varias pruebas, estas se realizaron realizando llamadas a las direcciones necesarias, contemplando que los códigos de estados fueran los esperados(200 ,OK), de modo que se realizó una colección en postam para comprobar el correcto funcionamiento de este módulo, la colección se encuentra en la siguiente url <https://documenter.getpostman.com/view/19481636/2s93Xzxhd7>

Front-end Para comprobar el correcto funcionamiento del front-end se realizaron varias pruebas en función de que estuvíramos probando. En primer lugar, tras inicializar el proyecto se comprobó que este funcionaba, bastaba con desplegar el servidor en local con el comando “ng serve -o” que nos abriría una ventana del navegador con el proyecto web. Las siguientes pruebas venían tras el desarrollo de las nuevas funcionalidades, y a medida que se iban desarrollando cosas se iban probando, en el caso del front-end es sencillo ya que a medida que editamos el código html, js o css podemos ir viendo los cambios que surgen, por ejemplo, al crear un botón podemos ver si tiene el estilo deseado o no, en caso de no cumplir con el estilo requerido, lo modificamos y comprobamos nuevamente. Se realizaron pruebas más exhaustivas para componentes como el de las gráficas donde debía ser comprobado si los datos mostrados en la gráfica se correspondían con los existentes en base de datos, para comprobar esto bastaba con contar a simple vista cuantos datos se había procesado para la misma fecha, cierto es que esto se realizo en primer momento a simple vista dado que es un proyecto inicial y pequeño, en el caso de contar con miles de datos no se habría realizado de este modo sino que se habría hecho una consulta sql que realizase un conteo de los datos mientras que la fecha y hora sea igual a la indicada, se detalla a continuación una consulta sql de ejemplo “SELECT COUNT * FROM TABLA WHERE FECHA = 17/12/2021”. Otra funcionalidad bastante probada fué la de ejecutar el sistema desde un botón en el front-end, para ello se pusieron puntos de depuración desde el navegador para ver cuando se ejecutaban los eventos js como en el editor de código y en el código js del back-end se pusieron múltiples impresiones por pantalla para ir observando los datos que tenemos y como se actualizan estos.

Conclusiones de las pruebas En resumidas cuentas se han realizado un gran número de pruebas hasta que el resultado alcanzado ha sido el esperado, además se han realizado múltiples tipos de pruebas, desde programas especializados para pruebas como puede ser postman hasta simples impresiones por pantalla para ver los resultados de cada cierto momento de la ejecución, con esto no significa que unas pruebas sean mejores que otras, sino que dependiendo del momento de las circunstancias y muchos mas factores, cada desarrollador usará un tipo de pruebas diferentes, pero esto no determina que una prueba sea mejor que otra, las pruebas son un paso más hacia alcanzar los objetivos esperados y la calidad del sistema, y el propio sistema final y su uso nos desvelarán si se realizaron pruebas y si estas fueron realizadas de

manera correcta.

3. Planificación del proyecto

Tras la primera reunión del proyecto, se establecieron las tareas que debían desarrollarse para satisfacer los requisitos, junto con una planificación inicial detallada de las tareas, las fechas de entrega y la estimación temporal de estas mismas, estas tareas son las que detallamos en las primeras secciones de este documento, de todas formas, se detallarán a continuación todas las tareas a realizar en el proyecto en una planificación inicial. La peculiaridad de esta planificación es que se dividirá en dos grandes hitos, el primero con la realización de lo que es el sistema interno, lo que engloba al modelo de inteligencia artificial, y el otro hito se corresponderá con la parte de la aplicación web.

- Reunión 0, establecimiento de los objetivos, las tareas y la planificación.
- Estudio de las tecnologías actuales para el desarrollo del proyecto, modelo de IA, etc.
- Estudio y pruebas del modelo de IA.
- Script para la captura del audio del micrófono.
- Creación de la base de datos.
- Inserción de los datos en la base de datos.
- Reunión 1, se estableció esta primera reunión a modo de revisión de las tareas anteriores, además de para ir viendo el avance del proyecto en sí.
- Script que permita la ejecución iterativa en segundo plano.
- Pruebas a todos los sistemas una vez se hayan acoplado
- Reunión 2, para revisar las tareas, el correcto funcionamiento de todos los módulos y poder dar el pase a la siguiente fase.
- Aquí finaliza el primer hito.
- Desarrollo de un API que consuma los datos de la base de datos para el posterior uso de un front-end.
- Reunión 3, para comprobar si la api funciona de la manera esperada, ya que es crucial para el posterior uso de los datos además de dar algunas pinceladas más de ayuda al desarrollo del front-end.
- Proyecto angular, inicializarlo y realizar la estructura base del proyecto,(configuración, barra de navegación, pie de página, etc)
- Desarrollo de la página de inicio.
- Desarrollo de las páginas con las tablas de datos.
- Desarrollo de las páginas de gráficas.

- Reunión 4, esta reunión se establece con el objetivo de revisar el trabajo desarrollado y añadir más tareas en función de lo que requiera el proyecto.
- Desarrollo del botón que ejecute el sistema.
- Desarrollo de funcionalidades como eliminar el registro u obtener más información
- Desarrollo de la pantalla de documentación del sistema.
- Realización de pruebas de funcionamiento.
- Reunión 5, para comprobar que se han completado todas las tareas de manera exitosa, y la aplicación cumple con lo esperado.
- Desarrollo de la memoria del proyecto.
- Desarrollo y preparación de la presentación del proyecto.
- Reunión 6, para revisar la memoria, corregir errores y terminar de preparar la entrega y la presentación.

La organización del proyecto se realizará con MS Project, comenzaremos modificando el calendario base del proyecto añadiendo a este los días festivos, de vacaciones, días para estudiar otros exámenes, además de establecer la fecha de inicio, estas fechas son las siguientes:

| | Nombre | Comienzo | Fin |
|----|-----------------------|------------|------------|
| 1 | Asuntos personales | 03/10/2022 | 04/10/2022 |
| 2 | Cita con el oculista | 07/10/2022 | 07/10/2022 |
| 3 | Festivo | 12/10/2022 | 12/10/2022 |
| 4 | Día libre | 17/10/2022 | 17/10/2022 |
| 5 | Cita con el médico | 19/10/2022 | 21/10/2022 |
| 6 | Estudio para examen | 22/10/2022 | 31/10/2022 |
| 7 | Entrega proyectos | 07/11/2022 | 11/11/2022 |
| 8 | Entrega práctica | 17/11/2022 | 17/11/2022 |
| 9 | Estudio exámenes | 21/11/2022 | 25/11/2022 |
| 10 | Estudio para exámenes | 05/12/2022 | 16/12/2022 |

Figura 3.1: Días durante los cuales no se trabajará en el proyecto

| | Nombre | Comienzo | Fin |
|----|-----------------------|------------|------------|
| 10 | Estudio para exámenes | 05/12/2022 | 16/12/2022 |
| 11 | Vacaciones | 26/12/2022 | 31/12/2022 |
| 12 | Vacaciones | 02/01/2023 | 15/01/2023 |
| 13 | Exámenes | 23/01/2023 | 27/01/2023 |
| 14 | Entrega trabajos | 06/02/2023 | 10/02/2023 |
| 15 | Viaje | 06/03/2023 | 10/03/2023 |
| 16 | Examen | 16/03/2023 | 16/03/2023 |
| 17 | Semana santa | 03/04/2023 | 07/04/2023 |
| 18 | Feria de abril | 24/04/2023 | 28/04/2023 |

Figura 3.2: Días durante los cuales no se trabajará en el proyecto 2

Recalcar que el comienzo del proyecto esta datado del día 01-10-2022 pero las tareas no comenzaron hasta el día 05-10-2022 debido a fines de semana y algún día en el calendario marcado como no disponible para trabajar. Podemos ver en la siguiente figura como las fechas tanto de comienzo como de fin esperado.

Información del proyecto 'Proyecto1'

| | |
|------------------------|--------------------------------|
| Fecha de comienzo: | sáb 01/10/22 |
| Fecha de fin: | jue 16/02/23 |
| Programar a partir de: | Fecha de comienzo del proyecto |

Figura 3.3: Tareas del proyecto

Podemos observar en la siguiente figura un ejemplo de mes en el que los días resaltados son aquellos en los no se trabajará en el proyecto.

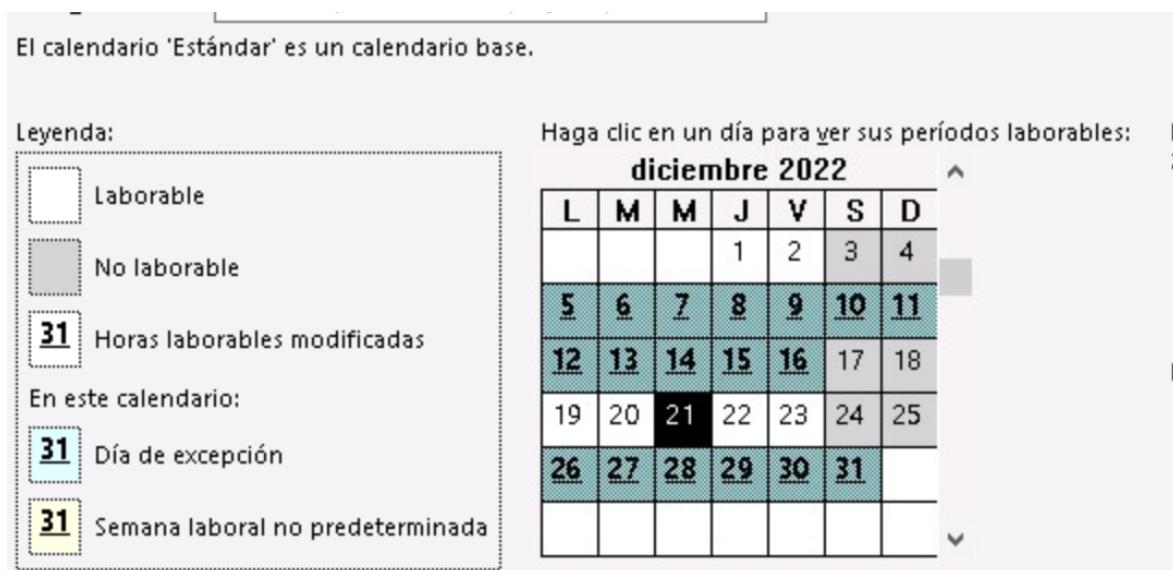


Figura 3.4: Mes de ejemplo con los días que no se trabajará en el proyecto resaltados

A continuación se mostrará el diagrama de Gantt de las tareas del proyecto, así como las fechas estimadas para su realización.

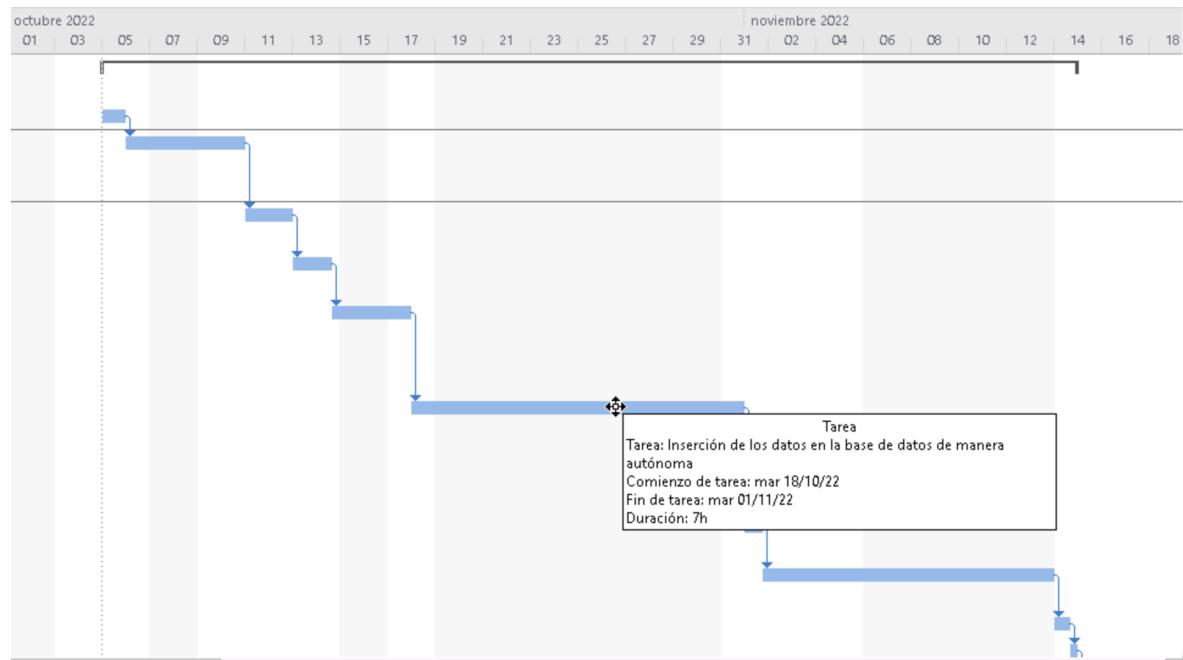


Figura 3.5: Diagrama de gant con las tareas del proyecto

En las tres siguientes figuras podemos ver las diferentes tareas que albergan tanto la fase uno como la fase dos del desarrollo del proyecto y las tareas finales previas a la entrega, junto con la duración estimada y las fechas en las que deberán realizarse dichas tareas.

| Modo de | Nombre de tarea | Duración | Comienzo | Fin | Predecesoras |
|---------|---|------------|--------------|--------------|--------------|
| ➡ | ▫ Fase 1, Sistema interno | 94,5 horas | mié 05/10/22 | mar 15/11/22 | |
| ➡ | Reunión 0 | 7 horas | mié 05/10/22 | jue 06/10/22 | |
| ➡ | Estudio de las tecnologías actuales | 15 horas | jue 06/10/22 | mar 11/10/22 | 2 |
| ➡ | Estudio y pruebas del modelo de IA | 8 horas | mar 11/10/22 | jue 13/10/22 | 3 |
| ➡ | Script para capturar el audio | 11 horas | jue 13/10/22 | vie 14/10/22 | 4 |
| ➡ | Creación de la base de datos de manera autónoma | 6 horas | vie 14/10/22 | mar 18/10/22 | 5 |
| ➡ | Inserción de los datos en la base de datos de manera autónoma | 7 horas | mar 18/10/22 | mar 01/11/22 | 6 |
| ➡ | Reunión 1, revisión | 5 horas | mar 01/11/22 | mar 01/11/22 | 7 |
| ➡ | Script ejecución iterativa | 25 horas | mar 01/11/22 | lun 14/11/22 | 8 |
| ➡ | Pruebas | 5,5 horas | lun 14/11/22 | lun 14/11/22 | 9 |
| ➡ | Reunión 2. | 5 horas | lun 14/11/22 | mar 15/11/22 | 10 |

Figura 3.6: Tareas del proyecto

| Modo de | Nombre de tarea | Duración | Comienzo | Fin | Predecesoras |
|---------|--|-----------|--------------|--------------|--------------|
| ➡ | «Fase 2, Aplicación web | 185 horas | mar 15/11/22 | lun 13/02/23 | |
| ➡ | Desarrollo API | 30 horas | mar 15/11/22 | mar 29/11/22 | 11 |
| ➡ | Reunión 3. | 6 horas | mar 29/11/22 | mar 29/11/22 | 13 |
| ➡ | Inicializar proyecto web | 10 horas | mar 29/11/22 | jue 01/12/22 | 14 |
| ➡ | Desarrollo de la página de inicio | 7 horas | jue 01/12/22 | jue 01/12/22 | 15 |
| ➡ | Desarrollo de las tablas | 35 horas | vie 02/12/22 | jue 22/12/22 | 16 |
| ➡ | Desarrollo de las gráficas dinámicas | 45 horas | jue 22/12/22 | jue 19/01/23 | 17 |
| ➡ | Reunión 4 | 5 horas | vie 20/01/23 | vie 20/01/23 | 18 |
| ➡ | Desarrollo botón ejecute sistema | 12 horas | vie 20/01/23 | mar 31/01/23 | 19 |
| ➡ | Desarrollo funcionalidades | 25 horas | mar 31/01/23 | vie 03/02/23 | 20 |
| ➡ | Pruebas | 5 horas | vie 03/02/23 | vie 03/02/23 | 21 |
| ➡ | Reunión 5 | 5 horas | vie 03/02/23 | lun 13/02/23 | 22 |
| ➡ | Desarrollo memoria proyecto | 15 horas | lun 13/02/23 | mié 15/02/23 | 23 |
| ➡ | Desarrollo de una presentación y preparación de esta | 7 horas | mié 15/02/23 | jue 16/02/23 | 24 |

Figura 3.7: Tareas del proyecto 2

| | | | | | |
|---|------------------|---------|--------------|--------------|----|
| ➡ | Reunión 6 | 6 horas | jue 16/02/23 | jue 16/02/23 | 25 |
| ➡ | Entrega proyecto | 2 horas | jue 16/02/23 | jue 16/02/23 | |

Figura 3.8: Tareas del proyecto 3

Finalmente se muestra en la siguiente figura las estadísticas del proyecto en el comienzo de este, podemos ver como fecha de inicio el 05 de Octubre de 2022, y el fin previsto de este el 16 de Febrero de 2023, y con una duración prevista y restante de unos 51.7 días de trabajo, esto equivale a 310.5 horas de trabajo.

| | Comienzo | Fin | |
|------------------------|--------------|---|--------|
| Actual | mié 05/10/22 | jue 16/02/23 | |
| Previsto | NOD | NOD | |
| Real | NOD | NOD | |
| Variación | 0d | 0d | |
| | Duración | Trabajo | Costo |
| Actual | 51,75d | 0h | 0,00 € |
| Previsto | 0d | 0h | 0,00 € |
| Real | 0d | 0h | 0,00 € |
| Restante | 51,75d | 0h | 0,00 € |
| Porcentaje completado: | | | |
| Duración: 0% | Trabajo: 0% | Cerrar | |

Figura 3.9: Estadísticas generales del proyecto antes de comenzar

A continuación se muestra cómo se verá el diagrama y las estadísticas a medida que se avanza en el proyecto, detallara que en el avance del proyecto este pude tanto reducir como ampliar las horas estimadas que tiene previsto durar, ya que podemos haber realizado algunas tareas en menos o más tiempo del estimado.

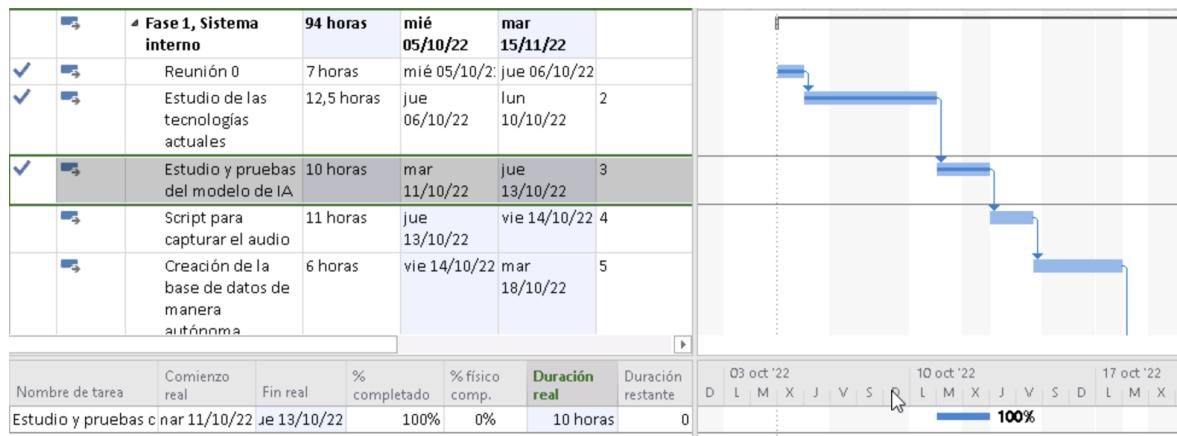


Figura 3.10: Diagrama y tareas con la realización del proyecto iniciada

Una vez hemos completado el proyecto y todas las tareas que a este engloban deberíamos de obtener un cómputo total de horas trabajadas aproximado al que estimamos a la iniciación del proyecto, de no ser así la planificación realizada no habría sido de utilidad para la consecución del proyecto, en nuestro proyecto completamos todas las taras finalizando este con un cómputo total de 305 horas, finalmente menos

Estadísticas del proyecto 'Proyecto1(1).mpp' X

| | Comienzo | Fin | |
|-----------|--------------|--------------|--------|
| Actual | mié 05/10/22 | jue 16/02/23 | |
| Previsto | NOD | NOD | |
| Real | mié 05/10/22 | NOD | |
| Variación | 0h | 0h | |
| | Duración | Trabajo | Costo |
| Actual | 307h | 0h | 0,00 € |
| Previsto | 0h | 0h | 0,00 € |
| Real | 29,31h | 0h | 0,00 € |
| Restante | 277,69h | 0h | 0,00 € |

Porcentaje completado:

| | | |
|---------------|-------------|------------------------|
| Duración: 10% | Trabajo: 0% | Cerrar |
|---------------|-------------|------------------------|

Figura 3.11: Estadísticas mientras se realiza el proyecto

de las establecidas inicialmente, pero bastante cercano a la planificación inicial que realizamos.

4. Conclusiones

Llegados a este punto contamos con un sistema final totalmente funcional, formado por una aplicación web que permite al usuario final interactuar con todo el sistema interno, esta aplicación web cuenta con gran cantidad de formas de visualizar los datos del sistema, desde tablas hasta gráficas, además de ofrecer posibilidades como borrar registros o ampliar la información de los datos, además permite ejecutar todo el sistema desde la interfaz con un simple botón, también la aplicación cuenta con su propia api que le permite obtener datos de la base de datos y hacer peticiones a la api de OpenAI. El desarrollo de la aplicación web ha servido para conocer nuevas tecnologías como son angular y typescript, el desarrollo de conexiones desde nuestra api con la api de OpenAI, también se reforzaron conocimientos en lenguajes como javascript. Esta aplicación se nutre de los datos almacenados en una base de datos relacional sqlite, la cual almacena dos tablas definidas a partir de un script python y almacenados de igual manera, de modo que aquí he aprendido a crear una base de datos sqlite desde python, así como la creación de las tablas y la inserción de los datos de la tabla desde python, de modo que he ampliado mi conocimiento en sql al realizar consultas en sql tanto para obtener los datos de la base de datos desde la api como para insertarlos en la base de datos, además se procedió al tratamiento de datos, desde la lectura de un fichero, hasta eliminar caracteres o modificar cadenas, y el uso del lenguaje python, que con el uso de las librerías es innumerable el número de cosas que podemos hacer con este lenguaje. Además se desarrolló el script que permitía capturar el audio del micrófono de nuestra máquina en lenguaje python, nuevamente con el uso de un par de librerías y aprendiendo nuevas funciones en estas librerías, fueron nuevas funcionalidades aprendidas dentro del lenguaje python.

También se desarrolló un script que permitía la ejecución de todo lo anterior en segundo plano, aprendiendo a como activar nuevos hilos de ejecución y a lanzar procesos en segundo plano con python, este fue de los códigos más interesantes bajo mi punto de vista y más útiles ya que podemos hacer infinidad de cosas con el ya que nos permite lanzar múltiples cosas en ejecución.

Todo ello se ha implementado para ofrecer datos en el caso de la captura del audio pero en la mayoría para el uso de los datos procedentes de los análisis del modelo de inteligencia artificial, este ha sido obtenido de un repositorio de github, y es cierto que ya está desarrollado, pero excedía las horas para este proyecto realizar cierto trabajo, además, si ya existe, podemos usarlo incluso mejorarlo, de este modo que le hemos añadido toda esta funcionalidad extra, pese a no haber desarrollado este código por mí, he aprendido bastante de él, ya que previamente a decidir que iba a ser usado en el proyecto se realizó un estudio de las características de este, cómo realizaba el tratamiento de los datos y como podíamos dotar a un sistema de todo lo necesario partiendo de los datos de este modelo.

Finalmente recalcar que he disfrutado bastante con la realización de este proyecto y dar mención Francisco Luna por su ayuda en algunos momentos de atasco en el desarrollo del proyecto y su ejemplar tutoría durante todo el proyecto. Animo a todo

aquel que le interese este mundo del desarrollo a echarle un vistazo al proyecto con el que podrán aprender bastante de nuevas tecnologías y sobre todo disfrutar mientras se aprender.

5. Referencias
