Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

Grado en Ingeniería Informática

Informatikako Ingeniaritzako gradua

Proyecto fin de grado

Gradu amaierako proiektua

Diseño y desarrollo de una app móvil multiplataforma que detecta la contaminación atmosférica mediante fotografías

Aitor Morais Miñambres

Director: Diego Casado Mansilla

Bilbao, Febrero de 2023

Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

Grado en Ingeniería Informática

Informatikako Ingeniaritzako gradua

Proyecto fin de grado

Gradu amaierako proiektua

Diseño y desarrollo de una app móvil multiplataforma que detecta la contaminación atmosférica mediante fotografías

Aitor Morais Miñambres

Director: Diego Casado Mansilla

Bilbao, Febrero de 2023

Resumen

Este proyecto de fin de grado trata del diseño y desarrollo de una aplicación no nativa con la que podremos medir la cantidad de partículas en suspensión (PM2.5, o PM10)que se encuentran en el aire. Para este fin, la aplicación hace uso de procesamiento de imágenes. En primer lugar, contamos con una página web que es la encargada de mostrar los datos de la contaminación que hemos obtenido mediante la carga y procesamiento de imágenes. En segundo lugar, disponemos de un Bot que puede recibir imágenes, que el usuario habrá tomado, estas imágenes se analizaran y se devuelven los resultados de la contaminación detectada en la fotografía. De la fotografía nos interesa su ubicación para poder geo posicionarla, una vez que el análisis de la imagen haya sido efectuado obtendremos los datos de la contaminación en el aire, estos datos serán almacenados en una base de datos y posteriormente cargamos estos datos en nuestra página web. Una vez la carga haya sido efectuada con éxito, podremos visualizar los datos con facilidad en una web cuyo objetivo será permitirnos ver de forma amigable la contaminación detectada en el aire en ciertas zonas.

Descriptores PM2.5,PM10,Bot, Web,Telegram

Índice general

Índice de cuadros

Capítulo 1

Antecedentes y Justificación

1.0.1. Antecedentes y estado del arte

La contaminación atmosférica es un problema generalizado que afecta tanto a la salud humana como al medio ambiente. Para adoptar medidas preventivas, es crucial detectar a tiempo la contaminación atmosférica. Sin embargo, la medición actual de la contaminación atmosférica se basa en estaciones de control caras y poco accesibles. El objetivo de mi proyecto es ofrecer una solución práctica y asequible al problema de la detección de la contaminación atmosférica utilizando imágenes captadas por el usuario. Algún ejemplo de esto serían los siguientes:

- Análisis de laboratorio de muestras de aire: Este proceso es costoso y requiere tiempo y esfuerzo para recopilar y enviar muestras a un laboratorio.
- Sistemas de monitoreo de redes: Estos sistemas requieren la instalación de sensores en varios puntos de la ciudad y suelen ser costosos y difíciles de acceder para la mayoría de la población.
- Tecnologías de satélite: Este enfoque es costoso debido a la necesidad de desarrollar y lanzar satélites especializados.

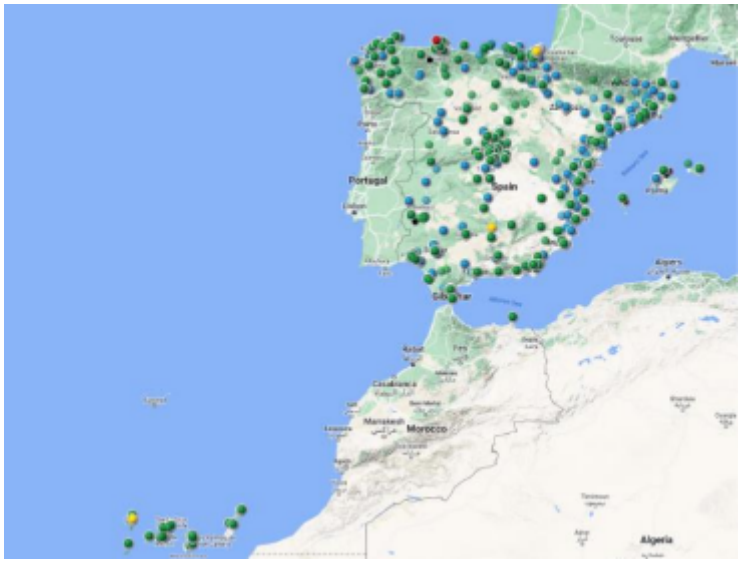
Actualmente, existen soluciones costosas y poco accesibles para detectar la contaminación del aire. Sin embargo, la tecnología de procesamiento de imágenes ha avanzado significativamente en los últimos años y es posible desarrollar una solución económica y accesible para detectar la contaminación del aire a través de imágenes capturadas por los usuarios. Este proyecto se diferencia de las tecnologías nombradas anteriormente, ya que proporciona una solución accesible y asequible que utiliza tecnologías existentes, como puede ser el uso de bot de Telegram, cámaras móviles y la API de Google maps. Además, al permitir a los usuarios contribuir con datos sobre la calidad del aire, este proyecto también aborda la necesidad de una mayor participación ciudadana en la monitorización de la calidad del aire. Existen iniciativas con objetivos similares a la de este proyecto.

- **Vigilantes del aire:** Proyecto FECYT liderado por IBERCIVIS donde se utilizaron plantas de fresa como estaciones de monitorización de la contaminación ambiental por metales. Aplicando

técnicas magnéticas son capaces de identificar la cantidad de metales acumulados por las hojas de estas plantas, estimando entonces la calidad del aire de su entorno.

- **Scicling:** una iniciativa de comunicación científica que ha puesto en contacto al Dr. Marín-Menéndez con más de 2000 estudiantes de secundaria de Uruguay, Canarias y Asturias en los últimos tres años. Esta propuesta se dirige no sólo a los alumnos y a las comunidades educativas, sino también a la figura del profesor. No sólo es investigadora, sino que invierte en métodos científicos, inspira carreras investigadoras y fomenta el uso de la bicicleta como medio de transporte intra e interurbano saludable y sostenible.

A nivel estatal, los niveles de PM10 son consistentemente altos, principalmente por eso. Las concentraciones aumentaron naturalmente debido a la invasión de las masas de aire africanas. En 2021 se ha modelado y estimado la cantidad de material y partículas en la región y se ha mostrado que las concentraciones continentales y en términos de cálculo, se resta dicha concentración a las medidas provistas por las escasas estaciones de referencia a nivel estatal. Los resultados son aceptables para mediciones establecidas por la UE. Sin embargo, el informe de 2021 de la División de Transición comparó los datos de la OMS con los datos del Ministerio de transición ecológica y reto demográfico, se puede observar que en la mayoría de las estaciones base exceden sus niveles de VLD y VLA en sus medidas. Aún es más, eliminar del cálculo las medidas estimadas por el polvo sahariano no deja ser un método para evitar un factor de confusión, pero la realidad es que, medido o no, ese polvo se respira igualmente. En resumen, podemos constatar que respiramos un aire de baja calidad. Además, el número de estaciones fijas por habitante a nivel estatal es bajo (600 en total), cabe mencionar que existen zonas que requieren de un mayor seguimiento e inspección de la calidad del aire, ya que superan los niveles de exposición diaria y anual a PM10 recomendados por la OMS. Por todo esto, la razón principal de este proyecto es dar una herramienta a la ciudadanía para que sea más accesible acceder a la información relevante de la calidad del aire que respiramos.
Adjunto imagen de las estaciones disponibles a nivel estatal.



1.0.2. Justificación

La justificación técnica de este proyecto se basa en la necesidad de una solución asequible y práctica para la detección de la contaminación atmosférica. Actualmente, la mayoría de la población considera que muchas de las soluciones existentes son caras y de difícil acceso, lo que limita su capacidad para controlar y comprender la calidad del aire en sus comunidades.

Además, la contaminación del aire es un problema global que afecta la salud humana y el medio ambiente, por lo que es importante que se desarrollen soluciones accesibles y asequibles para monitorear y comprender su impacto. Al permitir a los usuarios contribuir con datos sobre la calidad del aire, nuestro proyecto también promueve la participación ciudadana en la monitorización de la calidad del aire.

Desde el punto de vista técnico, este proyecto es viable gracias a la disponibilidad de tecnologías como las cámaras de los teléfonos móviles y las API de Google Maps, que permiten recopilar y procesar información sobre la calidad del aire de forma fácil y asequible. Además, el uso de un bot de Telegram y un sitio web construido en Django permite integrar y visualizar los resultados de forma sencilla, lo que facilita a los usuarios la comprensión y el uso de los datos.

En conclusión, la justificación técnica de este proyecto se basa en la necesidad de una solución asequible y práctica para la detección de la contaminación atmosférica y en la disponibilidad de tecnologías existentes que permiten su desarrollo y aplicación.

Capítulo 2

OBJETIVOS Y ALCANCE

2.1. Objetivos

El objetivo de este trabajo es el de Diseño y Desarrollo de una App móvil multiplataforma que detecte contaminación atmosférica mediante fotografías, para esto haremos uso de un bot de Telegram y una web que nos mostrara el contenido de manera amigable.

- El sistema ha de ser desacoplado para que este pueda ser fácilmente mantenible y escalable
- El sistema será fluido y con tiempos de respuesta aceptables

Se alcanzara el objetivo mediante una elección coherente de arquitectura, software y una buena programación del sistema

2.2. Alcance

Se trata del diseño y desarrollo de una app móvil multiplataforma capaz de detectar la contaminación atmosférica mediante fotografías. Para poder lograr esto haremos uso de un bot de Telegram en el que el usuario subirá la imagen en la que se han quedado las partículas y también mandara la ubicación en donde la imagen fue tomada. Una vez el bot tenga la imagen y la ubicación la imagen será procesada para contar las partículas en ella, con esta información se mandara una petición HTTP de tipo post a la base de datos de la web y alojara la información para posteriormente mostrarla en el mapa. A continuación, se define brevemente el alcance de cada una de las partes implicadas en el sistema. En conclusión, el sistema ha de reflejar la estructura mencionada con base en un desarrollo software coherente. Queda fuera del alcance del sistema la aplicación de técnicas de inteligencia artificial para procesar las imágenes, así como para reconocer el tipo de imagen, o forzar al usuario a meter una imagen especifica, es decir, se dará por hecho que el usuario siempre meterá una imagen correcta. También quedan fuera temas relacionados con ciberseguridad a la hora de mandar la información a la base de datos, esta parte ha quedado fuera, ya que el proyecto se desarrolla en la una máquina local y solo yo tengo acceso a esta información. Siendo

mejoras que se podrían llevar a cabo en futuras ampliaciones del proyecto o en un despliegue final. A continuación, se define brevemente el alcance de cada una de las partes implicadas en el sistema.

2.2.1. Bot de Telegram

El bot se encarga de recibir la información relevante, a la ubicación y la imagen. Esta parte implica la creación del bot y el uso de las API oficiales de Telegram y el desarrollo del código referente a la puesta en marcha del bot.

2.2.2. Web

La página web nos sirve para visualizar de una manera amigable la información que ha sido cargada previamente por otros usuarios y está alojada en una base de datos propia de Django.

2.2.3. Procesamiento de la imagen:

Se usa una función llamada suavizar incluida en el bot de Telegram para procesar la imagen subida por el usuario, esta función tiene como objetivo procesar una imagen específica para suavizarla y resaltar las manchas presentes en ella.

Capítulo 3

DESARROLLO

3.1. Fase Inicial

Una vez definido el objetivo del proyecto pasaremos a diferencias las partes más relevantes que se desean implementar.

- Bot de Telegram
- Algoritmo para procesar imágenes
- Página web

Tras enumerar todos los componentes del sistema, es necesario definir su estructura de diseño. El término .ªrquitectura de software"se refiere al proceso de definir los numerosos componentes que conforman un sistema y cómo se comunican entre sí utilizando un conjunto de directrices y abstracciones. En este caso, se ha optado por una arquitectura de microservicios, que se describe a continuación.

3.2. Arquitectura

La arquitectura de la aplicación está diseñada para gestionar el procesamiento y análisis de imágenes de partículas y su localización. Consta de tres componentes clave: el bot de Telegram, el módulo de procesamiento de imágenes y la interfaz web.

El bot de Telegram, codificado en Python, permite a los usuarios cargar una imagen de partículas y su ubicación. La imagen es examinada por el módulo de procesamiento de imágenes, que también determina dónde se encuentran las partículas. Este módulo se construye con Python y varias bibliotecas de procesamiento de imágenes, como OpenCV.

La interfaz web se encarga de mostrar los resultados del procesamiento de imágenes y la localización de partículas.

Toda la arquitectura está diseñada para ser modular y fácilmente ampliable, de modo que en el futuro puedan añadirse nuevos algoritmos o funciones de procesamiento de imágenes. Además, el sistema está diseñado para ser escalable.

Para nuestro caso en particular se identifican los siguientes servicios:

- **Bot de Telegram**: Se encargará de la interacción a través de Telegram con el usuario, para esto el usuario mandará unos datos al bot y una vez recolectados y procesados estos datos serán mandados a la web.
- **Procesamiento de imagen:** Una vez el usuario ha subido la foto, se procesará la imagen usando ciertos filtros y conversión de la imagen a escala de grises para obtener el resultado.
- **Página web:** La utilizaremos para mostrar los resultados obtenidos, así como la ubicación donde se tomó la foto.

3.3. Diagrama de flujo

3.4. Bot de Telegram

Se comenzó el desarrollo por la parte de la creación del bot de Telegram, ya que será el elemento con el cual el usuario entrara en contacto por primera vez y el que dará a pie a poder usar todo el sistema de la manera correcta. Además, al ser algo nuevo que podría requerir más tiempo, se vio acertado empezar por este apartado, ya que la creación de los otros elementos se tenía experiencia previa gracias a la formación académica recibida a lo largo de la carrera.

El proceso de desarrollo incluye la lectura de tutoriales y documentación oficial de Telegram sobre cómo crear bots en Telegram. El primer paso es crear un bot en Telegram, para lo que es necesario instalar la aplicación de mensajería, ya que es así como se lleva a cabo el proceso. Telegram cuenta con un bot oficial que los desarrolladores pueden utilizar para crear bots y actualizar sus datos.

El contacto con el bot oficial de Telegram @BotFather se establece a través de una ventana de chat privada. El bot proporciona una lista de comandos que se pueden utilizar para crear y gestionar bots. En esa lista de comandos se encuentra el comando /newbot, que permite crear un nuevo bot. Este es el primer comando que se utiliza. A continuación, el bot oficial solicita un nombre de usuario para crear el nuevo bot, que debe ser distinto de cualquier otro bot ya existente.

Una vez creado el bot, BotFather de Telegram proporciona el ID público del bot, que cualquier usuario puede utilizar para iniciar una conversación. Además, proporciona el token necesario para utilizar la API de bots de Telegram. De esta forma, se tiene acceso a todos los métodos de Telegram disponibles para conocer el funcionamiento del nuevo bot, incluyendo la capacidad de leer y enviar mensajes cuando un usuario contacta con él y un amplio abanico de otras capacidades.

Es crucial tener en cuenta que el token proporciona acceso completo al bot desarrollado; como resultado, por razones de seguridad, se debe tener cuidado de no revelar el token a terceros. En caso de filtración,

se puede revocar el token actual utilizando el comando /revoke antes de generar uno nuevo.

3.5. Instanciación y definición del bot

Para instanciar el bot se utiliza el módulo Telegram y su extensión ApplicationBuilder. En primer lugar, se crea una instancia de ApplicationBuilder, a la que se le pasa el token del bot obtenido al crear el bot en Telegram. El bot tiene como función procesar imágenes recibidas por el usuario y enviar los datos de la imagen procesada a una API.

3.5.1. Interacción con el usuario

Cuando el usuario envía una imagen al bot, se utiliza la función suavizar para suavizar la imagen y contar los puntos detectados en ella. Luego, se envían los datos de la imagen procesada, como la contaminación detectada y la ubicación de la imagen, a una API mediante una solicitud HTTP POST. El bot también tiene la capacidad de recibir la latitud y longitud del usuario y guardar esta información para enviarla junto con los datos de la imagen procesada a la API. Utiliza también la librería "logging"para registrar eventos y errores en un archivo de registro.

3.6. Procesar la imagen

Se hace uso de la biblioteca OpenCV para procesar una imagen. La función se llama "suavizarz toma como parámetro el nombre de una imagen. El proceso en sí consiste en varios pasos:

1. Primero, se verifica que la ruta de la imagen es válida utilizando la función os.path.isfile().
2. Luego, se carga la imagen utilizando la función çv2.imread()".
3. A continuación, se aplica un filtro de diferencia de mediana a la imagen cargada utilizando la función çv2.medianBlur()".
4. Se convierte la imagen suavizada a escala de grises utilizando çv2.cvtColor()".
5. Se calcula la diferencia entre la imagen original y la imagen suavizada utilizando çv2.absdiff()".
6. Finalmente, se guarda la imagen con las manchas resaltadas en el disco utilizando çv2.imwrite()".

3.7. Interfaz web

La interfaz web cumple con el objetivo de mostrar la información. Esta interfaz está construida con Django y tiene un front-end que utiliza JavaScript y HTML. La interfaz web también se comunica con una base de datos para almacenar las imágenes procesadas y la ubicación de los componentes. En este caso, se está utilizando SQLite como motor de base de datos y el archivo "db.sqlite3.[en] el directorio principal como archivo de la base de datos.

3.8. Almacenamiento de datos

Se utiliza la base de datos sqlite cabe destacar el uso de esta base de datos por sus siguientes beneficios:

1. Fácil de configurar: SQLite es una base de datos file-based, lo que significa que no requiere ninguna configuración adicional para trabajar con Django.
2. Pequeño tamaño: SQLite tiene un tamaño de archivo pequeño, lo que lo hace ideal para aplicaciones web pequeñas
3. Portabilidad: SQLite es una base de datos portable, lo que significa que puede ser usada en diferentes sistemas operativos sin necesidad de configuraciones adicionales.
4. Protección contra corrupción de datos: SQLite tiene mecanismos de protección contra corrupción de datos, lo que garantiza la integridad de los datos en caso de fallas del sistema.
5. Sin procesos de servidor: SQLite no requiere procesos de servidor para funcionar, lo que reduce la sobrecarga en el sistema y aumenta la escalabilidad.

La clase Übicación"define un modelo con cuatro campos: nombre, lat, lng y contaminación, y un campo de fecha. Cada vez que se crea una nueva instancia de la clase Ubicación, se creará una nueva fila en la tabla correspondiente en la base de datos con los valores especificados para cada campo. El campo "fecha"se actualizará automáticamente con la fecha en la que se creó la instancia.

3.9. API

En este proyecto se utilizan tanto la API de Google Maps como la de Telegram para desarrollar un sistema integral de detección de la contaminación atmosférica.

El desarrollo de un bot que permite a los usuarios enviar su ubicación y la imagen capturada de la contaminación en una hoja blanca hace uso de la API de Telegram. El bot recibe los datos del usuario a través de la API y procesa la imagen para contabilizar los puntos de contaminación que ha encontrado. A continuación, mediante una petición HTTP POST, la API permite enviar los datos procesados a la web Django.

Luego, el sitio web de Django utiliza la API de Google Maps para mostrar un mapa con marcadores que muestran la ubicación exacta donde el usuario cargó la foto y la cantidad de partículas contaminantes detectadas en esa ubicación. La API permite integrar fácilmente la funcionalidad de mapas en la web, y es esencial para visualizar la información procesada por el bot de Telegram.

En resumen, este proyecto combina las API de Telegram y Google Maps para crear una atmósfera completa que permite a los usuarios enviar su ubicación y fotos, procesar las imágenes para contar los puntos de contaminación detectados y mostrar la información en un mapa.

Capítulo 4

PLANIFICACIÓN

4.1. Investigación

Para el desarrollo de este proyecto se deben llevar a cabo una serie de investigaciones en las tecnologías que se van a usar, específicamente Django, Python y Telegram. Partimos de la base que ya se tenían ciertos conocimientos previos en torno a Django y Python gracias a la formación recibida en la carrera. Para poder desarrollar de manera satisfactoria el proyecto podemos identificar varias subtareas.

- Uso de la API de Telegram
- Uso de la API de Google Maps
- Uso de Django y Python
- Tratamiento de imágenes

4.1.1. Creación del bot

La creación del bot incluye la interacción con el bot maestro de Telegram y obtener el token de autorización. El proceso de definición implica el desarrollo del código para poder recibir la ubicación y las imágenes y la lógica de comportamiento ante ellos.

4.1.2. Tratamiento de la imagen y obtención de resultados

Para llevar a cabo esta tarea se deben entender los detalles detrás del script que se utiliza para tratar la imagen. Este script realiza tareas como cargar la imagen, aplicar un filtro de diferencia de mediana, convertir a escala de grises, aplicar la diferencia entre la imagen original y la imagen suavizada con el filtro de mediana, y finalmente, guardar la imagen con las manchas resaltadas.

Después de realizar esta primera tarea, el siguiente paso es contar los puntos o manchas en la imagen resaltada. Esto se hace aplicando threshold binario para eliminar el ruido, encontrar los contornos y

dibujarlos en la imagen original en rojo, y guardar la imagen con los puntos detectados marcados en rojo. Finalmente, se imprime el número de puntos encontrados.

4.1.3. Enviar datos a la web

Para enviar la información correctamente a la web y poder almacenarla en la web primero se verifica que la imagen sea válida, luego se hace uso del los scripts necesarios para procesar la imagen y contar los puntos. Una vez esto ha sido hecho se crea un diccionario con la información de la imagen que contendrá los datos que nos interesan para mandar a nuestra web, por último se manda una petición POST a la dirección "http://127.0.0.1:8000/api/data/çon el diccionario en formato JSON.

Capítulo 5

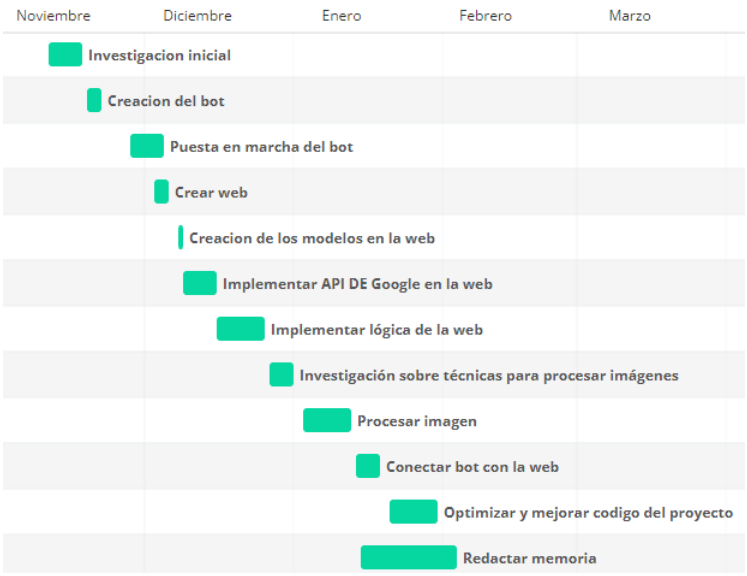
CARGAS DE TRABAJO

5.1. Resumen de cargas de trabajo

A continuación, se presentan y detallan las cargas de trabajo que se derivan de cada una de las tareas del proyecto.

1. Investigación inicial
2. Creación del bot
3. Puesta en marcha del bot
4. Crear web
5. Crear el modelo para almacenar datos en la web
6. Implementar API DE Google en la web
7. Implementar lógica de la web
8. Investigación sobre técnicas para procesar imágenes
9. Procesar imagen
10. Conectar bot con la web
11. Optimizar y mejorar código del proyecto
12. Redactar memoria

5.2. Diagrama de Gant


Noviembre
Diciembre
Enero
Febrero
Marzo
Investigacion inicial
Creacion del bot
Puesta en marcha del bot
Crear web
Creacion de los modelos en la web
Implementar API DE Google en la web
Implementar lógica de la web
Investigación sobre técnicas para procesar imágenes
Procesar imagen
Conectar bot con la web
Optimizar y mejorar codigo del proyecto
Redactar memoria

Capítulo 6

PRESUPUESTO

6.1. Desglose del presupuesto

El coste de los recursos y gastos de personal necesarios para desarrollar la solución del sistema propuesto. Estos gastos se desglosan en dos partes: los costes de personal asociados a las tareas de dirección, diseño, programación; y el coste de los recursos necesarios o mínimos para su implantación.

6.1.1. Gastos en personal

Tomando una tasa media de 50 euros por hora, sabiendo que este proyecto tomo un tiempo dedicado al desarrollo software de alrededor de 350 horas, el precio sería de, 17500 euros, Suponiendo que haría falta un director de proyecto con una estimación de salario de 90 euros por hora sumariamos un total de 2250 euros por 25 horas de trabajo. Lo que nos deja un presupuesto de, 19750 euros para el personal.

Gastos en personal			
Trabajador	Coste por horas €/h	Horas	Coste
Programador	50	350	17500€
Director	90	25	2250€

Cuadro 6.1: Gasto en personal

6.1.2. Gastos en software

Las herramientas software empleadas para este proyecto tienen licencia open source, por lo tanto, no se precisa la realización de un pago para su uso. El único pago que deberíamos de afrontar sería el despliegue de la web, el registro del dominio, así como el costo de certificado ssl.

- Registro de dominio: entre 10 euros por año

- Hosting para la web: 150 euros por año
- Costo de certificado SSL: 70 euros por año

Teniendo en cuenta estos gastos nos deja un coste aproximado de 230 euros anuales

Capítulo 7

CONCLUSIONES

7.1. Resultados obtenidos

Al acabar este proyecto, se puede concluir que este proyecto ha cumplido con los objetivos definidos inicialmente, implementación de un bot capaz de recibir imágenes y poder procesarlas, así como mandar la ubicación. Implementar una interfaz web capaz de mostrar los datos enviados por este bot.

7.2. Posibles mejoras

Teniendo en consideración la implementación del proyecto como el diseño y desarrollo base de, se presentan una serie de mejoras de cara a un despliegue a una escala mayor o una adición de funcionalidad para alguno de los servicios desarrollados.

1. Hosting en un servidor para el bot de Telegram
2. Utilizar un modelo de aprendizaje automático entrenado con imágenes etiquetadas para clasificar la imagen. Y así poder saber si la imagen subida por el usuario corresponde con la esperada.
3. En lugar de usar @csrfexempt, se debería utilizar un mecanismo de seguridad adecuado, como un token CSRF, para proteger la aplicación de ataques CSRF.

Capítulo 8

VALORACIÓN

8.1. Ética

Este proyecto tiene dos objetivos fácilmente identificables, siendo uno el poder acercar a la ciudadanía una herramienta fácil de utilizar para poder saber la calidad del aire en cualquier lado. Poder tener acceso a la información relativa sobre la calidad del aire fácil y accesible.

Además, el uso de tecnología para la detección de la contaminación es una forma de promover la transparencia y la responsabilidad, ya que permite a los ciudadanos y a las autoridades tener acceso a información precisa y actualizada sobre los niveles de contaminación en tiempo real.

El principio de beneficencia está presente en dar una herramienta que es capaz de otorgarnos información sobre la calidad del aire. Esta herramienta puedo ayudar a la sociedad a concienciarse sobre la calidad del aire que respiran y así poder tomar medidas preventivas para proteger su salud y la de sus seres queridos. También puede llevar a la ciudadanía a interesarse por las fuentes de contaminación en su entorno y así poder evitarlas, así como participar en la lucha contra la contaminación del aire.

8.2. Personal

Por último, cabe señalar que el progreso del proyecto ha tenido un impacto positivo en la vida intelectual y personal de los estudiantes. En primer lugar, los conocimientos adquiridos a lo largo de los años de formación se han aplicado a los múltiples aspectos relacionados con el proyecto. El desarrollo de un bot de Telegram y el tratamiento de las imágenes, han requerido, no obstante, el aprendizaje de estos temas. Asimismo, ha sido enriquecedor diseñar y construir una solución que pueda servir de base o punto de partida para el eventual lanzamiento del sistema a gran escala en el futuro.

Capítulo 9

BIBLIOGRAFÍA

9.1. Enlaces

[1] Tratamiento de imágenes https://docs.opencv.org/4.x/

[2] Django documentación https://docs.djangoproject.com/en/4.1/

[3] Telegram documentación https://core.telegram.org/bots/api

[4] Despliegue https://www.youtube.com/watch?v=hlZPC5L2Itc&t=1085s

[5] Información sobre el estado de aire a nivel estatal https://www.miteco.gob.es/es/calidad-y-evaluacion-ambie
temas/atmosfera-y-calidad-del-aire/calidad-del-aire/documentacion-oficial/Analisis-CA.
aspx

Apéndice A

ANEXO

Anexo

En este anexo explicaré brevemente como usar mi proyecto para medir la calidad del aire.

A.0.1. Primeros pasos:

Lo primero que deberíamos de hacer sera comprobar si Pagina web esta operativa, para el despliegue use una plataforma llamada pythonanywhere al ser gratuita tienes que reactivarla cada tres meses,en caso de que no este operativa

Listing A.1: instalar instalar todas las dependencias

```
!pip install -r requirements.txt
```

,en caso de que este operativa tendrías que clonar este repositorio una vez descargado aplica estos comandos

Listing A.2: instalar dependecias

```
!pip install telegram
!pip install pyTelegramBotAPI
!pip install python-telegram-bot
```

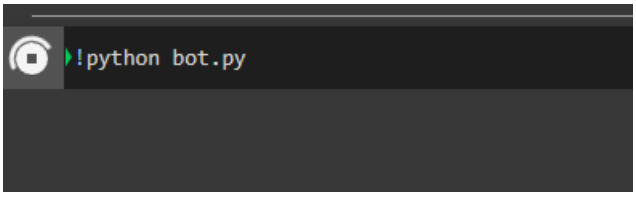
```
!pip install telegram
!pip install pyTelegramBotAPI
!pip install python-telegram-bot

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: telegram in /usr/local/lib/python3.8/dist-packages (0.0.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyTelegramBotAPI
  Downloading pyTelegramBotAPI-4.9.0.tar.gz (219 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 219.9/219.9 KB 6.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from pyTelegramBotAPI) (2.25.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->pyTelegramBotAPI) (2022.12.7)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->pyTelegramBotAPI) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->pyTelegramBotAPI) (1.24.3)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->pyTelegramBotAPI) (4.0.0)
Building wheels for collected packages: pyTelegramBotAPI
  Building wheel for pyTelegramBotAPI (setup.py) ... done
  Created wheel for pyTelegramBotAPI: filename=pyTelegramBotAPI-4.9.0-py3-none-any.whl size=203107 sha256=f57c9633436f02397907f9eb5016b5b62f2edb37046c4be4c35
  Stored in directory: /root/.cache/pip/wheels/95/a7/fd/f667b71e29e3f64f239f30195efc39de642e72f8a47a6fcd8d
Successfully built pyTelegramBotAPI
Installing collected packages: pyTelegramBotAPI
Successfully installed pyTelegramBotAPI-4.9.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting python-telegram-bot
  Downloading python_telegram_bot-20.0-py3-none-any.whl (484 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 484.3/484.3 KB 10.2 MB/s eta 0:00:00
Collecting httpx~=0.23.1
  Downloading httpx-0.23.3-py3-none-any.whl (71 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.5/71.5 KB 7.3 MB/s eta 0:00:00
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from httpx~=0.23.1->python-telegram-bot) (2022.12.7)
Collecting rfc3986[idna2008]<2,>=1.3
  Downloading rfc3986-1.5.0-py2.py3-none-any.whl (31 kB)
Collecting sniffio
  Downloading sniffio-1.3.0-py3-none-any.whl (10 kB)
Collecting httpcore<0.17.0,>=0.15.0
  Downloading httpcore-0.16.3-py3-none-any.whl (69 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.6/69.6 KB 7.9 MB/s eta 0:00:00
Collecting h11<0.15,>=0.13
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 KB 6.6 MB/s eta 0:00:00
Collecting anyio<5.0,>=3.0
  Downloading anyio-3.6.2-py3-none-any.whl (80 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.6/80.6 KB 9.0 MB/s eta 0:00:00
```

[H]

seguido usa este comando:

Listing A.3: instalar dependecias

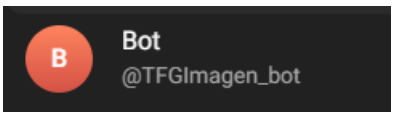
```
!python bot.py
```



deberías de ver lo mostrado en la imagen. Puede ser que veas un mensaje como el siguiente:

```
telegram.error.Conflict: Conflict: terminated by other getUpdates request; make sure that only one bot instance is running
```

en ese caso. no tendrias que lanzar el bot ya que indica que se esta ejecutando,deberias de deter la ejecucion para poder usar el bot. Accede a la aplicacion de telegram,es importante usarla desde el movil ya que por ahora no es posible mandar la ubicacion desde el ordendor introduce la ubicacion clickando el clip y



despues dandole a ubicacion,seguido añade la foto donde se encuentra plasmada la contaminacion,acto seguido accede a la web y veras la actualizacion en el mapa


Mapa
Satélite
Baquio
Urdaibai
Reserva de
la Biosfera
y Área de...
Sopelana
Castro-Urdiales
BI-637
Munguía
Getxo
N-634
BI-631
Portugalete
Guernica-Luno
Baracaldo
Bilbao
Galdácano
Mimetiz
Arrigorriaga
Sodupe
Balmaseda
AP-8
AP-68
Llodio
Arceniega
Areaza
Orozco
Urkio
Naturala