Trabajo de Fin de Grado

Plataforma WEB para la visualización y análisis de campañas de acústica pasiva con planeadores submarinos

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Sara Sánchez García

TUTORIZADO POR:
Jorge Cabrera Gámez
Antonio C. Domínguez Brito

Fecha 07/2024

Agradecimientos

Quiero expresar mi profundo agradecimiento a todas las personas que me han apoyado y guiado a lo largo de mi carrera universitaria y en la realización de este proyecto.

En primer lugar, agradezco enormemente a mi familia y amigos por su constante apoyo y aliento a lo largo de estos años. Su respaldo ha sido fundamental en cada paso que he dado.

También deseo agradecer a mis tutores, Jorge Cabrera Gámez y Antonio C. Domínguez Brito, por su invaluable orientación y asistencia durante el desarrollo de este proyecto. Sus conocimientos y consejos fueron cruciales para su éxito.

Por último, quiero reconocer a los miembros de la comunidad de la Escuela de Ingeniería Informática. Agradezco a todos los profesores, personal y compañeros alumnos que han compartido su conocimiento y han contribuido a mi formación profesional hasta este punto.

Su apoyo ha sido fundamental y estoy profundamente agradecida por haber tenido la oportunidad de aprender y crecer junto a todos ustedes.

Resumen

Este Trabajo de Fin de Título (TFT) es una herramienta gráfica que se centra en facilitar el análisis de datos generados por campañas de monitoreo acústico pasivo (PAM) en las que se utilizan planeadores subacuáticos (gliders) para identificar señales asociadas a especies específicas de cetáceos.

La herramienta gráfica integra y exhibe de manera efectiva la información obtenida. Esta herramienta proporciona vistas complementarias que fusionarán datos acústicos con información de navegación del planeador.

Asimismo, proporciona vistas para comparar los descriptores de señales particulares con los descriptores promedios de señales en una misma clase.

Abstract

This Final Degree Project (TFT) is a graphical tool that focuses on facilitating the analysis of data generated by passive acoustic monitoring (PAM) campaigns in which underwater gliders are used to identify signals associated with specific cetacean species.

The graphical tool effectively integrates and displays the information obtained. This tool provides complementary views that will merge acoustic data with glider navigation information.

It also provides views to compare particular signal descriptors with average signal descriptors in the same class.

Índice general

Índice de figuras

Índice de Algoritmos

Capítulo 1

Introducción

Este Trabajo de Fin de Título (TFT) es una herramienta gráfica que se centra en facilitar el análisis de datos generados por campañas de monitoreo acústico pasivo (PAM) en las que se utilizan planeadores subacuáticos (gliders) para identificar señales asociadas a especies específicas de cetáceos. “Ocean sounds” es el nombre elegido para la aplicación.

La herramienta gráfica integra y exhibe de manera efectiva la información obtenida. Este proyecto proporciona vistas complementarias que fusionan datos acústicos con información de navegación del planeador.

Asimismo, proporciona vistas para comparar los descriptores de señales particulares con los descriptores promedios de señales en una misma clase.

1.1. Contexto del proyecto

En la actualidad, la realización de campañas de monitoreo con acústica pasiva (PAM) por medio de planeadores subacuáticos (gliders) genera una enorme cantidad de datos que deben ser procesados para identificar las señales que se asocian a especies específicas de cetáceos. El Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI) posee un planeador subacuático de la clase SeaExplorer[1] con el cual se han realizado diferentes campañas, recogiendo principalmente dos tipos de ficheros en bruto: ficheros de audio y ficheros de navegación del glider. Tras el análisis de los datos, las detecciones quedan almacenadas en un conjunto relativamente disperso de ficheros de etiquetas, descripciones espectrales y segmentos de datos primarios, los cuales están asociados de forma unívoca con los intervalos de datos de los ficheros de audio registrados durante una campaña.

1.2. Problema planteado

El problema surge al intentar relacionar los datos referentes al glider, como la profundidad en la que se encontraba, con los datos referentes a los clics encontrados en una muestra de audio. Actualmente, aunque existen diversas herramientas específicas y de uso general para visualizar datos del glider, ninguna ofrece la capacidad de integrar esta información con los datos de audio en una única interfaz.

Es así como surge la idea de este TFT, la realización de una herramienta gráfica que permita presentar toda la información generada en una serie de vistas complementarias que integren los datos resultantes de la descripción acústica de las señales con los datos de navegación del planeador. La aplicación no se plantea como una herramienta sustituta de las existentes sino a una herramienta complementaria que aporte información de manera visualmente agradable e inteligible y facilite el análisis y la interpretación de los datos.

1.3. Descripción del proyecto

El proyecto consiste en la creación de una interfaz web clara e intuitiva que permite analizar campañas submarinas de manera más cómoda y completa. Además, cuenta con una API (interfaz de programación de aplicaciones) encargada de tratar los datos para lograr una mayor velocidad en la aplicación web, incluso al tratar grandes volúmenes de información.

La interfaz web se compone de dos vistas distintas: la vista general y la vista detallada. Ambas poseen una estructura similar, aunque la vista general se enfoca en mostrar datos de un fichero completo y la vista detallada en los detalles de una sola detección.

En la vista general, primero se presenta un selector de fichero con el que el usuario puede elegir con qué fichero trabajar. Una vez elegido el fichero, aparece una pantalla que contiene una tabla con datos estadísticos de los descriptores, un mapa que muestra las detecciones, y una gráfica que muestra la relación entre el número de detecciones y la profundidad. Además, en la parte superior se muestra el título, la fecha del fichero y dos botones que permiten cambiar de vista y de fichero.

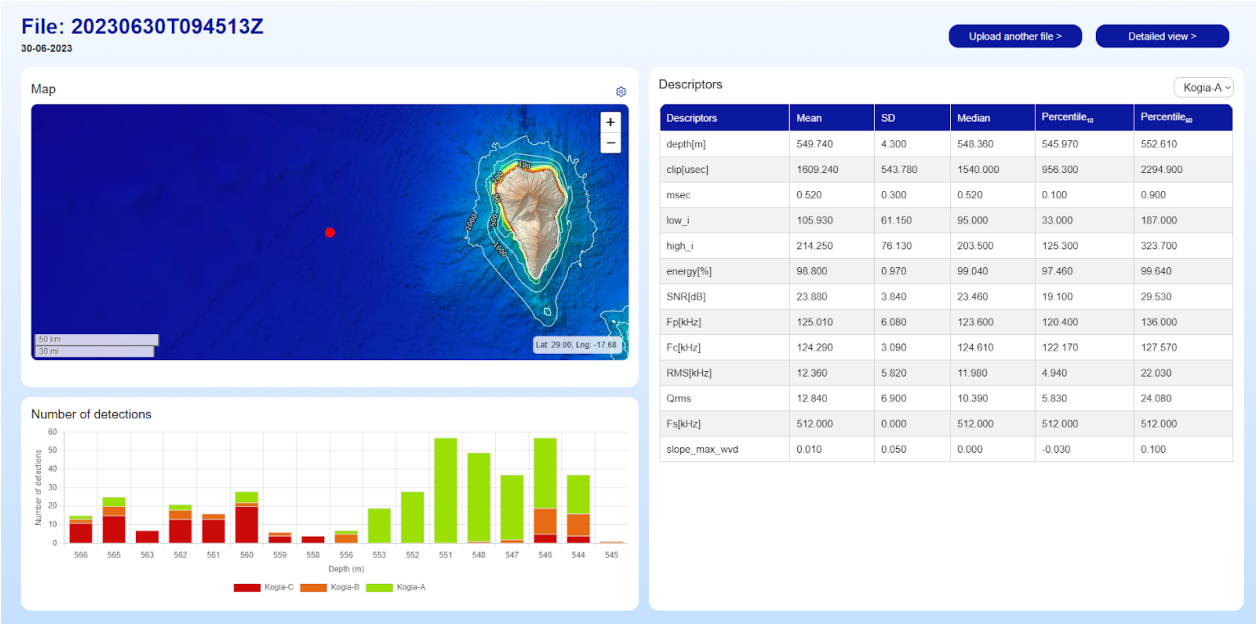

Ilustración 1.1: Vista general de la aplicación web

En la vista detallada, primero se presenta un selector de detección. Una vez elegida la detección, se pueden observar la localización en el mapa, los descriptores propios de la detección, la distribución de Wigner-Ville que es una representación conjunta en el dominio tiempo-frecuencia y la representación de la onda. Al igual, que en la vista detallada, se muestra el nombre del fichero, la fecha, la hora y botones que permiten cambiar de detección.

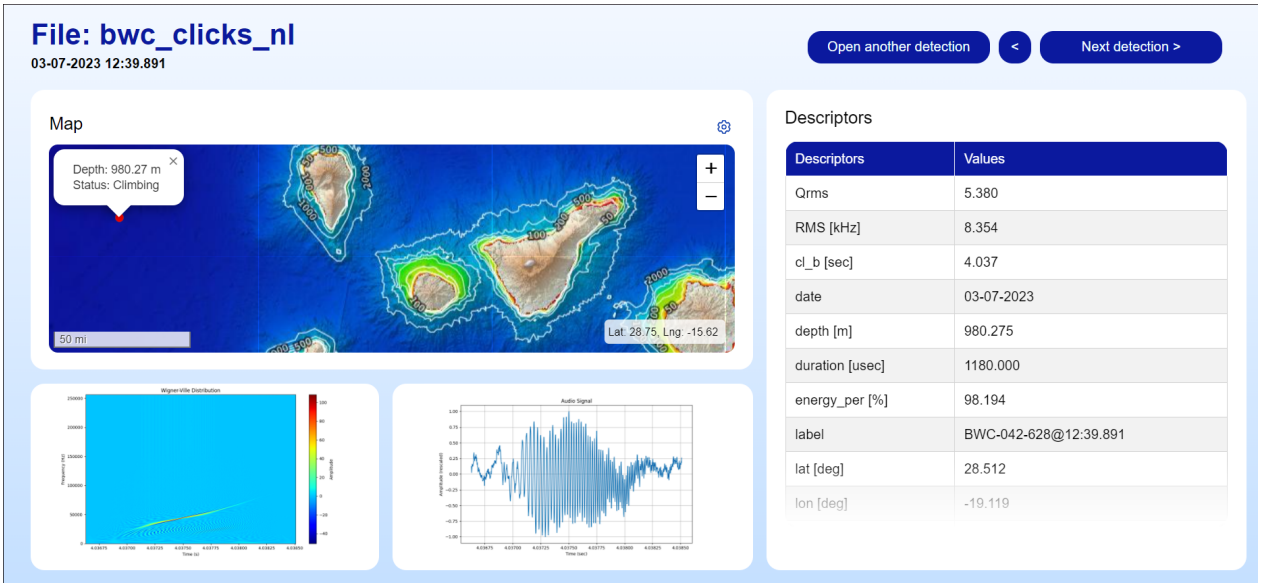

Ilustración 1.2: Vista detallada de la aplicación web

La web ha sido pensada para que sea posible tener la pestaña general siempre abierta y

a la vez las pestañas con vista detallada, lo que hace posible trabajar con varias pantallas y comparar una detección con otra o con los valores generales.

1.4. Objetivos del proyecto

- ✔ Una interfaz sencilla e intuitiva.
- ✔ Visualmente ergonómica y de fácil manejo, incluso cuando se trabaja con un gran volumen de datos.
- ✔ Adaptabilidad a diferentes tamaños de pantallas de ordenadores.
- ✔ Velocidad en la carga de datos.
- ✔ Versatilidad con los ficheros de datos con los que puede trabajar.
- ✔ Utilidad a nivel científico y didáctico.

En el siguiente capítulo “Estado y objetivos iniciales” se detalla en profundidad dichos objetivos.

1.5. Competencias específicas

En este TFT se utilizarán diferentes tecnologías sin la interacción de las mismas no sería posible la realización del proyecto. Así que las competencias específicas que mejor se adecuan al proyecto son:

CI1 Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

CI7 Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema

TI6 Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

1.6. Organización del documento

El documento se organiza en diversos capítulos, cada uno de los cuales aborda un aspecto fundamental del proyecto. A continuación se describen los capítulos en detalle:

1. **Estado Actual y Objetivos Iniciales:** Aquí se detalla diferentes herramientas para la visualización de datos y se especifican los objetivos iniciales planteados. Se proporciona una visión clara del punto de partida del proyecto y de las metas que se pretenden alcanzar.

2. **Competencias Específicas:** En este capítulo se enumeran y describen las competencias técnicas y académicas desarrolladas a lo largo del proyecto. Estas competencias están directamente relacionadas con los objetivos y el desarrollo del TFT.

3. **Herramientas Utilizadas:** Se describen las herramientas tecnológicas y metodológicas empleadas durante todo el proceso de elaboración del TFT. Este capítulo proporciona una visión detallada de los recursos utilizados y su relevancia para el proyecto.

4. **Diseño:** En este capítulo se explican las decisiones de diseño tomadas a lo largo del proyecto. Se justifica cada elección de diseño basándose en criterios técnicos y funcionales, proporcionando una comprensión profunda del proceso de diseño.

5. **Desarrollo:** Como su nombre indica, este capítulo narra el proceso de desarrollo del TFT. Se detallan las etapas de implementación, los desafíos encontrados y las soluciones adoptadas durante el desarrollo del proyecto.

6. **Estado final:** En este capítulo se detalla el estado final de la aplicación.

7. **Conclusiones y Trabajo a Futuro:** Este capítulo final resume las principales conclusiones del TFT, destacando las aportaciones del trabajo realizado. Además, se proponen líneas de trabajo futuro, sugiriendo posibles mejoras y expansiones del proyecto basadas en los resultados obtenidos.

Capítulo 2

Estado actual y objetivos iniciales

El proyecto ha transcurrido por distintas fases. En este capítulo se detallan otras herramientas utilizadas para la visualización de datos PAM y se detallan los objetivos iniciales cubiertos en el proyecto final.

2.1. Otras herramientas para la visualización de datos

Actualmente, las herramientas disponibles para la visualización de datos PAM (Monitoreo Acústico Pasivo) se pueden dividir en dos categorías: herramientas de uso general y herramientas especializadas.

En primer lugar, se encuentran las herramientas de uso general, como Audacity, Matlab y Raven Pro. Audacity [2] es una herramienta dirigida a cualquier persona interesada en trabajar creativamente con registros digitales de sonido. En las campañas de Monitoreo Acústico Pasivo (PAM), se utiliza para el análisis de espectrogramas y oscilogramas, así como otros datos relacionados con el análisis gráfico de los audios recolectados. Sin embargo, a pesar de su utilidad, Audacity no está optimizado para el uso científico en campañas submarinas debido a su enfoque general y la falta de funcionalidades específicas para el análisis de datos PAM.

Raven Pro [3] es un programa de software diseñado para la adquisición, visualización, medición y análisis de sonidos. Este software ofrece una herramienta potente y fácil de usar tanto para la investigación como para la enseñanza, dirigida a científicos que trabajan con señales acústicas. Raven Pro se distingue por sus vistas altamente configurables, las cuales proporcionan una flexibilidad sin igual en la visualización de datos, permitiendo adaptarse a diversas necesidades de análisis y estudio.

Matlab[4] no es en sí misma una herramienta de análisis. Eso sí, algunos paquetes muy usados (Triton, p.e.) están desarrollados en MATLAB. En las campañas PAM, las herramientas que aporta Matlab son muy útiles para la manipulación de datos recolectados durante el

monitoreo. No obstante, su uso no es sencillo ni intuitivo; requiere conocimientos en programación y puede resultar complejo para aquellos sin experiencia técnica.

Por otro lado, existen herramientas especializadas en datos PAM, entre las que destacan Triton, PamGuard y PAMlab INT.

Triton [5] es un paquete de Matlab especializado en el análisis y visualización de datos acústicos submarinos. Es una plataforma robusta diseñada para abordar los desafíos específicos del Monitoreo Acústico Pasivo (PAM). Sus capacidades en análisis espectral, detección de eventos e interfaz fácil de usar la convierten en una herramienta valiosa para investigadores

PamGuard[6] es una plataforma de software de código abierto diseñada para la detección, localización y clasificación de sonidos marinos, particularmente los emitidos por mamíferos marinos como los cetáceos. Es ampliamente utilizada en la investigación y monitoreo acústico marino, contribuyendo significativamente a la conservación de la vida marina y la mitigación de impactos ambientales. Desarrollada en Java, es un proyecto de larga tradición dirigido por Douglas Gillespie de la Universidad de St. Andrews (Escocia).

PAMlab INT[7] es una aplicación avanzada para la detección automatizada y el procesamiento de datos acústicos. Se integra eficientemente en sistemas de monitoreo y análisis, permitiendo un análisis inteligente a bordo de plataformas y vehículos remotos.

2.2. Objetivos iniciales

Este TFT propone una herramienta complementaría e innovadora en la visualización de datos recogidos por campañas submarinas. El nombre “Ocean sounds” se eligió puesto que la aplicación muestra datos de campañas submarinas. El usuario puede en “Ocean sounds” consultar datos referentes al glider como datos referentes a las detecciones.

Los objetivos se centran en el desarrollo de una interfaz sencilla e intuitiva, en la que el usuario sepa en todo momento que puede hacer.

2.2.1. Interfaz sencilla e intuitiva

Para el diseño de la interfaz gráfica se han tenido en cuenta consejos y reglas que otros científicos han estudiado con anterioridad y demuestran su eficacia. Para conseguir este resultado se han aplicado las reglas de oro de Ben Shneiderman[8], se ha tenido especial cuidado con la elección de colores siguiendo la teoría del color y se han tomado inspiraciones de las ideas propuestas por Edward Tufte [9] para el diseño de gráficos y exposición de gran volumen de datos

En el diseño de esta aplicación se ha priorizado la utilidad y practicidad pero teniendo muy en cuenta el diseño.

2.2.2. Adaptabilidad

La adaptabilidad a diferentes tamaños de pantalla de ordenador es un objetivo principal permitiendo que las aplicaciones sean accesibles y funcionales. Para alcanzar este objetivo, se ha aprovechado las capacidades de adaptabilidad ofrecidas por varias bibliotecas y tecnologías clave en el proyecto.

Se eligió la biblioteca Chart.js[10] para la creación de gráficos interactivos y responsivos. Chart.js es conocida por su flexibilidad y facilidad de integración con React[11], lo que permite crear gráficos que se ajustan automáticamente al tamaño de la ventana del navegador.

Además de Chart.js, también se planeó utilizar Leaflet[12] para la visualización de mapas y React-Super-Responsive-Table[13] para tablas dinámicas y responsivas. Estas bibliotecas, junto con React como framework, permiten construir interfaces que se adapten fluidamente a diferentes tamaños de pantalla. React, siendo un framework basado en componentes, facilita la creación de interfaces reutilizables y eficientes, mientras que Leaflet y React-Super-Responsive-Table ofrecen soluciones especializadas para mapas y tablas que son fundamentales para nuestra aplicación.

En la vista detallada, los gráficos se planificó realizarlos con Python[14] desde la API pues dichos gráficos al tener mayor complejidad y tratar ficheros de audio no se podían realizar con Chart.js. Pero no se iba a dejar de lado la responsividad así que se desidió que el tamaño de la imagen se ajustara al tamaño de la pantalla.

En resumen, la combinación de estas herramientas y técnicas nos permite desarrollar una aplicación web altamente adaptable a una variedad de dispositivos y tamaños de pantalla, garantizando una experiencia de usuario óptima independientemente del dispositivo se utilice.

2.2.3. Velocidad de procesamiento

Otro aspecto crucial en el desarrollo de este proyecto es la optimización de la velocidad de procesamiento y visualización de datos. En este Trabajo Final de Título se busca una ejecución rápida con datos preprocesados. Para lograrlo, se ha decidió implementar una API en Python que realiza el procesamiento de datos sólo la primera vez que se ejecuta un archivo. Basándonos en pruebas realizadas, este proceso promedia menos de treinta segundos, lo que representa un valor añadido en términos de velocidad.

La visualización de gráficos más complejos como son el espectrograma y el espectro de la vista detallada necesitan mayor tiempo de procesamiento pero que en conjunto no supera los cuarenta y cinco segundos.

Esta inmediatez en la visualización de datos se debe principalmente a la eleción del uso de bibliotecas de visualización de datos que operan de manera eficiente y al marco de trabajo de React. React, conocido por su rendimiento y eficiencia, facilita la creación de interfaces de usuario interactivas y dinámicas, lo que permite una visualización de datos instantánea una vez que los datos han sido procesados. Este enfoque no solo agiliza el proceso de visualización,

sino que también mejora la experiencia del usuario al proporcionar resultados de manera inmediata y fluida.

2.2.4. Heterogeneidad de los ficheros de datos

Uno de los objetivos principales del proyecto es garantizar la versatilidad en la manipulación de archivos de datos. La API está diseñada para ofrecer flexibilidad en la selección de los descriptores que se desean mostrar, incluso cuando los archivos contienen columnas con nombres diferentes o cuando se desea agregar nuevas columnas de datos. Esto se logra mediante la especificación explicita de los nombres de las columnas relevantes, como se muestra en el siguiente fragmento de código:

```
all_descriptors = [
    'Id', 'state', 'date', 'time', 'label', 'depth[m]', 'clip[usec]', 'msec',
    'energy[%]', 'SNR[dB]', 'Fp[kHz]', 'Fc[kHz]', 'RMS[kHz]', 'Qrms',
    'Fs[kHz]', 'slope_max_wvd', 'lat[deg]', 'lon[deg]'
]
```

Algoritmo 2.1: Ejemplo de código lista de descriptores

Además, el programa está diseñado para manejar cambios en el orden de las columnas o la presencia de columnas adicionales que no estén detalladas en `all_descriptors`. Esto asegura que el programa funcione sin problemas, mostrando siempre los descriptores detallados sin importar las variaciones en la estructura de los archivos de datos.

Desde la planificación del proyecto se decidió que algunas columnas fueran obligatorias, ya que son esenciales para el funcionamiento de la aplicación. Estas son: 'Id', 'date', 'time', 'lat[deg]', 'lon[deg]', 'state' y 'depth'. La ausencia de alguna de estas columnas provocaría un error en el servidor al ejecutar el código.

Cabe resaltar que si el archivo es del tipo `_clicks`, también serían necesarias las columnas `cl_b[sec]` y `duration[usec]` para el muestreo del espectrograma y espectro. Para el resto de archivos, serían necesarias las columnas `msec` y `begin[sec]`. Las columnas `Id`, `date`, `time`, `state`, y las relacionadas con el momento de inicio y duración previamente mencionadas son necesarias para el muestreo de los gráficos y datos esenciales para la vista detallada. Por otro lado, `lat[deg]`, `lon[deg]` y `depth` son necesarias para el mostrado de los datos relacionados con el mapa y el gráfico de la vista general. Si no se especifican las unidades, se utilizan las del Sistema Internacional, por ejemplo: la profundidad (depth) en metros y el tiempo (time) en segundos.

La aplicación se planteó para manejar diferentes tipos de archivos, los cuales se diferencian por el nombre utilizando un etiquetado específico que denota ciertas cualidades. El nombre del archivo representa la fecha de la grabación del audio, por ejemplo: `24052022.csv`. Si abarca varios días, el nombre puede ser cualquier identificador significativo, como `bwc.csv`. Si el nombre contiene `_nl` (No Label), significa que el archivo tiene solo un tipo de etiqueta, lo cual puede ocurrir cuando se está analizando solo un tipo de especie de cetáceo y no se hacen diferencias en la calidad del click. La segunda etiqueta es `_clicks`, utilizada si el archivo es una unión de clicks, incluso en distintas campañas de monitoreo. Estas etiquetas

pueden utilizarse simultáneamente, resultando en un archivo que contiene clicks de diferentes campañas de una sola especie sin diferenciar en la calidad del click.

Esta característica otorga robustez y flexibilidad es fundamental para adaptarse a diferentes formatos de datos y requisitos de análisis, lo que aumenta la utilidad y la aplicabilidad de la herramienta en diversos contextos de investigación y análisis de datos.

2.2.5. Utilidad didáctica y científica

La aplicación se plantea para que los usuarios sean capaces de identificar patrones, tendencias y relaciones en los datos a través de tablas, gráficos y mapas. Esto es particularmente valioso en el ámbito científico, donde la identificación de patrones y tendencias es crucial para la formulación de hipótesis, la validación de teorías y la generación de nuevos conocimientos. El diseño de la aplicación pretende facilitar la comprensión, análisis y comunicación de datos complejos de manera eficiente, al cumplir con todos estos objetivos, la aplicación se establece como una herramienta útil a nivel científico y didáctico.

Capítulo 3

Competencias específicas y aportaciones del trabajo

En este capítulo se realiza una justificación detallada de las competencias específicas que se han desarrollado y demostrado a lo largo del presente Trabajo de Fin de título (TFT). La creación de la herramienta gráfica "Ocean sounds" no solo responde a una necesidad concreta dentro del campo del monitoreo acústico pasivo utilizando planeadores subacuáticos, sino que también ha implicado la aplicación de conocimientos avanzados en diversas áreas de la ingeniería informática.

A continuación, se explicará cómo cada competencia específica ha sido abordada en el desarrollo del proyecto, destacando la integración de distintas tecnologías, la implementación de estructuras de datos adecuadas, el diseño y construcción de una aplicación robusta y segura, así como la concepción de sistemas basados en tecnologías de red.

3.1. CI1: Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente

La competencia CI1 se cumple a lo largo de todo el desarrollo de la aplicación "Ocean sounds". Desde el diseño inicial hasta la implementación final, se han tomado decisiones cuidadosamente orientadas a asegurar la fiabilidad, seguridad y calidad del sistema.

3.1.1. Diseño y desarrollo

La arquitectura del sistema se ha diseñado para incluir una API en Python para el procesamiento de datos y una interfaz gráfica web intuitiva, desarrollada con React. Este diseño modular se ha hecho para que cada componente del sistema sea robusto y eficiente, facilitando tanto el mantenimiento como la escalabilidad. La API en Python permite un procesamiento de datos preciso y eficiente, mientras que React proporciona una interfaz de usuario interactiva y amigable.

3.1.2. Selección de Tecnología

Las tecnologías utilizadas por ejemplo, Chart.js y Leaflet para gráficos y mapas interactivos, React-Super-Responsive-Table para tablas dinámicas, buscan garantizar la adaptabilidad a diferentes tamaños de pantalla. Por lo que aseguran calidad.

3.1.3. Evaluación y Pruebas

Durante todo el desarrollo, se realizaron pruebas para asegurar que la aplicación funcione correctamente bajo diferentes condiciones.

3.2. CI7: Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema

La competencia CI7 se refleja en la selección y utilización de estructuras de datos apropiadas para el manejo y procesamiento eficiente de grandes volúmenes de información.

3.2.1. Estructuras de Datos

La API en Python utiliza estructuras de datos como DataFrames de la librería Pandas[15] para organizar y procesar los datos de forma eficiente. Esto permite una manipulación rápida y flexible de grandes conjuntos de datos.

3.2.2. Almacenamiento y Recuperación

Los datos procesados se almacenan en estructuras organizadas, permitiendo una recuperación eficiente para su visualización en la interfaz gráfica. Además, se ha diseñado el sistema para adaptarse a diferentes formatos y estructuras de datos, proporcionando versatilidad en su uso.

3.2.3. Optimización de Proceso

Se han implementado técnicas de preprocesamiento para reducir el tiempo de carga y visualización de datos, asegurando que la aplicación sea rápida y responsiva incluso al manejar grandes volúmenes de información.

3.3. TI6: Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil

La competencia TI6 se cumple a través de la creación de una aplicación web interactiva y adaptable que integra varias tecnologías de red y servicios interactivos.

3.3.1. Tecnologías Web

La aplicación "Ocean sounds" está diseñada como una aplicación web que utiliza tecnologías como React para la interfaz de usuario y Flask[16] para la API en el servidor. Esto permite una integración fluida de servicios interactivos y una experiencia de usuario eficiente.

3.3.2. Servicios Interactivos

La aplicación incluye componentes interactivos como gráficos dinámicos y mapas, proporcionando una experiencia de usuario rica y funcional. Estos elementos interactivos facilitan la exploración y el análisis de los datos de monitoreo acústico pasivo.

En resumen, el proyecto "Ocean sounds" no solo cumple con las competencias específicas definidas, sino que también proporciona una solución innovadora y práctica para la visualización y análisis de datos complejos de monitoreo acústico pasivo, integrando datos del Glider y de audio en una interfaz amigable y accesible.

Capítulo 4

Herramientas

El proyecto consta de dos partes bien delimitadas entre ellas. La API se encarga del manejo de ficheros de datos, modificación y envío de ellos. Esta parte queremos que sea rápida y adaptable a diferentes tipos de datos. Por otro lado, tenemos la aplicación web, en ella se muestran de manera legible y visualmente agradable los datos previamente adaptados por la API. La misión de esta parte es que sea sencilla, interactiva, adaptable y estética. Debido a que cada parte tiene un propósito distinto se utilizan tecnologías diferentes. Todas las tecnologías utilizadas tienen en común que constan de una comunidad activa y soporte pues se busca la mejor calidad y la escalabilidad para el proyecto.

4.1. Python

Python[14] es un lenguaje de programación de alto nivel, interpretado y de objetivo general. Se eligió como lenguaje para la elaboración de API debido a su simplicidad y legibilidad que facilita el desarrollo y mantenimiento de APIs, la inmensa cantidad de bibliotecas y marcos que existen y la gran comunidad activa, lo que significa que hay muchos recursos disponibles y soporte.

Hay otros aspectos a considerar, como la escalabilidad que brinda el framework de Flask el cual posibilita la creación de APIs escalables que se adapten a las necesidades de la aplicación. Además, aporta portabilidad pues es compatible con la mayoría de los sistemas operativos, lo que facilita la implementación en diferentes entornos. De igual forma, la facilidad en la integración es una característica ventajosa ya que es fácil de trabajar con otros lenguajes y tecnologías, lo cual es útil para crear una API que se comunique con diferentes sistemas.

Ilustración 4.1: Logo de Python

4.2. Flask

Flask[16] es un framework de Python minimalista que permite crear rápidamente aplicaciones web utilizando la menor cantidad de líneas de código posible. Flask proporciona configuración y convenciones, con valores por defecto sensatos pero muy completos. La incorporación de esta tecnología ha implicado un proceso de aprendizaje inicial, dado que no se contaba con experiencia previa en el manejo de APIs. Por tanto, Flask ha sido ideal debido a que proporciona esa sencillez y completitud necesaria para el desarrollo del proyecto.

Ilustración 4.2: Logo de Flask

4.3. Librerías de python

En lo que al desarrollo de la API se refiere cabe destacar el uso de diferentes bibliotecas que facilitan y optimizan el trabajo. A continuación se detallan:

4.3.1. Datetime

Datetime[17] es un módulo que permite operaciones aritméticas con fechas y horas, en el proyecto su objetivo fue poder extraer campos de forma eficiente para su posterior manipulación.

Se utilizó la función 'strptime()', que devuelve una cadena que representa la fecha, controlada por una cadena de formato explícita. Útil para extraer la fecha de nombre del fichero con el que se está trabajando.

4.3.2. Pandas

Pandas[15] es una librería software construida sobre el lenguaje de programación Python. Es una excelente herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar. Su función en este trabajo ha sido el tratamiento de datos. Se ha utilizado la función `read_csv()` para leer de archivos CSV y transformarlos en dataframes, estructuras tabulares de datos orientadas a columnas, con etiquetas tanto en filas como en columnas. Gracias a esta transformación la manipulación de los datos se agiliza. Asimismo, se ha utilizado la función `DataFrame()` para la creación de nuevos dataframes originalmente vacíos.

4.3.3. Os

El tratamiento de datos se hace en un servidor el cual consta de los ficheros en almacenamiento local que posteriormente van a ser tratados y enviados al servicio web. Para ellos, es necesario un módulo encargado de utilizar las funciones asociadas con el sistema operativo, en este caso OS[18]. Se ha utilizado las funciones 'os.mkdir()' para crear carpeta con los archivos procesados y 'os.path.exists()' para comprobar que el path al que se intenta acceder existe.

4.3.4. Scipy

SciPy[19] es una biblioteca de Python utilizada principalmente para cálculo y simulación matemática, para aplicaciones científicas e ingenieriles. Proporciona muchas funciones útiles para el procesamiento y análisis de datos, integrando sub-bibliotecas como, entre otras, scipy.signal para procesamiento de señales, scipy.fft para transformadas rápidas de Fourier.

En el proyecto se utiliza en la parte donde se realiza la Transformada de Fourier de Corto Tiempo (STFT) para generar el espectrograma y para leer el fichero de audio.

4.3.5. Threading

La biblioteca threading[20] de Python se utiliza para manejar la concurrencia y asegurar que las operaciones críticas se ejecuten de manera segura en un entorno multi-hilo. En el proyecto se utiliza para garantizar que el código funcione correctamente y los datos se mantengan íntegros, incluso cuando múltiples hilos intenten ejecutar estas funciones simultáneamente.

4.3.6. Matplotlib

Matplotlib[4] es una completa librería para crear visualizaciones estáticas, animadas e interactivas en Python. Se utiliza en la creación del espectrograma y espectro.

4.3.7. Base 64

Base64[21] es un método de codificación que convierte datos binarios en una representación de texto utilizando un conjunto limitado de 64 caracteres ASCII. Base64 se utiliza para convertir la imagen del espectrograma generada en un formato adecuado para ser transmitido.

4.3.8. Io

El módulo IO[22] proporciona una abstracción uniforme para trabajar con flujos de datos. En el proyecto se utiliza .io.BytesIO para crear un flujo de datos en memoria donde se guarda una imagen del espectrograma generada con matplotlib. Esto es especialmente útil porque se necesita enviar imágenes.

4.3.9. JavaScript

JavaScript[23] es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos. Se utiliza principalmente en el lado del cliente, implementado como parte del navegador web, lo que permite una interfaz de usuario mejorada y páginas web dinámicas. Ha sido el lenguaje seleccionado para el desarrollo de la interfaz web pues ya se poseía habilidad con este lenguaje después de haberse usado en diferentes asignaturas del grado.

Ilustración 4.3: Logo de Javascript

4.4. React

El reto técnico que suponía exponer gran volumen de datos en un servicio web se resolvió gracias al desarrollo de la API ya mencionada con anterioridad junto con la elección de una biblioteca de Javascript que cumpliese con los requisitos de velocidad y adaptabilidad que requería el proyecto. Por tanto React que es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página fue la candidata número uno para el desarrollo del proyecto.

React[11] utiliza un DOM (Document Object Model)[24] virtual, que es una representación ligera del DOM real. Por tanto, cuando se realizan cambios en la interfaz de usuario,

React actualiza de manera eficiente solo las partes que han cambiado, lo que mejora el rendimiento de la aplicación, una parte esencial en este trabajo. Además, React permite dividir la interfaz de usuario en componentes reutilizables. Cada componente maneja su propio estado y lógica, lo que facilita el desarrollo y mantenimiento del código. Esto es especialmente útil en aplicaciones de muestreo de datos donde tienen múltiples componentes como gráficos, tablas, filtros y mapas de entrada.

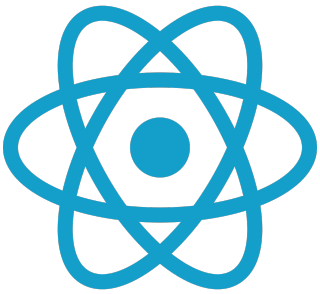

Ilustración 4.4: Logo de React

4.5. Librerías de React

Las librerías complementarias de React añaden funcionalidades avanzadas que no están incluidas en la base del framework. Estas bibliotecas proporcionan capacidades adicionales, como la integración de gráficos interactivos y mapas dinámicos, entre otras características.

4.5.1. PropTypes

PropTypes[25] es un paquete que contiene un mecanismo que asegura que el valor pasado es del tipo de dato correcto. Se usa para asegurar que los datos que se pasan a un componente son del tipo esperado, ayudando a prevenir errores y a hacer el código más robusto. Esta librería promueve la consistencia y estandarización en cómo se definen y utilizan las propiedades a lo largo de una aplicación React, además actúa como una forma de documentación, indicando claramente qué tipo de datos se espera que reciba un componente.

4.5.2. Chart.js

Chart.js[10] es la librería elegida para la implementación de gráficos. Las ventajas de esta herramienta es que es código abierto mantenido por la comunidad por lo que se puede modificar tanto como se desee, hay diferentes formas en las que visualizar los datos siendo cada una de ellas animable y completamente personalizable. Además, el gran detonante para elegir esta tecnología es el alto rendimiento de renderizado en todos los navegadores modernos y la adaptabilidad de los gráficos que se redibujan al cambiar el tamaño de la ventana para una granularidad de escala perfecta.

4.5.3. Chartjs-adapter-date-fns

Chartjs-adapter-date-fns[26] es un adaptador para la biblioteca de visualización de gráficos Chart.js que permite utilizar date-fns para el manejo y la manipulación de fechas en los gráficos. Esta herramienta se utiliza en uno de los gráficos para que pueda manejar correctamente las horas.

4.5.4. Leaflet

Leaflet[12] es la biblioteca JavaScript de código abierto líder en mapas interactivos adaptables a todos los dispositivos. Ofrece un amplio abanico de funciones cartográficas ideal para cualquier tipo de desarrollo en que se necesite un mapa. Gracias a Leaftlet se ha logrado el desarrollo del mapa presente en el proyecto. Debido a que puedes importar tileLayer haciendo usos de WMS (Web Map Service)[27].

El WMS es un estándar definido por el Open Geospatial Consortium (OGC)[28] que facilita la entrega de mapas georreferenciados a través de la web. Los mapas proporcionados por un WMS son imágenes generadas a partir de datos geoespaciales. En este caso se ha utilizado el servicio ofrecido por el proyecto EMODnet[29] (European Marine Observation and Data Network) que proporciona acceso a servicios web de mapeo relacionados con la batimetría. En la actualidad, es el único servicio de código abierto que ofrece servicio de batimetría. El uso correcto de esta herramienta supuso gran cantidad de tiempo de investigación y uso de herramientas externas como QGIS[30] para analizar qué capa del mapa era la que se adaptaba mejor al proyecto.

4.5.5. MUI

MUI[31] proporciona un conjunto completo de herramientas de interfaz de usuario gratuitas para ayudar a implementar funcionalidades. Se ha utilizado para facilitar el desarrollo de las siguientes características visuales: FormControlLabel, cuya función principal es proporcionar una etiqueta asociada a un elemento de entrada en un formulario; FormGroup es un componente para agrupar varios controles de formulario; Checkbox es un componente utilizado para crear casillas de verificación en formularios. Estas casillas permiten a los usuarios seleccionar el tipo de mapa que desea y Modal es un componente que muestra contenido sobre una superposición, utilizado para seleccionar el archivo a mostrar y el tipo de mapa.

4.5.6. React-select

React-select[32] es una popular biblioteca para React que proporciona un componente de selección con muchas características avanzadas. Es altamente personalizable y soporta funcionalidades como permitir que las opciones de selección varíen a medida que se escriba en el selector. Es una característica añadida para facilitar al usuario la búsqueda de un archivo específico.

4.5.7. React-super-responsive-table

React-Super-Responsive-Table[13] es una biblioteca de React que tiene la capacidad para adaptarse dinámicamente a diferentes tamaños de pantalla, se ha utilizado en las tablas de descriptores para garantizar una experiencia de usuario fluida y coherente en todas las plataformas.

4.6. Herramientas y Tecnologías No Programáticas

Para la elaboración de este TFT, se emplearon diversas tecnologías y herramientas que no implican la programación directa, pero que fueron esenciales para la gestión, documentación, y presentación del proyecto.

4.6.1. QGIS

QGIS[30] es un Sistema de Información Geográfica de software libre y de código abierto para diversas plataformas. Es una herramienta idónea para seleccionar la capa del WMS con el que se esté trabajando. Además, se puede comprobar cómo se va a visualizar. Esta herramienta se utilizó para determinar que capa era la necesaria adquirir de todas las ofertadas por el WMS gestionado por EMODnet.

4.6.2. Figma

El boceto inicial de la aplicación se realizó con papel y bolígrafo, pero el diseño final se realizó con la herramienta web Figma[33]. Figma es una herramienta de diseño de interfaces basada en la web. Figma es un editor de gráficos vectoriales y una herramienta de generación de prototipos que permite la colaboración en tiempo real y facilita la creación de diseños detallados y precisos. Su capacidad para crear prototipos interactivos ayudó a visualizar y validar la experiencia del usuario antes de iniciar el desarrollo.

4.6.3. Procreate

Procreate es una aplicación de edición de gráficos rasterizados especializada en pintura digital. En el presente proyecto, se utilizó Procreate para diseñar el logotipo de “Ocean sounds”, el cual se implementó posteriormente como favicon en la aplicación web.

4.6.4. Canva

Canva[34] es una plataforma en línea que permite a los usuarios crear diseños gráficos de manera sencilla y accesible. En este proyecto se ha utilizado para la realización de los casos

de uso.

4.6.5. GitHub

GitHub[35] es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se ha utilizado para el control de versiones tanto para la API como para la aplicación web.

4.6.6. Visual Studio Code

Visual Studio Code (VS Code)[36] es un editor de código fuente desarrollado por Microsoft que se ha convertido en una herramienta muy popular entre desarrolladores de software debido a su potencia, flexibilidad y extensibilidad. Ha sido el editor de código elegido para todo el proyecto.

4.6.7. LaTeX

LaTeX es un sistema de composición de textos diseñado para la creación de documentos con alta calidad tipográfica. En este proyecto, LaTeX ha sido la herramienta utilizada para la elaboración de la documentación del Trabajo de Fin de Grado.

Capítulo 5

Diseño

El diseño de "Ocean sounds" se fundamenta en la aplicación de principios establecidos en la teoría del diseño y la visualización de datos. Este capítulo aborda tanto el diseño gráfico como el diseño interno de la aplicación web y API, asegurando que cada aspecto de la interfaz sea intuitivo y eficaz para el usuario. Desde la elección de colores hasta la estructura interna del sistema, se han implementado diversas estrategias asegurando que cada aspecto de la interfaz sea intuitivo y eficaz para el usuario. Se han implementado diversas estrategias para optimizar la usabilidad, la claridad y la eficiencia de la aplicación

5.1. Diseño gráfico

En primer lugar, se han aplicado las reglas de oro de Ben Shneiderman[8] en el diseño. La primera "Strive for consistency", que aboga por la consistencia, se refleja en la estandarización en el diseño inicial y el trabajo por componentes. La tercera regla "offer informative feedback", que destaca la importancia de proporcionar comentarios informativos, se ejemplifica en la pantalla de selección de archivos, donde se indica claramente "Choose a file", guiando al usuario sobre qué acción realizar en dicho selector. Por último, la octava regla "reduce short-term memory load", que promueve reducir la carga de memoria a corto plazo, se evidencia en todo el diseño al ser sencillo pero completo, ofreciendo interacciones justas con la aplicación y mostrando todos los datos necesarios para que el usuario pueda ajustarlos según sus necesidades.

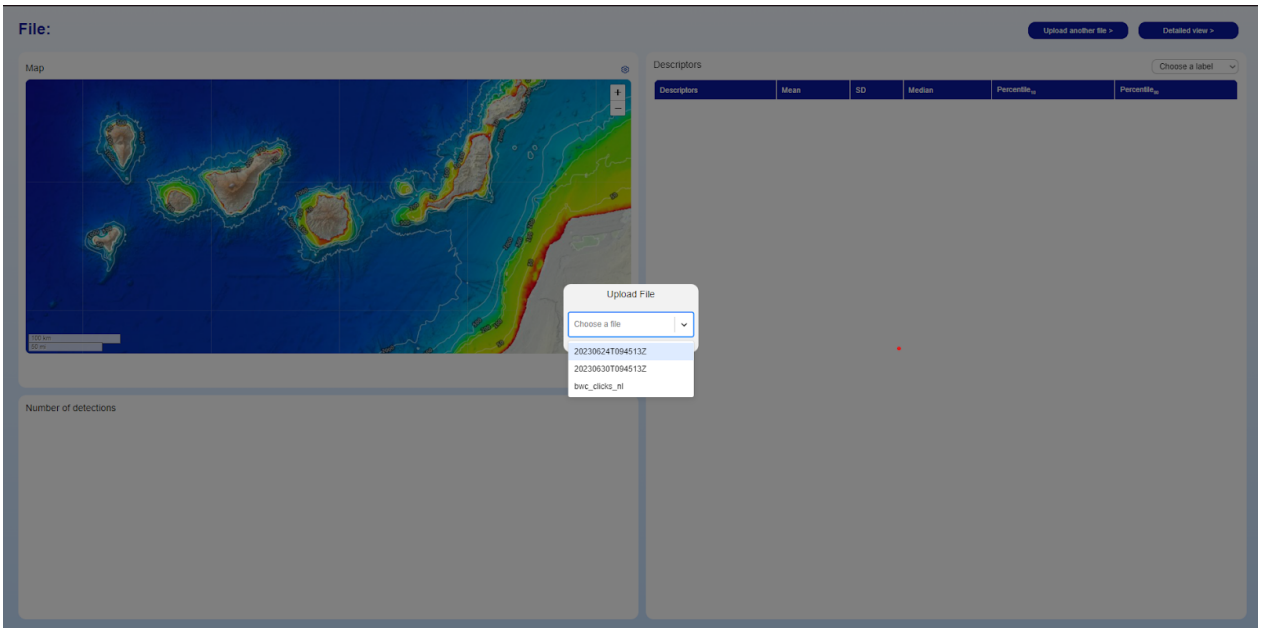

Ilustración 5.1: Modal de selección de fichero

En el diseño de interfaces y gráficos, especialmente cuando se manejan grandes volúmenes de datos, es crucial lograr una presentación visualmente agradable. Para lograr esto, se han adoptado estrategias específicas en cuanto a la elección de colores. Se ha optado por utilizar los mismos colores que se emplean en mapas batimétricos, ya que estos mapas son intrínsecamente ricos en color. La adición de más colores podría resultar en una sobrecarga visual, por lo que se ha decidido mantener la paleta de colores consistente con el mapa batimétrico para evitar esta situación.

El color de fondo se ha seleccionado como un azul suave, lo cual contribuye a una sensación de calma y un diseño profesional. Este tono azul también facilita la legibilidad de cualquier texto o elemento superpuesto sobre él. Por otro lado, el color de resaltado se ha establecido como el azul marino, un color que se utiliza comúnmente para indicar áreas importantes o destacadas, siguiendo así la convención de uso en mapas batimétricos.

Para asegurar que el diseño de los gráficos sea tanto simple como informativo, se han tomado inspiraciones de las ideas propuestas por Edward Tufte en su obra “The Visual Display of Quantitative Information” [9]. Esta guía enfatiza la importancia de mantener el diseño lo más sencillo posible sin sacrificar la capacidad de transmitir información clara y precisa. Además, se subraya la necesidad de que los colores utilizados sean representativos de la información que muestran, o, si no lo son, deben ser seleccionados con mucho cuidado y deliberación.

En la gráfica que muestra el número de detecciones, se han aplicado los colores del mapa batimétrico, pero con una selección cuidadosa para garantizar que cada color tenga un significado claro. Así, que si las etiquetas representan la calidad de la detección, el rojo se reserva para las detecciones de menor calidad (tipo C), el naranja para aquellas de calidad moderada (tipo B), y el verde para las de alta calidad (tipo A). Esto permite a los usuarios

identificar rápidamente la calidad de las detecciones simplemente observando el color. Si no se tratase de etiquetas que representan la calidad de la detección sino diferentes especies u otros datos, estos colores siguen siendo lo suficientemente diferentes unos de otros para que se facilite la diferenciación de las etiquetas en el gráfico.

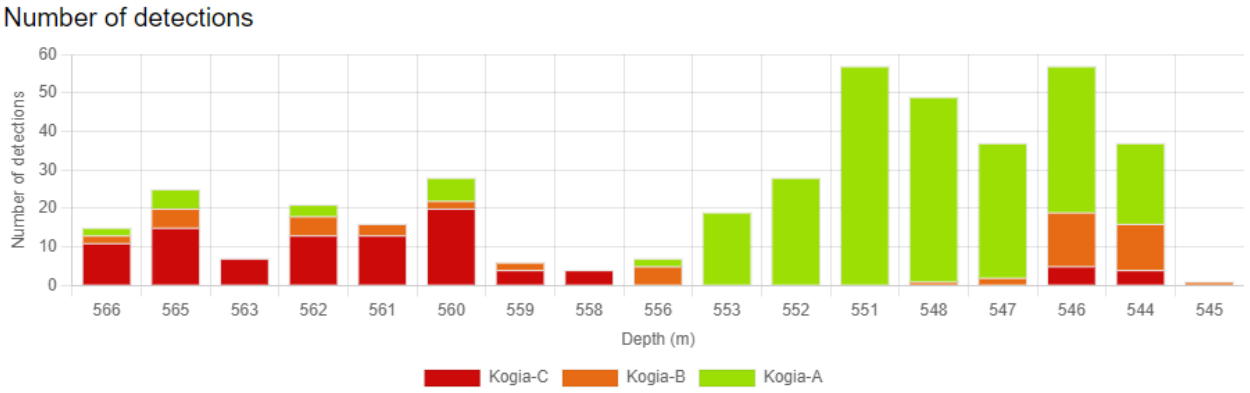

Ilustración 5.2: Gráfico relación número de detecciones profundidad

En lo que a la vista detallada respecta se han tomado las mismas decisiones con los colores, la consistencia y simplicidad. El gráfico del espectro se ha diseñado con un tono azul, coherente con el resto de la interfaz. Este enfoque cohesivo en la elección de colores ayuda a mantener la estética general del diseño mientras se proporciona información valiosa de manera efectiva.

5.2. Diseño interno del proyecto

El diseño interno de la aplicación web y la API de “Ocean sounds” se ha centrado en la adaptabilidad, la eficiencia y la responsividad. Para alcanzar estos objetivos, se han empleado diversas bibliotecas y tecnologías que optimizan tanto la presentación visual como el rendimiento de la aplicación.

5.2.1. Diseño aplicación web

La aplicación web tiene la estructura propia de un proyecto de React. Es decir, está organizada en páginas (pages), componentes (components) y recursos (assets). La aplicación cuenta con dos páginas principales: la vista detallada (detailedView) y la vista general (overview). Cada una de estas vistas tiene su propio archivo de estilos “.css” donde se define la apariencia específica de cada página. Sin embargo, existe un archivo global donde se encuentran datos compartidos como variables de estilo y configuraciones generales que son aplicables a toda la aplicación.

Dentro de los componentes de la aplicación web, se pueden identificar aquellos que son específicos de la vista general y aquellos que son propios de la vista detallada. En la vista general, se incluyen componentes como la tabla de descriptores (Descriptors), el gráfico que relaciona el número de detecciones con la profundidad (Detections), y el mapa que muestra las detecciones (DetectionsMap). Por otro lado, en la vista detallada, se encuentran componentes como la tabla de descriptores de una sola detección (DetailedDescriptor), el mapa de la detección (DetailedMap), el espectrograma (Spectrogram), y la señal de onda (AudioSignal).

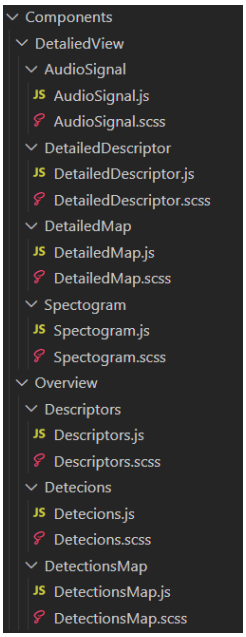

Ilustración 5.3: Estructura de los ficheros aplicación web

En cuanto a los recursos, se destaca la carpeta “icons”, la cual contiene todos los iconos utilizados a lo largo de la aplicación. Entre estos recursos se incluye un punto rojo, representado por el archivo `location_dot_red.svg`, que se utiliza en el mapa detallado de las detecciones. Además, se encuentra el icono “options.svg”, empleado para abrir las opciones del mapa.

La estructura de las páginas se organiza en varias secciones clave. En primer lugar, se realizan las importaciones necesarias. A continuación, se define el componente y, finalmente, se declara la exportación del componente. Dentro de la definición del componente, se encuentra inicialmente la declaración del mismo y su estado inicial. Seguidamente, se aborda el manejo de eventos, como por ejemplo, el cambio de fichero. La sección siguiente está dedicada a las funciones de tipo fetch, las cuales se utilizan para realizar solicitudes HTTP a la API y obtener datos. Por último, se procede al renderizado del componente. Dentro del renderizado, se incluye tanto el renderizado de la página en sí como el del modal.

La estructura de los componentes sigue un procedimiento similar al de las páginas, como se muestra en la ilustración 5.5. Primero se realizan las importaciones necesarias, luego se define el componente y finalmente se realiza su exportación. La definición del componente

se desglosa en varias secciones: inicialmente, la declaración del componente en sí; seguido por el manejo de eventos, donde se abordan las acciones específicas como la detección de cambios en los datos recibidos. Posteriormente, se incluye el uso de Hooks de React, que permiten gestionar efectos secundarios en componentes funcionales. Además, se encuentran las funciones auxiliares, en caso de que el componente las necesite. A continuación, se lleva a cabo el renderizado del componente y finalmente se realiza la validación de las propiedades utilizando PropTypes.

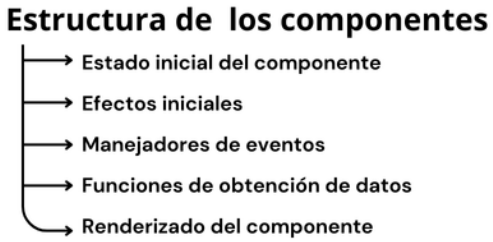

Ilustración 5.4: Estructura base de los componentes

5.2.2. Diseño de la interfaz de programación de la aplicación

La API, al igual que la aplicación web, se estructura en diferentes secciones como se muestra en la imagen.

Ilustración 5.5: Estructura de la API

La idea principal de este diseño se centra en la modularidad. Este enfoque permite una mayor claridad, mantenibilidad y escalabilidad del código.

La modularidad se la aporta la creación de múltiples funciones que posteriormente se han decidido dividir por funcionalidades para facilitar la compresión del código.

En primer lugar, se realizan las importaciones de las librerías necesarias. A continuación, se configura Matplotlib para funcionar sin un servidor específico y se crea la aplicación Flask. Posteriormente, se procede con la declaración de variables globales y el bloqueo de hilos.

La siguiente sección se centra en las funciones de utilidad, que incluyen: fileDate(), fileLabels(), fileStartAudio(), detectFiles() y getDetectionData(detectionName). Luego, se encuentran las funciones auxiliares para la generación de gráficos: `wigner_distribution(x, use_analytic=True, sample_frequency=None, t_0=0, t_1=1, flip_frequency_range=True)` y `rescaleAudio()`; y las encargadas de generar gráficos, como `generate_spectrogram(start_time, duration)` y `generate_audio_signal(start_time, duration)`.

Después, se definen funciones para manejar `DataFrames` con los datos necesarios, entre las cuales se destacan: `mapData(overview_df)`, `detectionsData(overview_df)` y `descriptorsData(overview_df)`. Seguidamente, se encuentra `processData(fileName)`, que procesa los datos del archivo CSV original y guarda los resultados en otros archivos CSV dentro de la carpeta de procesados.

Además, se tienen funciones para abrir los archivos procesados: 'openMap(fileName)', 'openDetecions(fileName)' y 'openDescriptors(fileName, label)'.

A continuación, se detallan las rutas GET de la API:

1. **/processedData/map:** Devuelve el archivo CSV de mapas procesado.
2. **/processedData/detections:** Devuelve el archivo CSV de detecciones procesado junto con las etiquetas.
3. **/processedData/descriptors:** Devuelve el archivo CSV de descriptores procesado basado en una etiqueta específica.
4. **/processedData/files:** Devuelve la lista de nombres de archivos disponibles.
5. **/processedDetection/`all_detections`:** Devuelve una lista de nombres de todas las detecciones en el archivo CSV.

Y las rutas POST:

1. **/processedData/audio**$_{s}ignal$: $Generaydevuelveungráficoconlaformadeondadelsegmentoselecci$
2. **/processedDetection:** Devuelve información detallada sobre una detección específica basada en el nombre de la detección.
3. **/processedData/:** Procesa un archivo CSV y devuelve información básica sobre él.

Por último, se inicia la aplicación Flask para que la API esté operativa.

5.2.3. Diseño estructura de almacenamiento de datos

El servidor que aloja la API utiliza un sistema de organización de carpetas estructurado en "rawData", "processedData" y "wavData". En la carpeta "rawData", se encuentran los archivos originales en formato CSV que la API transforma en diversos archivos procesados.

Dentro de la carpeta "processedData", existen subcarpetas denominadas "processedNombreDelFicheroOriginal", donde se encuentran los archivos ya procesados generados por la API. Estos archivos procesados son los que se envían como respuestas HTTP a la aplicación web.

Por otro lado, en la carpeta “wavData”, se almacenan los archivos “.wav” correspondientes a los archivos “.csv” ubicados en “rawData”. Ambos tipos de archivos comparten el mismo nombre, diferenciándose únicamente por la extensión del archivo.

5.2.4. Diseño estructura ficheros CSV

La API crea una carpeta llamada “processedNombreDelFicheroOriginal”. Dentro de esta carpeta se generan los siguientes archivos:

1. **`map_NombreDelFicheroOriginal.csv:`** Contiene tres columnas: “id”, “lat” y “lon”, y tiene tantas filas como detecciones existan en el archivo original. Este archivo representa los datos utilizados para la visualización de mapas.
2. **`detections_NombreDelFicheroOriginal.csv:`** Este archivo tiene las columnas “id”, “label” y “depth”, y se utiliza para la creación del gráfico que relaciona la profundidad con el número de detecciones.
3. **`descriptors_NombreDelFicheroOriginalEtiqueta.csv:`** Se crean tantos archivos como etiquetas diferentes existan en el fichero original. A menos que el nombre original contenga `_nl`, lo que indica que no se utilizan etiquetas, en cuyo caso solo se genera un archivo llamado `descriptors_NombreDelFicheroOriginalAll.csv`. Este último archivo tiene como columnas “Descriptor””Descriptor”, “Mean”, “SD”, “Median”, “Percentile10” y “Percentile90”, y contiene tantas filas como descriptores se hayan especificado en la variable `all_descriptors` dentro de la API.

5.3. Diseño del logo

El diseño del logo se ha concebido con la simplicidad como principio fundamental, en línea con la filosofía general del proyecto. Utiliza el color azul marino, que también se emplea como color destacado en la aplicación web para mantener coherencia visual. Además, representa la aleta caudal de un cetáceo, lo cual es pertinente dado que la aplicación está destinada al estudio de campañas PAM de cetáceos. El logo ha sido creado a mano utilizando la herramienta Procreate.

Ilustración 5.6: Logo de la aplicación

Capítulo 6

Desarrollo

El proyecto ha pasado por distintas fases. Se podrían nombrar como fase de análisis previo, diseño, implementación del sistema, realización de test con datos reales y desarrollo de la documentación.

6.1. Análisis previo

La primera parte del proyecto se centró en realizar un exhaustivo análisis. Se realizaron numerosas reuniones con los tutores puesto que están muy familiarizados con los datos con los que se iban a trabajar. En estas reuniones se decidió qué datos sería interesante mostrar, qué tipo de datos y con qué tecnologías se iba a trabajar.

En este punto los tutores facilitaron ficheros CSV con datos recogidos con el glider ya tratados. Se tomó la decisión de trabajar con este tipo de datos, además de ficheros WAV de audio. Estos ficheros aportan todos los datos necesarios.

Una vez elegidos los datos, se diseñó la manera de mostrarlos. En este momento, surgió la idea de mostrar un mapa, gráficos y una tabla con descriptores. Luego, se bocetó el diseño de la aplicación siguiendo las ideas surgidas de dichas reuniones.

En esta parte, se decidió trabajar en una aplicación web debido a que esto facilita el acceso de los potenciales usuarios con independencia de su localización geográfica. Además, se decide usar React puesto que aporta la flexibilidad de dispositivos y la rapidez de carga de datos. También se propone trabajar en una API de Python pues es la solución ideal para el tratamiento de datos de gran volumen y la conexión http con React. También, se decidieron las bibliotecas de React con las que se iban a trabajar los mapas y los gráficos.

6.2. Diseño

Una vez tomadas las decisiones de con qué tecnologías trabajar, qué datos utilizar y qué datos se desean mostrar se comienza con el diseño. En primer lugar, se realizan diferentes casos de usos. A continuación, se enumeran.

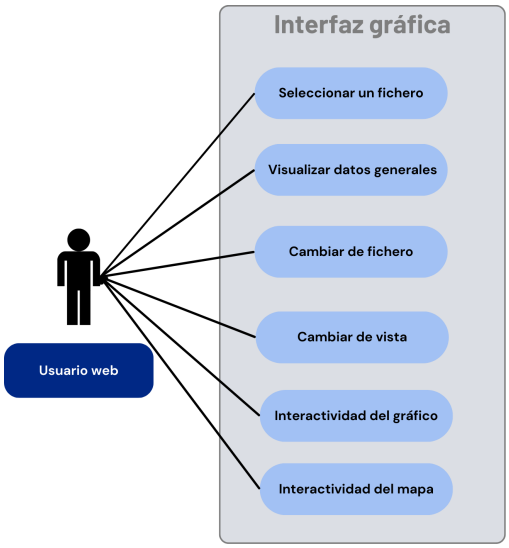

Ilustración 6.1: Casos de uso Aplicación Web

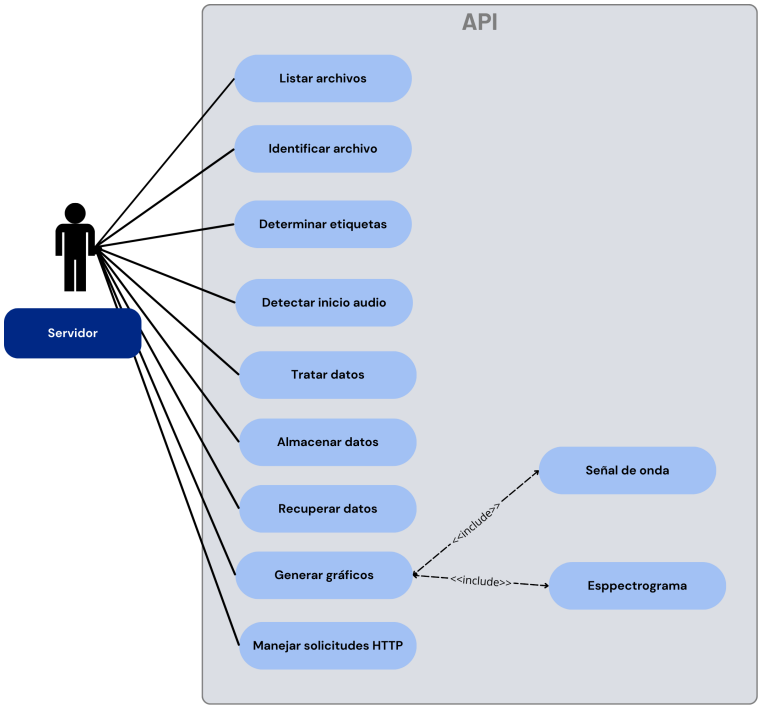

Ilustración 6.2: Casos de uso API

Una vez determinados los casos de uso, se realiza el diseño de las dos vistas de la aplicación web. Los diseños fueron realizados con el recurso web Figma. Se adjuntan los diseños originales.

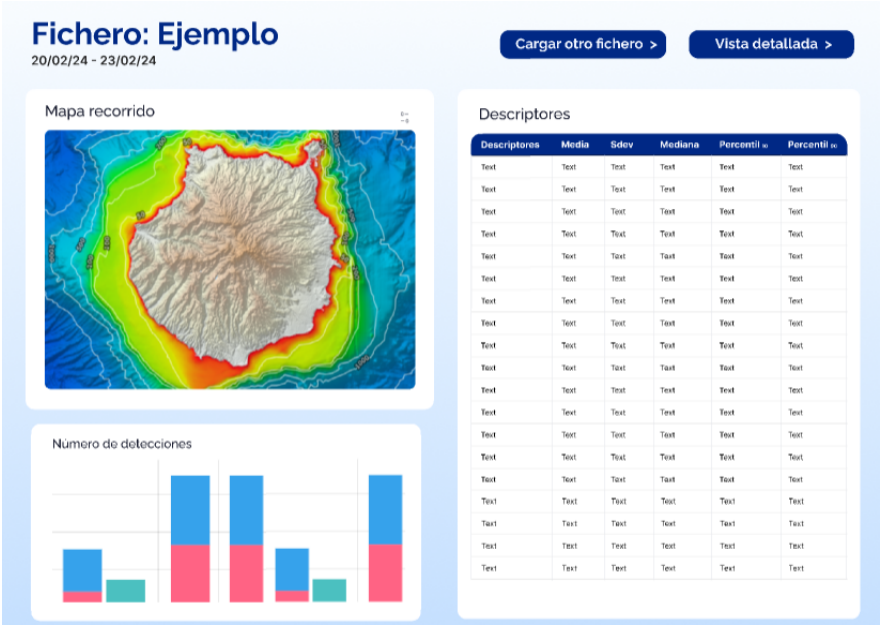

Ilustración 6.3: Diseño original vista general

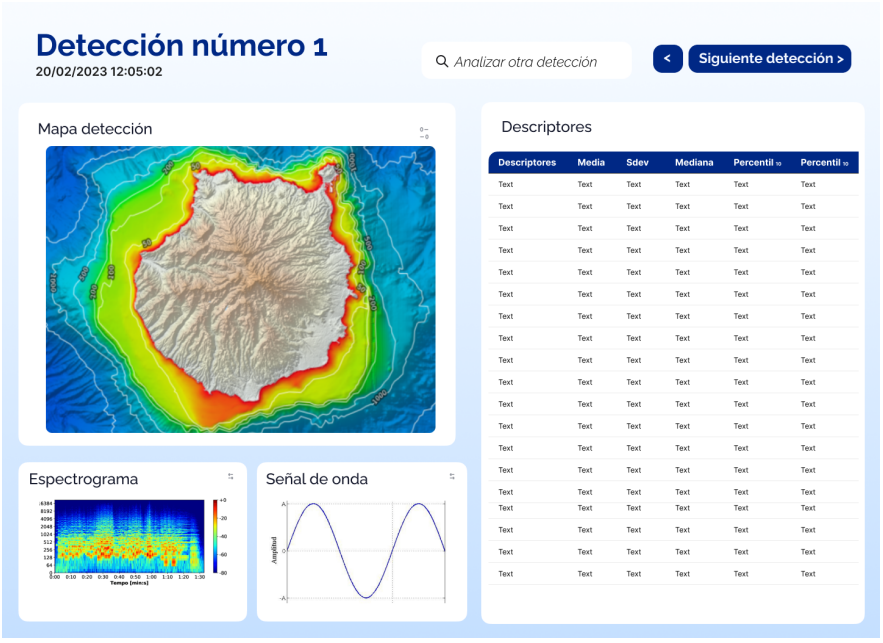

Ilustración 6.4: Diseño original vista detallada

6.3. Desarrollo del proyecto

En el proyecto se utilizó una metodología ágil, específicamente Scrum[37], para gestionar y ejecutar las tareas de manera eficiente y organizada. El trabajo se dividió en sprints, que son períodos de tiempo cortos y fijos, durante los cuales se completaron diferentes partes del proyecto. Al inicio de cada sprint, se llevó a cabo una planificación detallada donde se definieron los objetivos y las tareas a realizar. El desarrollo de la aplicación se dividió varios sprints focalizados en aspectos clave. Dentro del desarrollo del proyecto se pueden destacar cuatro grandes partes, el desarrollo de la aplicación web, el desarrollo de la API, la conexión entre la API y la aplicación web y la realización de pruebas. Cada sprint tuvo un coste temporal de dos semanas.

6.3.1. Sprint 1: Desarrollo de la interfaz gráfica sin datos

El primer sprint se centró en el desarrollo de la interfaz gráfica de la aplicación web. En primer lugar, se inicializó el proyecto de React en el editor de código Visual Studio Code, se creó el repositorio en Github y se conectó con el repositorio en local. A continuación, se diseñaron y desarrollaron las dos páginas principales de la web: la vista principal y la vista detallada.

Dado que la aplicación se estructura en componentes, se desarrollaron los componentes clave como la tabla, los botones y los modales para la selección de archivos. Desde el inicio, se prestó especial atención a la adaptabilidad de los componentes y las páginas para asegurar que fueran completamente adaptables a diferentes tamaños de pantalla de monitores. Para conseguirlo se importaron diversas bibliotecas que asegurarían la adaptabilidad y la cohesión de diseño.

La implementación de los modales se realizó con la biblioteca MUI (Material-UI), que proporciona componentes de interfaz de usuario modernos y personalizables. Para la creación y gestión de tablas, se utilizó la biblioteca React-super-responsive-table de React. Esta herramienta permitió desarrollar tablas altamente adaptables que se ajustan automáticamente a diferentes resoluciones y tamaños de pantalla. Los botones y otros elementos base se desarrollaron utilizando componentes nativos de HTML y estilos personalizados en CSS.

Estas tareas no dieron ningún error, simplemente una cantidad razonable de código a realizar en HTML y CSS.

6.3.2. Sprint 2: Desarrollo de la interfaz gráfica con datos estáticos

El segundo sprint comenzó con la importación de las librerías necesarias para la realización de los componentes restantes, es decir, el mapa y la gráfica. Al realizar las importaciones y pruebas con ejemplos de las documentaciones de “Leaflet” para los mapas y “Chart.js” para la gráfica no hubo ningún problema.

El siguiente paso fue realizar la gráfica de relación profundidad y número de detecciones separado por tipo de etiqueta con datos estáticos. La documentación de "Chart.js" es muy completa y detallada, así que no supuso un reto complejo aunque fuese la primera vez que se trabajara con dicha librería.

Primero se utiliza un "Hook"[38] para inicializar el gráfico cuando los datos están preparados. Dentro de este "Hook" está la función "fetchData", que actualmente contiene solo un conjunto de datos de prueba. El "Hook" se ejecuta únicamente si hay cambios en "docName", mientras que la función "fetchData" solo se activa si "docName" no está vacío. Esto garantiza que la aplicación web se conecte a la API en busca de datos únicamente cuando se elige un fichero. Además, solo se volverá a conectar para actualizar los datos si se selecciona un nuevo fichero.

Posteriormente, se implementa un efecto para preparar los datos necesarios para el gráfico basándose en los datos de "dataArray". Si tiene más de una etiqueta significa que hay diferentes tipos etiquetas y por tanto, se preparan los datos agrupados por etiquetas. Por otro lado, si solo existe una, se preparan los datos agrupados por profundidad.

Además, se incluyen funciones auxiliares como "prepareDataByDepth()", encargada de preparar los datos agrupados por profundidad, y "prepareDataByLabels()", función para preparar los datos agrupados por etiquetas. Se utiliza también "uniqueLabels(dataArray)" para obtener etiquetas únicas basadas en la profundidad de los datos, "countDetectionsPerDepth(dataArray)" para contar el número de detecciones por profundidad y etiqueta, y "countDetectionsPerDepthNoLabel(dataArray) "para contar el número de detecciones por profundidad sin etiqueta.

Finalmente, se realiza el renderizado del componente y se lleva a cabo la validación de las propiedades, siguiendo el estándar comúnmente utilizado en el desarrollo de componentes.

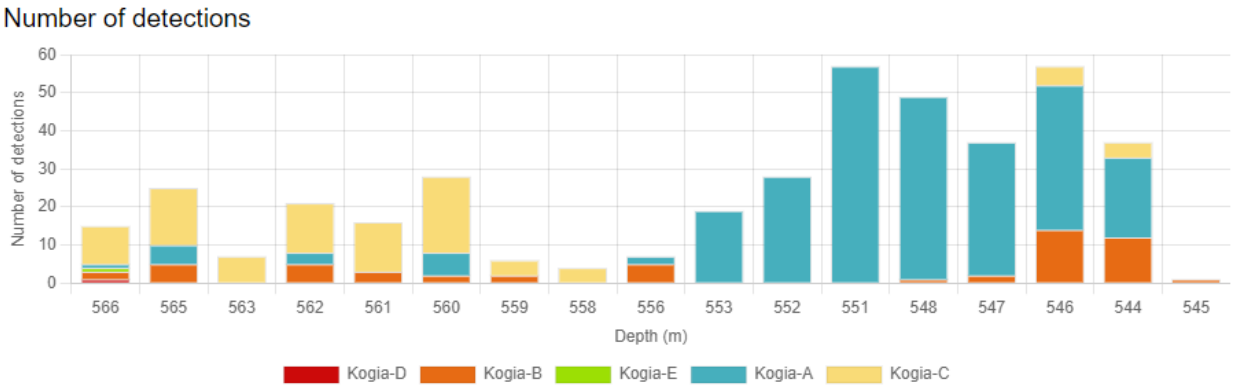

Ilustración 6.5: Gráfico relación número de detecciones profundidad

A continuación se comenzó con el desarrollo del mapa, fue todo un reto, lo complejo de esta parte fue lograr encontrar capas de datos batimétricos y usarlos correctamente. En la actualidad solo el servicio ofrecido por el proyecto EMODnet[29] (European Marine Observation and Data Network) proporciona acceso gratuito a servicios web de mapeo relacionados

con la batimetría. Para hacer uso de este servicio se ha usado su WMS. El uso correcto de esta herramienta supuso gran cantidad de tiempo de investigación y uso de herramientas externas como QGIS[39] para analizar qué capa del mapa era la que se adaptaba mejor al proyecto.

Una vez conseguido el mapa base, se implementó la posibilidad de cambiar de estilo de mapa pudiendo ponerlo en "estándar" o "batimétrico".

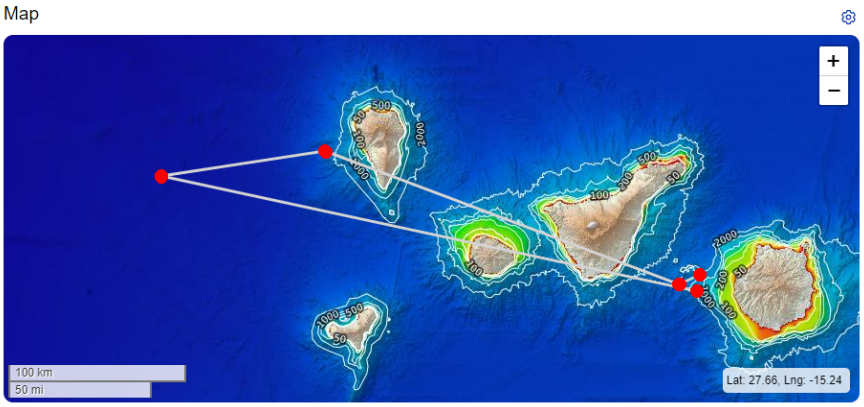

Ilustración 6.6: Mapa batimétrico

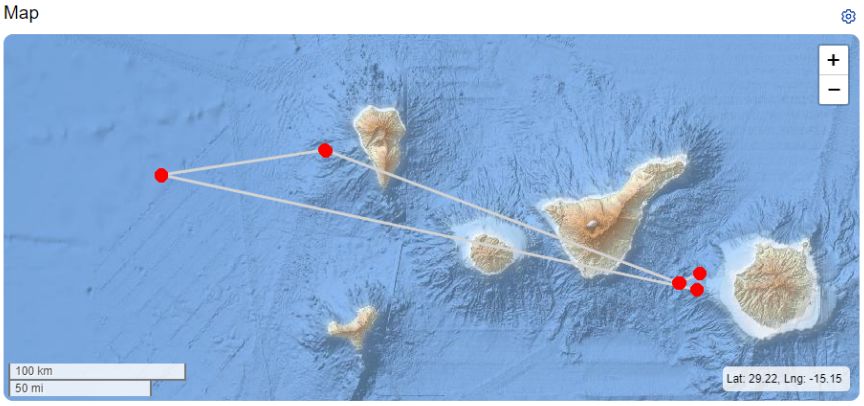

Ilustración 6.7: Mapa estándar

Posteriormente, se añadieron datos estáticos de detecciones. Tras esta implementación surgieron varios problemas. Originalmente, las detecciones se unían usando la función "polyline" propia de "Leaflet", pero realmente estos datos eran incorrectos porque la unión de las detecciones no representaba necesariamente el recorrido exacto que realiza el glider. Por tanto, se tomó la decisión de indicar las detecciones con puntos rojos usando la función "circleMarker" y unir en gris el posible camino tomado por el glider. Esto también supone una mejora considerable cuando los datos no son de una sola campaña, sino de varias, y por tanto, los datos que se visualizan reflejan mejor la naturaleza y origen de los datos, evitando posibles interpretaciones erróneas.

Además, se añadieron los controles de zoom, escala[40] y la funcionalidad de que el posicionamiento del cursor sobre el mapa indicase la latitud y longitud[41] en la que se encuentra. Esto fue crucial porque el mapa ajusta el zoom automáticamente con respecto a las detecciones encontradas, y en algunos casos, cuando se trataba de un fragmento de campaña, se presentaba una sucesión de puntos rojos unidos por una línea gris sobre un fondo completamente azul. Esta situación impedía que el usuario tuviera algún tipo de retroalimentación sobre la ubicación de esas detecciones.

Esta serie de mejoras no solo optimizó la precisión y la utilidad de los datos visualizados, sino que también incrementó significativamente la usabilidad de la aplicación, proporcionando a los usuarios una herramienta más efectiva y fiable para el análisis de las detecciones del glider.

6.3.3. Sprint 3: Desarrollo de la API sin llamadas HTTP

Una vez finalizada la aplicación web estática, se procedió al desarrollo de la API con el objetivo de facilitar la conexión entre ambas partes posteriormente.

En primer lugar, se realizaron las importaciones adecuadas y se comenzaron a desarrollar algunas funciones auxiliares:

1. **fileDate():** Determina la fecha asociada al nombre del archivo, interpreta el formato y devuelve una cadena de fecha formateada.
2. **fileLabels():** Extrae y devuelve las etiquetas únicas de detección contenidas en el archivo CSV, representando diferentes categorías o tipos de eventos acústicos registrados.
3. **fileStartAudio():** Determina el tiempo de inicio del audio basado en el archivo CSV de datos, crucial para calcular la ubicación temporal de las detecciones.
4. **detectFiles():** Lista todos los archivos disponibles en el directorio "./rawData/", que contiene los datos brutos de detección.

El siguiente paso fue la creación de la función "getDetectionData(detectionName)", encargada de filtrar y devolver los datos de una única detección específica en forma de DataFrame. Además, se desarrollaron las funciones "mapData", "detectionsData" y "descriptorsData", las cuales, a partir del DataFrame original, obtienen los datos necesarios.

Posteriormente, se implementó la función "processData(fileName)", la cual crea la carpeta "processedNombreDelArchivo", lee el archivo CSV de datos brutos asociado con "fileName", realiza diversas transformaciones y análisis estadísticos, y llama a las funciones "mapData", "detectionsData" y "descriptorsData", guardando los resultados en directorios específicos dentro de "./processedData/".

Finalmente, se crearon las funciones de apertura de datos procesados ("openMap()", "openDetecions()", "openDescriptors()"), las cuales permiten acceder y leer los datos procesados almacenados en archivos CSV dentro de la carpeta "./processedData/".

6.4. Sprint 4: Desarrollo de funciones de creación de gráficos en python

El cuarto sprint estuvo exclusivamente destinado al desarrollo de las funciones de creación de gráficos de Python. Estas son: `generate_spectrogram()`, `generate_audio_signal()` y la funciones axiliares `wigner_distribution()` y `rescaleAudio()`.

Estas funciones fueron probadas inicialmente de manera independiente en un entorno de desarrollo.

La función `generate_spectrogram()` se encarga de generar un gráfico que representa la distribución Wigner-Ville a partir de un archivo WAV ubicado en la carpeta "wavData". La distribución Wigner-Ville es una técnica utilizada en el análisis de señales para obtener una representación conjunta en el dominio tiempo-frecuencia. Esta función comienza intentando leer el archivo WAV especificado. Si el archivo no se encuentra o si surge algún otro error durante la lectura, la función devuelve un mensaje de error adecuado. Luego, se selecciona el intervalo de tiempo que abarca los datos de la detección, si el tiempo se encuentra fuera de límite la función devuelve un mensaje de error informando de esto. A continuación, se ejecuta la función `rescaleAudio()` encargada de normalizar los datos de audio para que estén dentro del rango de -1 a 1, donde 1 es la amplitud máxima y -1 es la amplitud mínima.

```
def rescaleAudio(data):
    max_val = np.max(data)
    min_val = np.min(data)

    epsilon = 1e-10
    normalized_data = 2 * (data - min_val) / (max_val - min_val + epsilon) - 1

    return normalized_data
```

Algoritmo 6.1: Normalización de los datos de audio

A continuación, se ejecuta la función `wigner_distribution()` que se muestra en el algoritmo 6.2 que calcula la distribución Wigner-Ville discreta de una señal de entrada. Este método permite analizar cómo la energía de la señal se distribuye en el tiempo y la frecuencia, proporcionando una representación tiempo-frecuencia.

Los argumentos de la función son:

1. **x:** Es un array unidimensional que representa la señal de entrada de longitud N.
2. **use_analytic:** Es un booleano que indica si se debe usar la señal analítica asociada a x. Por defecto está configurado en True.
3. **sample_frequency:** Es la frecuencia de muestreo de la señal.
4. **t_0:** Es el tiempo en que se registró la primera muestra.
5. **t_1:** Es el tiempo en que se registró la última muestra.

6. **flip_frequency_range:** Es un booleano que indica si se debe voltear el rango de frecuencias para que la frecuencia mínima aparezca en la esquina inferior izquierda de la matriz de salida.

Tras la ejecución de la función se retona:

1. **wigner_distribution:** Es una matriz N x N que contiene la distribución Wigner-Ville de la señal.
2. **frequency_bins:** Es un array que representa el rango de frecuencias.

La función comienza verificando si la señal de entrada "x" es un "array" de NumPy. Si no lo es, la convierte usando "np.asarray(x)".

A continuación, se asegura de que x sea un "array" unidimensional. Si x tiene más de una dimensión, lanza un error para indicar que la entrada debe ser una serie temporal unidimensional.

Si el argumento "use_analytic" es "True", la función verifica si la señal es real. Si es real, la convierte a su forma analítica utilizando la transformada de Hilbert[42] "(signal.hilbert(x))". Si x es compleja, se lanza un error, ya que la señal analítica solo puede calcularse para señales reales.

La longitud de la señal N se obtiene usando "x.shape[0]". Se inicializa un vector de tiempo t y una matriz "wvd" de ceros con dimensiones N x N para almacenar la distribución Wigner-Ville. Se utiliza un bucle "for" para iterar sobre cada punto en el tiempo n. Dentro del bucle, se calculan los límites "tau_min" y "tau_max" para los desplazamientos de tiempo, y se crea un array "tau" que contiene estos desplazamientos.

Luego, para cada "n", se actualiza la matriz "wvd" con productos cruzados de la señal desplazada y su conjugado complejo. Se aplica la Transformada de Fourier a la matriz wvd en la dirección de las frecuencias para obtener la representación en el dominio tiempo-frecuencia. Esto se realiza en tres pasos: primero, se usa "np.fft.fftshift" para centrar el espectro de Fourier; luego, se aplica la transformada de Fourier utilizando "np.fft.fft"; y finalmente, se vuelve a centrar el espectro usando "np.fft.fftshift".

Si no se proporciona "sample_frequency", se calcula como "N / (t_1 - t_0)", que es la inversa del intervalo de tiempo total.

La frecuencia máxima depende de si se usó la señal analítica. Si "use_analytic" es "True", la frecuencia máxima es la mitad de la frecuencia de muestreo. De lo contrario, es un cuarto de la frecuencia de muestreo. Si "flip_frequency_range" es "True", la matriz "wvd" se invierte en la dirección de las frecuencias para que la frecuencia mínima esté en la esquina inferior izquierda.

Finalmente, la función devuelve la parte real de la matriz "wvd" y la frecuencia máxima calculada.

```
def wigner_distribution(x, use_analytic=True, sample_frequency=None,
                        t_0=0, t_1=1, flip_frequency_range=True):
    """Discrete Pseudo Wigner Ville Distribution based on [1]
```

```
    Args:
        x, array like, signal input array of length N
        use_analytic, bool, whether or not to use analytic associate of input
            data x by default set to True
        sample_frequency, sampling frequency
        t_0, time at which the first sample was recorded
        t_1, time at which the last sample was recorded
        flip_frequency_range, flip the data in about the time axis such that
            the minimum frequency is in the left bottom corner.

    Returns:
        wigner_distribution, N x N matrix
        frequency_bins, array like, length N frequency range

    References:
        [1] T. Claasen & W. Mecklenbraeuker, The Wigner Distribution -- A Tool
        For Time-Frequency Signal Analysis, Phillips J. Res. 35, 276-300, 1980
    """

    # Ensure the input array is a np array
    if not isinstance(x, np.ndarray):
        x = np.asarray(x)
    # Compute the autocorrelation function matrix
    if x.ndim != 1:
        raise ValueError("Input␣data␣should␣be␣one␣dimensional␣time␣series.")
    # Use analytic associate if set to True
    if use_analytic:
        if all(np.isreal(x)):
            x = signal.hilbert(x)
        else:
            raise RuntimeError("Keyword␣'use_analytic'␣set␣to␣True␣but␣signal"
                               "␣is␣of␣complex␣data␣type.␣The␣analytic␣signal"
                               "␣can␣only␣be␣computed␣if␣the␣input␣signal␣is"
                               "␣real␣valued.")

    # Calculate the wigner distribution
    N = x.shape[0]
    bins = np.arange(N)
    indices = linalg.hankel(bins, bins + N - (N % 2))

    padded_x = np.pad(x, (N, N), 'constant')
    wigner_integrand = \
        padded_x[indices+N] * np.conjugate(padded_x[indices[::, ::-1]])

    wigner_distribution = np.real(np.fft.fft(wigner_integrand, axis=1)).T

    # Calculate sample frequency
    if sample_frequency is None:
        sample_frequency = N / (t_1 - t_0)

    # Calculate frequency range
    if use_analytic:
        max_frequency = sample_frequency/2
    else:
        max_frequency = sample_frequency/4

    # Flip the frequency range
    if flip_frequency_range:
        wigner_distribution = wigner_distribution[::-1, ::]
```

```
return wigner_distribution, max_frequency
```

Algoritmo 6.2: Función wigner_distribution

Una vez ejecutada la función `wigner_distribution()`, se utilizan funciones de la biblioteca MatplotLib para la creación del gráfico, se guarda la imagen en memoria en formato PNG y finalmente, se convierte la imagen en una cadena Base64 para facilitar su uso en aplicaciones web.

Por otro lado, está la función `generate_audio_signal()` que genera la señal de onda de un segmento de audio. Los pasos que se realizaron en está función fueron los mismos que para `generate_spectrogram()`, la diferencia erradica en la ejecución de `np.linspace()` que se muestra en el algoritmo 6.3 en vez de `wigner_distribution()`. La función `np.linspace()` crea un "array" de NumPy que representa el eje del tiempo para un segmento de datos de una señal.

```
time_axis = np.linspace(start_time, start_time + duration, len(data_segment))
```

Algoritmo 6.3: Función

Una vez introducido el código en la API se tuvo que resolver el problema de la concurrencia, como la web hace las llamadas a la vez a la API de la creación del espectrograma y de la señal de onda desembocaba en errores. Por lo que se implementó un cerrojo para impedir que las dos gráficas se crearán a la vez dando errores. Además, la imagen se convirtió a base64 para posteriormente ser enviada por HTTP.

6.4.1. Sprint 5: Conexión de la API con la aplicación web

En la aplicación web, todas las páginas y componentes utilizan la función "fetchData". Esta función está integrada en un "Hook" que se activa cuando hay cambios en los datos relevantes. Su objetivo es realizar una solicitud HTTP a la API para obtener datos. Después de verificar la respuesta y convertirla al formato JSON necesario, los datos formateados se almacenan en el estado de la aplicación web (dataArray) o en variables individuales como "DocDate", "DocName" y "DocLabels".

En la API, se implementaron varios métodos GET que hacen uso de funciones específicas. Inicialmente, tenemos `map_csv()`, `detections_csv()`, y `descriptors_csv()`. Estas funciones abren archivos CSV respectivos y devuelven su contenido en formato JSON. Luego, la función `files()` utiliza `detectFiles()` para obtener una lista de nombres de archivos disponibles, que luego se retorna en formato JSON. Además, `detections()` lee el archivo CSV de datos brutos (`./rawData/ + fileName`), extrae los nombres de las detecciones y los devuelve en formato JSON.

Además, se desarrollaron métodos POST con funcionalidades específicas. Uno de ellos es `process_detections()`, que utiliza `getDetectionData(detectionName)` para obtener detalles de una detección específica. Este método retorna el nombre del archivo, fecha, hora y datos de la detección en formato JSON.

Otro método es `process_data()`, el cual verifica la validez del nombre de archivo proporcionado y calcula la fecha asociada, las etiquetas de detección y el tiempo de inicio del audio. Si el directorio de procesamiento correspondiente no existe, se llama a `processData(fileName)` para realizar el procesamiento y almacenamiento de los datos. Posteriormente, retorna el nombre del archivo procesado, la fecha y las etiquetas de detección en formato JSON.

6.4.2. Sprint 6: Realización de pruebas

En este sprint, se llevó a cabo la fase de pruebas del proyecto para garantizar la funcionalidad correcta y la estabilidad de la aplicación desarrollada. Las pruebas desempeñan un papel crucial en la validación de cada componente y funcionalidad implementada, asegurando que cumplan con los requisitos especificados y que se comporten de manera esperada bajo diversas condiciones.

Las pruebas manuales jugaron un papel fundamental en la validación de cada componente y funcionalidad implementada. Se verificó que todos los requisitos especificados se cumplieran adecuadamente y que la aplicación se comportará de manera esperada bajo diversas condiciones operativas.

Una vez completadas las pruebas manuales, se procedió a analizar el código utilizando SonarQube[43], una plataforma de código abierto diseñada para evaluar la calidad del código mediante análisis estático. Proporciona informes detallados sobre diversas métricas de calidad del software, incluyendo la detección de código duplicado, vulnerabilidades de seguridad, cumplimiento de estándares de codificación y prácticas recomendadas de programación. Este análisis ayuda a identificar áreas de mejora en el código base, asegurando así que el software desarrollado cumpla con los estándares de calidad y sea más mantenible a largo plazo.

En todos los apartados de SonarQube se obtuvo una calificación de “A”, siendo este el valor más alto posible. Los únicos fallos registrados fueron leves y se deben a errores de lectura del propio SonarQube, es decir, son fallos que pueden ignorarse. Estos errores incluyen casos como los mostrados en la captura adjunta, donde se puede observar que SonarQube no detecta correctamente la “e” en “ClassName”.

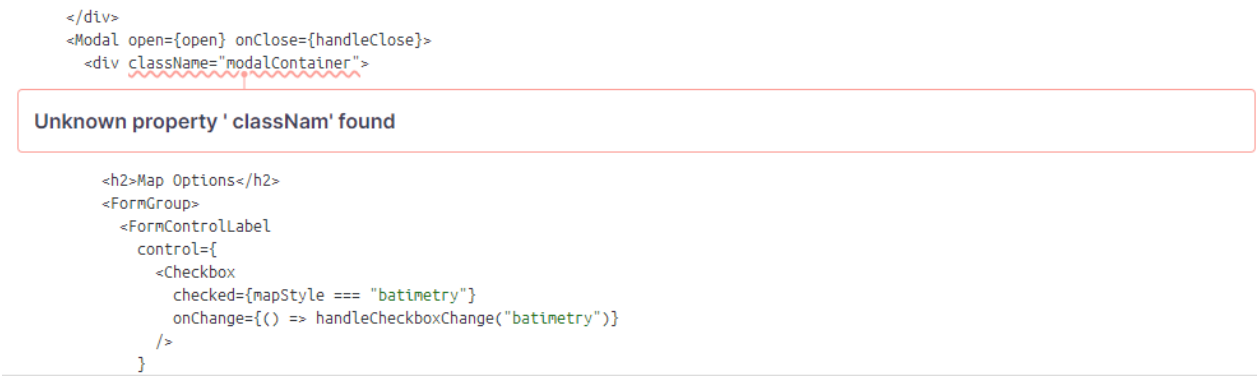

Ilustración 6.8: Fallos SonarQube

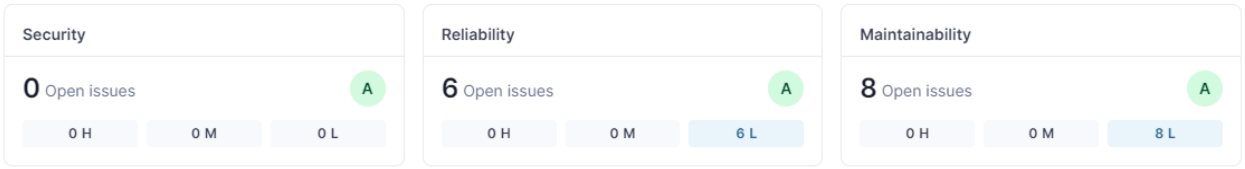

Ilustración 6.9: Análisis SonarQube

También se utilizó PageSpeed[44] utilizando un servidor de prueba en Vercel[45]. El rendimiento mostró excelentes resultados en todas las métricas evaluadas, destacando los siguientes puntos: El tiempo de la "First Contentful Paint" (FCP) esta métrica mide el tiempo que tarda el navegador en renderizar el primer elemento de contenido, fue de 0,2 segundos, mientras que el mayor elemento visible (LCP) se cargó en 0,9 segundos. El tiempo total durante el cual la página fue bloqueada por scripts largos, que impidieron la interacción del usuario (TBT) fue mínimo, con solo 40 milisegundos, y el índice de velocidad (SI), qué tan rápido se carga visualmente el contenido de la página, fue de 0,6 segundos. Estas métricas indican una carga extremadamente rápida de la página, crucial para mejorar la experiencia del usuario y la retención.

En cuanto a la accesibilidad, se identificó un único error relacionado con la verificación de contraste de color: "Los colores de fondo y de primer plano deben tener una relación de contraste adecuada". Este error específico se debe al fondo del modal, que se considera contenido crítico que los usuarios deben poder ver claramente, aunque no sea el objetivo principal de la aplicación.

La puntuación en las prácticas recomendadas alcanzó un 96 %, con la única limitación siendo de la falta de conexión con la API en el servidor de prueba de Vercel.

Por último, en SEO la aplicación obtuvo una puntuación perfecta del 100 %, lo que indica que cumple con las directrices básicas de optimización para motores de búsqueda. Esto contribuye significativamente a mejorar la visibilidad en los resultados de búsqueda.

Ilustración 6.10: Análisis PageSpeed

6.5. Desarrollo de la memoria

Durante el desarrollo del proyecto, se tomaron notas preliminares que sirvieron como base para la elaboración de la memoria final. La redacción de la memoria se completó en un período de tres semanas, durante el cual el tutor revisó y corrigió el documento para asegurar

su calidad y precisión. La memoria se realizó utilizando LaTeX. El proyecto se estructuró de la siguiente manera: el primer mes se dedicó al análisis y diseño, las siguientes 12 semanas a la implementación y, finalmente, las últimas tres semanas al desarrollo de la memoria.

Capítulo 7

Estado final

El proyecto actual se divide en dos componentes principales: una API para la administración de datos y una interfaz gráfica para su visualización.

La API, construida con Python, incluye varias funciones clave. Inicialmente, hay cinco funciones dedicadas a tareas específicas: una para identificar el nombre del archivo, otra para determinar sus etiquetas, una tercera para listar archivos locales disponibles, una cuarta para detectar el inicio del audio y la última para almacenar detalles del archivo en un "DataFrame".

A continuación, nos encontramos con funciones encargadas de realizar el espectrograma y el en la representación de la onda de la detección en Python. Dentro de estas funciones se han utilizados cerrojos puesto que las funciones se llaman a la vez cuando se accede a la vista detallada y si se ejecutase a la vez los datos dichas gráficas saldrían corruptos causando errores en las gráficas. Además, hay funciones auxiliares encargada de la realización de la distribución de Wigner-Ville y el reescalado del audio.

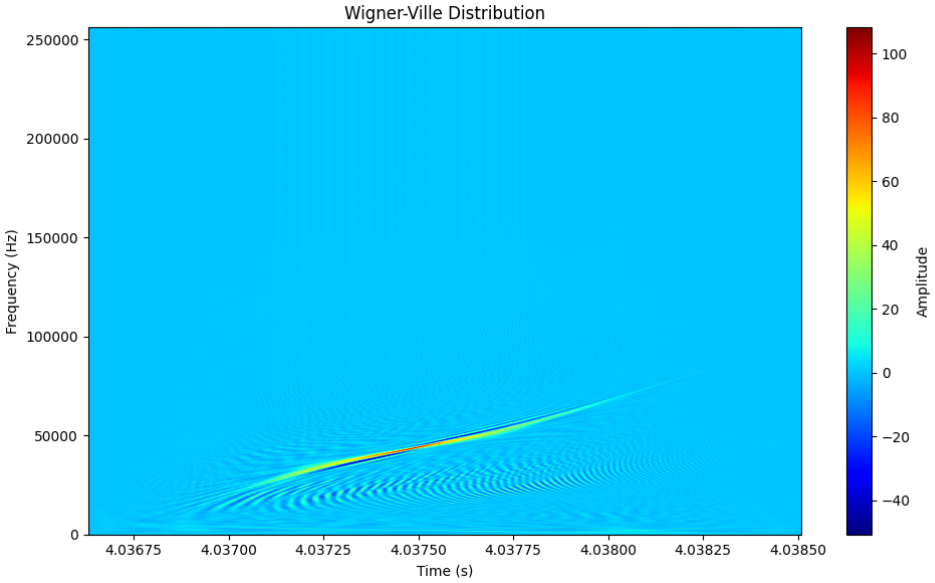

Ilustración 7.1: Espectrograma

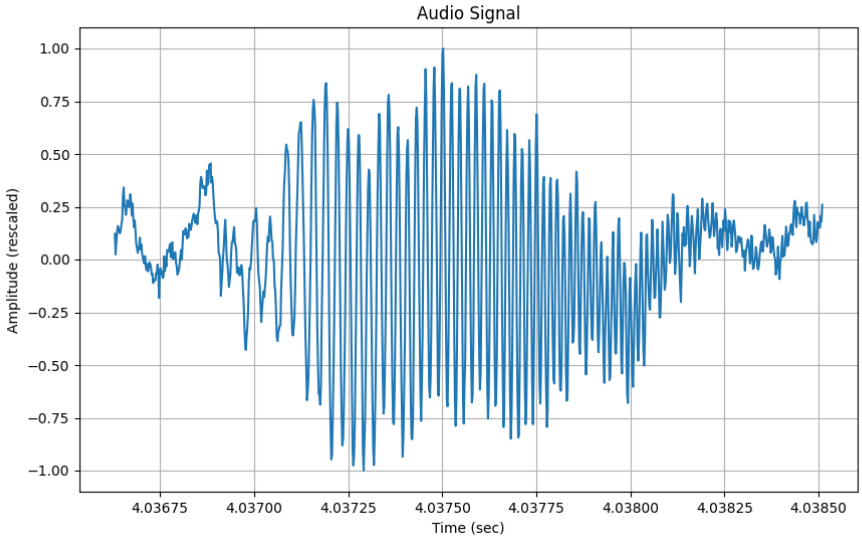

Ilustración 7.2: Señal de onda

Posteriormente, se encuentran tres funciones diseñadas para recuperar información de cada componente de la interfaz gráfica: 'mapData', 'detectionsData' y 'descriptorsData'. Además, existe la función 'processData', que se activa únicamente si el archivo a mostrar en la interfaz gráfica es nuevo. Esta función se encarga de extraer, procesar y almacenar los datos del archivo CSV en una carpeta denominada 'processedNombreDelFichero'. Sigue a esto una serie de tres funciones que manejan la lectura de archivos previamente procesados: 'openMap', 'openDetections'y 'openDescriptors'.

En cuanto a las solicitudes HTTP, se han definido cinco rutas específicas que responden a dichas solicitudes. Cuando se accede a estas URLs, Flask ejecuta automáticamente las correspondientes funciones (`map_csv`, `detections_csv`, `descriptors_csv`, `files` y `detections`), las cuales recolectan información del servidor y la envían al cliente solicitante en formato JSON.

También se han definido dos endpoints que permiten a los usuarios enviar solicitudes POST para generar y recibir imágenes de espectrogramas y espectros de audio.

Finalmente, se cuentan con dos rutas adicionales destinadas a solicitudes HTTP POST, las cuales esperan recibir datos en el cuerpo de la solicitud. Una de estas funciones se enfoca en recoger información sobre una detección específica, mientras que la otra se ocupa de un archivo.

En cuanto a la interfaz gráfica consta de dos páginas, la vista detallada y la vista general. Al abrir la aplicación por primera vez aparece una ventana para elegir fichero, esta ventana es un selector de todos los ficheros que se encuentran en local en el servidor y se puede escribir para encontrar uno específico. La vista general consta de título, fecha y botones que nos permiten cambiar de fichero o cambiar de vista.

Además, el mapa nos permiten ubicar las detecciones con puntos rojos y une con líneas grises las detecciones para tener una aproximación del recorrido del glider.

El mapa posee un botón que nos permite cambiar el diseño del mapa de batimétrico a estándar, asimismo incluye en la parte inferior izquierda una escala y en la parte inferior derecha una indicación de la longitud y latitud a la que se encuentra nuestro cursor sobre el mapa, estas herramientas nos ayudan a poner en contexto al usuario.

La tabla de descriptores muestra los valores estadísticos de los datos seleccionados en la API en la parte superior derecha se encuentra un selector que nos permite variar las etiquetas de los datos que se están mostrando.

Descriptors

Kogia-A

Descriptors	Mean	SD	Median	Percentile$_{10}$	Percentile$_{90}$
depth[m]	549.680	4.190	548.360	545.970	552.610
clip[usec]	1611.670	543.380	1544.000	956.200	2296.600
msec	0.520	0.300	0.520	0.100	0.900
low_i	106.210	61.100	95.000	33.000	187.000
high_i	214.590	76.080	204.000	126.200	323.800
energy[%]	98.800	0.970	99.040	97.450	99.640
SNR[dB]	23.850	3.840	23.440	19.100	29.520
Fp[kHz]	125.060	6.050	123.600	120.400	136.000
Fc[kHz]	124.330	3.030	124.610	122.300	127.570
RMS[kHz]	12.390	5.810	12.010	4.970	22.030
Qrms	12.800	6.880	10.370	5.830	24.080
Fs[kHz]	512.000	0.000	512.000	512.000	512.000
slope_max_wvd	0.010	0.050	0.000	-0.030	0.100

Ilustración 7.3: Tabla de descriptores

El gráfico muestra la relación entre la profundidad y el número de detenciones separado por etiquetas si las hubiesen, este gráfico es interactivo. Es posible deslizar el cursor sobre él para ver los valores, además de hacer clic en una etiqueta en la leyenda para que desaparezcan los elementos etiquetados con ella.

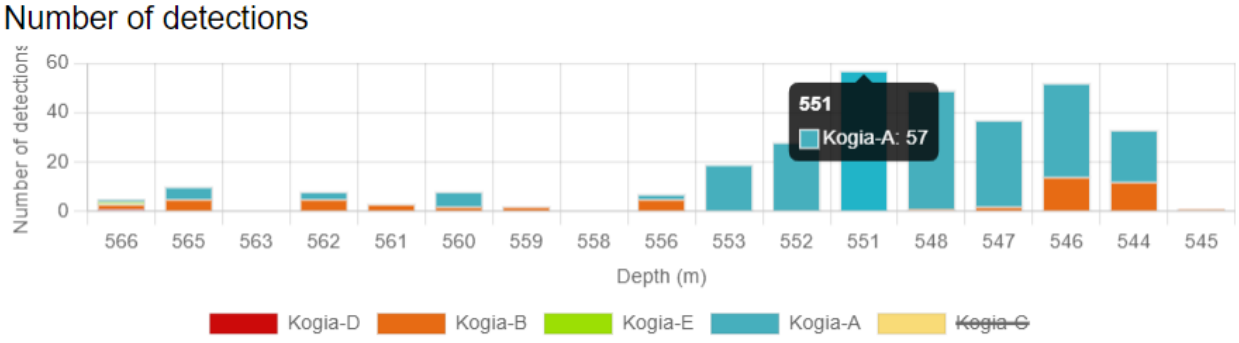

Ilustración 7.4: Ejemplo de interactividad del gráfico

La vista detallada muestra título, fecha, hora, botones para cambiar de detección, un mapa, la tabla de descriptores, el espectrograma, y el espectro.

El mapa tiene las mismas características que el de la vista detallada pero en este solo se muestra el punto que corresponde con la detección específica, además se abre una pequeña ventana emergente que muestra el estado y la profundidad del glider en ese instante. Tanto

el espectro como el espectrograma son imágenes estáticas que se calculan en tiempo real en el servidor y se ajustan al tamaño de la pantalla.

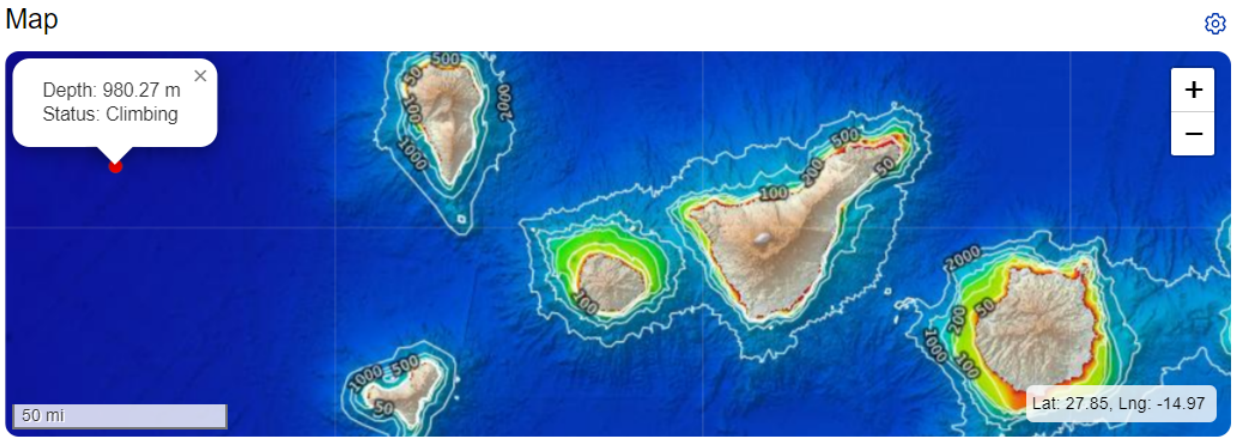

Ilustración 7.5: Mapa vista detallada

Las campañas de monitoreo acústico pasivo generan un gran volumen de datos que pueden ser difíciles de gestionar y analizar. La falta de interfaces que integren datos específicos de los gliders junto con las muestras de audio complica aún más esta tarea. Este Trabajo de Fin de Título (TFT) pretende ser un paso inicial hacia un sistema de visualización de datos cómodo, rápido y sencillo, que abarque tanto los datos del glider como las muestras de audio.

Capítulo 8

Conclusiones y trabajo futuro

El proyecto de Trabajo de Fin de Título ha culminado en el desarrollo de una aplicación web completamente funcional dirigida a usuarios interesados en el monitoreo acústico pasivo. El enfoque de este proyecto es innovador en su ámbito, ya que extrae información tanto de los ficheros de navegación del glider como de los archivos de audio WAV recogidos por el mismo.

Esta aplicación destaca por su interfaz intuitiva y ergonómica, diseñada para facilitar el manejo incluso cuando se trabaja con grandes volúmenes de datos. Además, se ha logrado una adaptabilidad efectiva a diferentes tamaños de pantallas de ordenadores, asegurando una experiencia consistente para todos los usuarios.

Uno de los logros significativos del proyecto ha sido la optimización de la velocidad en la carga de datos, fundamental para el manejo eficiente de archivos de audio y navegación del glider. Esta optimización no solo mejora la eficiencia del análisis, sino que también proporciona una experiencia fluida al usuario durante el uso intensivo de datos.

La aplicación demuestra una notable versatilidad al trabajar con una variedad de ficheros, incluyendo formatos como los archivos de audio WAV y los datos de navegación del glider. Esta capacidad de manejar diferentes tipos de datos refuerza su utilidad tanto en el ámbito científico, al extraer información valiosa para el monitoreo acústico, como en el ámbito educativo, al ofrecer herramientas didácticas para estudios relacionados.

8.1. Comparación del proyecto con respecto a otras herramientas

Este Trabajo de Fin de Título no pretende sustituir a las herramientas mencionadas, sino ser una herramienta complementaria que facilite el muestreo de grandes volúmenes de datos tanto del glider como de los archivos de audio de manera simple y útil. Las principales diferencias y ventajas de esta aplicación incluyen:

1. **Enfoque Dual:** Capacidad para muestrear y visualizar tanto los datos específicos del glider como los datos de audio recogidos durante las campañas, integrando ambos tipos de datos en una sola interfaz.
2. **Usabilidad:** Puesto que se presta alta atención a la estética y usabilidad de la aplicación, diseñada para ser accesible y fácil de usar tanto para estudiantes como para científicos. Esto asegura que cualquier persona, independientemente de su nivel de experiencia técnica, pueda acceder y analizar los datos de manera intuitiva.
3. **Educacional y Científico:** La aplicación está especialmente diseñada para su uso en entornos educacionales y científicos, proporcionando una herramienta que no solo es funcional sino también visualmente atractiva y pedagógica.

8.2. Trabajo a futuro

A pesar de que el proyecto es funcional en un servidor local, existen diversos aspectos que podrían ser mejorados y ampliados. Entre las posibles mejoras y ampliaciones destacamos:

1. **Mejora del servidor:** Actualmente, la API funciona en local. Lo ideal sería implementar un servidor remoto y actualizar las direcciones locales usadas en la aplicación web por direcciones públicas.
2. **Interactividad en las gráficas:** En el proyecto actual, las gráficas generadas a partir de los ficheros de audio son estáticas. Sería beneficioso añadir interactividad a estas gráficas. Dado que las librerías disponibles para trabajar con gráficas de audio con tantos datos son limitadas, sería necesario un trabajo en profundidad para implementar esta funcionalidad.
3. **Posibilidad de uso de ficheros propios por el usuario:** Se podría permitir a los usuarios web subir sus propios ficheros, no solo utilizar los que están disponibles en el servidor. Para esto, sería necesario crear ficheros temporales en el servidor que procesen los datos adjuntados por el usuario y sean eliminados posteriormente para evitar la saturación del servidor.

En conclusión, el proyecto ha sentado una base sólida para una herramienta útil y eficiente en el monitoreo acústico pasivo, cumpliendo con sus objetivos iniciales de proporcionar una plataforma funcional, versátil y de fácil manejo. Las mejoras sugeridas prometen enriquecer aún más esta aplicación, haciéndola más adaptable a las necesidades cambiantes de sus usuarios y asegurando su relevancia en futuros desarrollos dentro del campo.

Bibliografía

[1] "SeaExplorer." https://www.cascoantiguopro.com/es/oceanografia/vehiculos-roboticos-oceanografia/seaexplorer.html. Accedido por última vez el 20/06/2024.

[2] "Audacity." https://www.audacityteam.org. Accedido por última vez el 20/06/2024.

[3] "Raven Pro: Obtener la herramienta." https://store.birds.cornell.edu/products/raven-pro-1-6-us-only. Accedido por última vez el 20/06/2024.

[4] "Matplotlib: Página oficial." https://matplotlib.org.

[5] "Triton: Obtener el paquete." https://www.cetus.ucsd.edu/technologies_triton.html. Accedido por última vez el 20/06/2024.

[6] "PAM Guard." https://www.pamguard.org. Accedido por última vez el 20/06/2024.

[7] "PAM Lab." https://static1.squarespace.com/static/52aa2773e4b0f29916f46675/t/6050d5d3bb796f06d877da20/1615910358455/PAMlab+INT+Brochure.pdf. Accedido por última vez el 20/06/2024.

[8] B. Shneiderman, "Designing the user interface: Golden rules," 1997.

[9] E. R. Tufte, *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphics Press, 1986.

[10] "Chart.js: Página oficial." https://www.chartjs.org. Accedido por última vez el 20/06/2024.

[11] "React: Página oficial." https://es.react.dev. Accedido por última vez el 20/06/2024.

[12] "Leaflet: Página oficial." https://leafletjs.com. Accedido por última vez el 20/06/2024.

[13] "React Super Responsive Table." https://www.npmjs.com/package/react-super-responsive-table. Accedido por última vez el 20/06/2024.

[14] "Python: Página oficial." https://www.python.org. Accedido por última vez el 20/06/2024.

[15] "Pandas: Página oficial." https://pandas.pydata.org. Accedido por última vez el 20/06/2024.

[16] "Flask: Documentación." https://flask-es.readthedocs.io. Accedido por última vez el 20/06/2024.

[17] "Datetime: Documentación de Python." https://docs.python.org/es/3/library/datetime.html. Accedido por última vez el 20/06/2024.

[18] "Os: Documentación de Python." https://docs.python.org/es/3.10/library/os.html. Accedido por última vez el 20/06/2024.

[19] "SciPy: Página oficial." https://scipy.org. Accedido por última vez el 20/06/2024.

[20] "Threading: Documentación de Python." https://docs.python.org/es/3.8/library/threading.html. Accedido por última vez el 20/06/2024.

[21] "Base64: Documentación de Python." https://docs.python.org/es/3/library/base64.html. Accedido por última vez el 20/06/2024.

[22] "Io: Documentación de Python." https://docs.python.org/3/library/io.html.

[23] "Javascript: Documentación de JavaScript." https://developer.mozilla.org/es/docs/Web/JavaScript. Accedido por última vez el 20/06/2024.

[24] "DOM." https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. Accedido por última vez el 20/06/2024.

[25] "How to Use PropTypes in React." https://www.freecodecamp.org/news/how-to-use-proptypes-in-react/. Accedido por última vez el 20/06/2024.

[26] "Adapter Date-fns: Documentación de Chart.js." https://www.npmjs.com/package/chartjs-adapter-date-fns/v/0.1.0. Accedido por última vez el 20/06/2024.

[27] "WMS." https://www.oracle.com/es/scm/logistics/warehouse-management/what-is-warehouse-management/. Accedido por última vez el 20/06/2024.

[28] "OGC." https://www.ogc.org. Accedido por última vez el 20/06/2024.

[29] "EMODnet: Bathymetry." https://ows.emodnet-bathymetry.eu. Accedido por última vez el 20/06/2024.

[30] "QGIS: Página oficial." https://www.qgis.org/es/site/. Accedido por última vez el 20/06/2024.

[31] "Material-UI: Documentación MUI." https://mui.com.

[32] "React-Select: Página oficial." https://react-select.com/home. Accedido por última vez el 20/06/2024.

[33] "Figma: Página oficial." https://www.figma.com. Accedido por última vez el 20/06/2024.

[34] "Canva: Página oficial." https://www.canva.com. Accedido por última vez el 20/06/2024.

[35] "GitHub: Página oficial." https://github.com/. Accedido por última vez el 20/06/2024.

[36] "Visual Studio Code: Página oficial." https://code.visualstudio.com/. Accedido por última vez el 20/06/2024.

[37] "Scrum." https://www.scrum.org/learning-series/what-is-scrum/. Accedido por última vez el 20/06/2024.

[38] "Hooks en React." https://legacy.reactjs.org/docs/hooks-intro.html. Accedido por última vez el 20/06/2024.

[39] "Working with WMS." https://www.qgistutorials.com/en/docs/3/working_with_wms.html. Accedido por última vez el 20/06/2024.

[40] "Añadiendo funcionalidad al proyecto Leaflet: Mapa de localización y controles." https://zonegis.es/anadiendo-funcionalidad-al-proyecto-leaflet-mapa-de-localizacion-y-controles/. Accedido por última vez el 20/06/2024.

[41] "Leaflet.Coordinates: Repositorio de GitHub." https://github.com/MrMufflon/Leaflet.Coordinates. Accedido por última vez el 20/06/2024.

[42] J. J. Benedetto, *Harmonic Analysis and its Applications*. Boca Raton, FL: CRC Press, 1996.

[43] "SonarQube." https://www.sonarsource.com/products/sonarqube. Accedido por última vez el 20/06/2024.

[44] "Page Speed." https://pagespeed.web.dev. Accedido por última vez el 20/06/2024.

[45] "Vercel." https://vercel.com/. Accedido por última vez el 20/06/2024.