Extracción y representación de datos del satélite Sentinel
Extraction and representation of Sentinel satellite data

Por
Elam Uceda Herrero
Jiali Zheng

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería del Software
Facultad de Informática

Mercedes García Merayo

Madrid, 2019–2020

Agradecimientos

Elam Uceda Herrero

Lo primero agradecer todo su esfuerzo a Mercedes Merayo mi tutora del tfg, que a pesar de su indisposición siempre ha estado ahí para resolver cualquier duda y ayudarnos a que este proyecto se haya llevado a cabo. Igual agradecerle a mi compañero Jiali su encomiable aportación a la parte práctica, del código del programa.

Por la parte personal deseo agradecer el apoyo a toda mi familia, pero a tres personas muy importantes en mi vida deseo agradecerles de una manera especial.

En primer lugar a mi madre Elisa, que aún padeciendo una muy grave enfermedad tumoral, jamás ha dejado de darme el ánimo y apoyo a lo largo de esta carrera universitaria, y con este proyecto con el cuál deseo obtener mi título, hacer que se sienta muy orgullosa de su ingeniero.

En segundo lugar a mi tío Miguel, doctor en inteligencia artificial, cuya sabiduría me ha ayudado más en los cursos, que todas las ayudas en internet disponibles juntas.

Y por último a mi novia Ethel, la cuál conocí justo en la mitad de mi carrera, y sin sus consejos y formas de pasarme la lección o ayudarme a distraerme de mis obligaciones por un ratito, en mis épocas de agobiante estudio, no estaría hoy donde estoy.

A todos ellos Gracias.

Jiali Zheng

En primer lugar, agradezco a mi tutora del TFG, Mercedes Merayo, por todas las ayudas que nos ha dado durante el curso.

En segundo lugar, agradezco a mi compañero Elam, me ayudaba a corregir todos los textos que he escrito, ya que soy débil de redactar.

Al final, agradezco a mis padres por ser mis padres.

Love and peace.

Sobre TEFLON

Teflon(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado o Trabajos de Fin de Máster. La versión usada es la 1.3.

V:1.3 Overleaf V2 with pdfLaTeX, margin 1in, NO-bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: asciifdi@gmail.com
Despacho 110 - Facultad de Informática

Índice general

Página

Resumen

Presentamos con este proyecto de fin de grado, las competencias adquiridas a lo largo del grado de ingeniería software. Estos conocimientos han hecho posible la creación de una página web encargada de la representación de la contaminación atmosférica presente en la atmósfera, generado por el NO2.

Este proyecto consta de la conexión y extracción a tiempo real de los datos del satélite, su posterior procesamiento, y por último su representación ayudada por el sistema de elección de día a consultar para mayor precisión y comodidad del usuario.

Con este proyecto además de demostrar las competencias adquiridas en el grado, con él pretendemos ayudar a hacer de este mundo un lugar un poquito mejor, por todo esto presentamos este trabajo en calidad de TFG.

Palabras clave
Satélite, Agencia Espacial Europea, ESA, Sentinel, Sentinel-5P, NetCDF, Python, Leaflet, Git, Github Pages, Automatización

Abstract

With this graduation project, we present the competencies acquired throughout the software engineering degree. This knowledge has made possible the creation of a web page in charge of representing the air pollution present in the atmosphere, generated by NO2.

This project consists of the connection and extraction of data in real-time from the satellite, subsequent processing, and finally the representation helped by the system of choice of day to consult for greater accuracy and convenience of the user.

With this project besides demonstrating the competencies acquired in the degree, with it we intend to help make this world a little better, for all this we present this work as graduation project.

Keywords
Satelite, European Space Agency, ESA, Sentinel, Sentinel-5P, NetCDF, Python, Leaflet, Git, Github Pages, Automation

Capítulo 1

Introducción

En este capítulo realizaremos una introducción al proyecto que hemos desarrollado y que detallamos a lo largo de esta memoria. Comenzaremos con la motivación que nos impulsó a mi compañero y a mi a elegir la realización de este proyecto. Tras ello, expondremos los objetivos que pretendemos alcanzar durante el desarrollo.

1.1. Motivación

La exploración espacial es un tema apasionante en esta época en la que nos encontramos, debido a que es muy posible que asistamos, en esta década que comienza, a la primera huella humana sobre Marte, sueño perseguido por tantos, durante tantos años, y el que según parece estamos muy cerca de conseguirlo.

No obstante, no debemos dejarnos eclipsar por estas ambiciones, propias de los humanos, ya que somos una especie exploradora. En nuestras manos tenemos una responsabilidad mucho mayor, que es cuidar nuestro planeta.

La contaminación y deforestación que asola nuestra Tierra es un problema real y acuciante y es hora y momento de atajarlo. En esto encontramos mi compañero y yo la motivación de realizar este TFG, porque vemos una oportunidad de aportar algo que ayude a luchar contra la contaminación y degradación de nuestro planeta. Esta aportación consiste en la detección de los gases contaminantes producidos por la actividad humana para poder ayudar a gobiernos y poblaciones a visualizar las nubes de contaminación. De entre todos los gases contaminantes hemos elegido el NO2, como el gas objetivo de este trabajo. El dióxido de nitrógeno es producido en su mayoría por la acción humana y en muy menor medida de forma natural. En la naturaleza lo encontramos por incendios forestales y en las nubes de gas y humo generadas en las erupciones volcánicas. El gas generado por la acción humana, es producido por los motores de combustión interna, centrales eléctricas y producción industrial. En general, cualquier combustión de derivados del petróleo.

Este proyecto centra su interés en ayudar a detectar estas concentraciones de gas contaminante, para con esa información tratar de evitar que haya grandes aglomeraciones de dióxido de nitrógeno en áreas pequeñas habitadas, como ciudades o áreas en las que haya mucho tránsito de personas.

Con esto, intentaremos contribuir a esta temprana detección de estas áreas donde se concentra el gas, para que de esta manera, se pueda trabajar en reducir esas cantidades y minimizar la exposición de las personas a respirarlo.

La inhalación de NO2 es muy perjudicial para la salud humana pues según un estudio de la Unión Europea [13], la mala calidad del aire provoca enfermedades que pueden devenir en muerte prematura, afectando de manera anual a aproximadamente 400.000 personas. Esta cifra es alarmante y por ello es deber de todos evitar que nuestra actividad productiva y bienestar social, tenga como coste las graves consecuencias en salud para las población a la que se somete a este gas.

Los principales problemas de salud derivados de la inhalación, son los siguientes:

1. En primer lugar, el problema de salud que manifiestan las personas que acostumbran a respirar en su día a día NO2, desarrollan enfermedades respiratorias, como la bronquitis aguda, la disminución de capacidad de oxigeno que pueden retener los pulmones, lo cual recorta mucho la resistencia física de las personas y aumenta el agotamiento, y por último en algunos casos puede derivar en un enfisema pulmonar (que consiste en una obstrucción crónica del órgano).

2. Otros problemas de salud añadidos son los relacionados con el corazón. Las partículas en suspensión superiores a 2.5 micras generan problemas cardíacos, aumentando el riesgo de infarto de miocardio, además de multiplicar las posibilidades de padecer hipertensión e hipotensión

3. Ya en menor medida, pero en caso de producirse serían padecimientos graves, este gas aumenta la posibilidad de sufrir problemas renales, ictus y favorece que surjan algunos casos de cáncer.

En cuanto a las personas, unas son más susceptibles de ver empeorada su salud respecto a otras. Dentro de estas poblaciones de riesgo encontramos, mujeres embarazadas, los niños y recién nacidos, además de personas con problemas respiratorios o fumadores.

Los estudios no vinculan directamente la inhalación con el aumento de la mortalidad, pero la exposición a este gas es uno de los componentes que más contribuyen al deterioro general de la salud.

Los motivos antes mencionados, son los que nos hicieron elegir la temática del proyecto en cuanto a nuestras ganas de aportar un bien a la humanidad y la comunidad científica, pero existe otro motivo por el cual también decidimos realizar este proyecto.

Creemos, mi compañero y yo, que la ciencia y su estudio, debe ser universal, financiado de forma pública o privada, pero no que su financiación recaiga en los científicos o per-

sonas interesadas en aprender alguna área. De esta forma, realizamos una recopilación de las opciones existentes a la fecha de comienzo de nuestro proyecto de visualización de los niveles de NO2 en la atmósfera medidos por el satélite Sentinel. Existen algunos minoritarios pero hay uno en especial que destaca por su mayoritario uso frente a los demás, el Sentinel visualizer.

http://www.fmsasg.com/Products/SentinelVisualizer/visualization.asp

Estos programas ofrecen la visualización de todo el conjunto de datos que transmite el satélite y te lo presenta o en ficheros o en gráficas para su análisis y estudio. Entre todos los datos que envía también se encuentran las concentraciones de NO2.

La página web, reseñada más arriba, junto con las demás existentes de menor difusión, guardan algo en común, todas ellas trabajan bajo una licencia de pago que se puede adquirir en sus respectivas plataformas. La problemática es que estas licencias tienen un coste muy alto, por lo que únicamente estarán al alcance de laboratorios, ministerios y profesionales, pero los interesados en el tema serán disuadidos, si su intención es realizar un pequeño estudio, pues el precio de acceso es exagerado.

Esta razón, unida a la antes mencionada, ha hecho surgir nuestra motivación por el trabajo, ya que nuestro interés final es crear esta plataforma para que interesados no profesionales en este tema puedan de forma gratuita visualizar los datos de contaminación enviados por Sentinel y así contribuir con nuestro pequeño granito de arena a hacer de este mundo un planeta menos contaminante y más respetuoso con el medio ambiente

1.2. Objetivos

El principal objetivo consiste en permitir al usuario el estudio de la contaminación atmosférica mediante la visualización en una página web de información sobre las nubes de NO2 existentes en la atmósfera, a partir de los datos que son captados gracias al satélite Sentinel. La forma de mostrar los datos, será sobre un mapa cartográfico mundial, representando sobre él, en forma de nubes de calor, las concentraciones de este gas objeto de estudio proporcionadas por el satélite.

Para comodidad del usuario a la hora de visualizar los datos mediante la página se dará la opción de seleccionar la fecha y lugar donde centrar el estudio para que la representación de los datos se centre en sus intereses.

Capítulo 2

Entornos de Desarrollo

Este capítulo introduce las herramientas que hemos empleado para la realización del proyecto que hemos desarrollado, incluyendo los frameworks y librerías además de los lenguajes de programación y la base de datos. Tras ello mencionaremos cómo realizamos el control de versiones durante el desarrollo.

2.1. Frameworks y Librerías

2.1.1. Front End

El requisito de la parte front end es seleccionar . a fecha y mostrar los datos en el mapa. Se puede realizar con jQuery, Ajax, enviando la petición con la fecha interesada, y el servidor devolverá los datos para esa fecha. Para ello se han utilizado las siguientes librerías

1. Leaflet [8]: Es una librería escrita en JavaScript de código abierto, cuya función es la representación de mapas cartográficos interactivos, con funcionalidades de representación de una malla de puntos en forma de mapa de calor. Lo emplearemos para la muestra sobre el mapa de las concentraciones atmosféricas de NO2.

2. jQuery [7]: Es una librería multiplataforma de JavaScript. Nos simplifica la manera de interactuar con los documentos HTML y los objetos DOM. Además con la técnica de Ajax, nos permite enviar peticiones de forma asíncrona.

3. Lightpick [9]: También una librería escrita en JavaScript de código abierto, cuya función es crear un selector de fecha. En internet hay numerosas librerías que tienen la misma funcionalidad. Hemos elegido esta porque es una librería ligera, tiene un buen diseño de gráfico, y su documentación es completa. Además hay pequeños demos que te enseñan cómo funciona los métodos, con lo que se reduce mucho el tiempo de aprendizaje.

4. Moment [11]: Es una librería ligera de JavaScript, para analizar, validar, manipular y dar formato a las fechas. Es una dependencia de la librería Lightpick.

2.1.2. Back End

El requisito de la parte back end es descargar los archivos que ofrece la página de Sentinel, sacar los datos que nos interesan, guardarlos en un fichero .csv, y subir este fichero al servidor.

1. netCDF4 [12]: netCDF es un formato de fichero abierto. Su uso principal es crear, almacenar y leer los datos científicos orientados a los conjuntos. netCDF4 es la cuarta versión de la librería que nos permite acceder a los ficheros de formato netCDF.
2. NumPy [14]: Es una librería matemática de Python. Tiene funciones matemáticas de alto nivel para operar con vectores y matrices. Es una librería dependiente de la librería netCDF4.
3. pandas [15]: Es una librería extensión de NumPy. Ofrece estructuras de datos y funciones para analizar y manipular tablas numéricas y series. Es una librería dependiente de netCDF4 también.
4. gitPython [6]: Es una librería de Python que nos facilita operar el repositorio con las funciones de git desde Python.

2.2. Lenguajes de Programación

En este proyecto aunaremos varios lenguajes de programación que nos proporcionarán la flexibilidad de crear nuestra aplicación en los diferentes medios disponibles.

Para la parte de visualización web del mapa emplearemos html en la creación de la pagina web y css en la creación de su estructura y formato.

En cuanto a la parte dinámica de la página web usaremos javascript para trabajar con la librería de Leaflet.

Hemos usado Python para descargar y procesar los datos de Sentinel, ya que es un lenguaje simple, potente y rápido. Ofrece numerosas librerías que nos pueden facilitar el desarrollo. Python también tiene una comunidad muy activa, eso significa que cuando tengamos algún problema, podremos encontrar soluciones fácilmente.

2.3. Base de Datos

Este proyecto trabajará con un repositorio incluido en el servidor que ejecuta la página web, de esta forma este repositorio almacenará los archivos puente con los datos extraídos de la información del satélite en el formato .csv.

En nuestro caso, no se empleará una base de datos SQL debido a que los datos como antes hemos mencionado, únicamente los almacenaremos en forma de fichero temporal puente, debido a que estos ficheros consisten en la parte filtrada del total proporcionado en la página web de la ESA [5], es decir, con la parte que nos interesa únicamente. Este archivo sirve para tener un acceso sencillo y filtrado a la información que nos atañe. En este caso, el NO2.

2.4. Control de Versiones

Para el control de versiones, empleamos la tecnología que nos ofrece Git. De esta forma, usamos su sistema de control de versiones denominado GitHub para la gestión de los sucesivos cambios que durante el desarrollo del TFG hemos ido realizando.

Capítulo 3

Diseño e Implementación

En este capítulo explicaremos cada una de las fases por la que el proyecto ha ido pasando, desde que se concibió la idea hasta que finalizó el desarrollo y fue testeado. Cada una de las fases serán explicadas y expuestas con detalle haciendo especial hincapié en la última fase, la de implementación, en la que se explicará cada sección del código final del software desarrollado.

3.1. Fase de análisis

El análisis de este proyecto consistió en dar forma a la idea de tener un lugar web donde, seleccionando el día de interés,se mostrase un mapa mundial, indicando mediante un mapa de color las concentraciones de contaminación para así estudiar y ver el impacto de la actividad humana sobre la atmósfera de nuestro planeta.

De esta forma, extraemos que los requisitos de este proyecto son los siguientes.

El programa consistirá en una página web soportada por un servidor que le proporcionará los datos para que se puedan representar en la página.

La aplicación solicitará al usuario una fecha en la que representar la información y de manera opcional un lugar geográfico donde centrar la visualización.

Tras esto, el servidor cargará la información solicitada a partir de un repositorio donde se encuentra preparada en formato y forma dicha información, dispuesta a ser cargada en el mapa de calor para su visualización.

A la hora de seleccionar el gas objeto de estudio de entre todos los gases contaminantes en la atmósfera, se ha acordado el estudio de las concentraciones de NO2.

3.2. Fase de investigación

La investigación inicial del proyecto, comenzó con la obtención de información sobre los diferentes mecanismos existentes de captación y registros de NO2. Existen muchas estaciones terrestres en las ciudades y lugares habitados, además de numerosos satélites científicos encargados de estas mediciones.

Nuestro proyecto tiene como interés mostrar las concentraciones sobre un mapa mundial, por lo que descartaremos estudiarlo a partir de estaciones terrestres, ya que en ese caso solo tendríamos datos de las regiones pobladas que hagan estudios de la calidad de su atmósfera. Por este motivo decidimos decantarnos por las misiones espaciales con satélites, centrándonos en una en especial: las misiones Sentinel de estudio de la superficie de nuestro planeta, realizadas por la Agencia Espacial Europea.

Sentinel es un conjunto de misiones. A fecha en la que nos encontramos existen cinco, todas dedicadas al estudio de diferentes ámbitos de la atmósfera y superficie terrestre con carácter científico. De todas estas, nos quedaremos con la última, la misión Sentinel-5, pues entre sus muchas tareas, se dedica a medir y estudiar las concentraciones de NO2 sobre el total de la superficie de nuestro planeta. Esta misión se encarga del estudio y la medición de la atmósfera en cuanto a elementos, composición y concentraciones de diversos gases. Además de esto, incluye imágenes y estudios diversos de multitud de aspectos sobre la superficie terrestre. Todos estos datos los aglutina la ESA y los pone a disposición de cualquier interesado gracias a su página web.

Por lo tanto, para comenzar con nuestro proyecto, nos centramos en la obtención de este conjunto de datos.

La ESA pone a disposición de los interesados un repositorio con todos los datos de las diferentes misiones, ordenados únicamente por fecha, por lo que necesitaremos crear un sistema automatizado que vaya descargando estos ficheros y tras ello comenzar con todo el tratamiento posterior que de lugar a la aplicación web con el mapa de calor sobre un plano que abarque todo el mundo.

3.3. Fase de Instalación y Configuración

En la parte relacionada con la muestra de datos, desarrollaremos una aplicación web, es decir, una web dinámica que emplee HTML5, CSS y JS. Lo primero que vamos a hacer es elegir el IDE (entorno de desarrollo integrado). La elección de una buena herramienta nos puede facilitar y ayudar mucho en el desarrollo. Tras la aparición de Internet, comenzó inmediatamente el desarrollo de las páginas web y debido a esta gran demanda surgieron muchísimos IDE para el desarrollo web, como por ejemplo, Visual Studio Code, Atom, Komodo Edit, Brackets, etc.

Después del estudio de todas las herramientas disponibles, valorando todos sus pros y contras, decidimos utilizar Visual Studio Code [18], debido a que es un editor ligero

y sencillo de utilizar, pero poderoso, y con gran cantidad de recursos y posibilidades. Además tiene un mercado de extensiones, el cual permite instalar herramientas adicionales que pueden aumentar la eficiencia y ahorrar tiempo de desarrollo.

Entre las muchas extensiones disponibles instalamos las dos siguientes por su capacidad de mejora de nuestro entorno de desarrollo: Debugger for Chrome[4] y Live Server[10]. (Figura 3.1)

- Debugger for Chrome, como su propio nombre indica, nos da la capacidad de depurar nuestros códigos de JavaScript en Chrome. Esta extensión nos dará una gran ventaja, pues la eliminación de errores, además de evitarlos en el desarrollo, nos permitirá que la versión en Chrome esté libre de fallos, muy importante pues la cuota de mercado es casi del 70 %, por lo que la mayoría de usuarios ejecutarán nuestro programa a través de este buscador web.
- Live Server nos ayuda y brinda la posibilidad de crear un servidor local para páginas estáticas y dinámicas. Además recarga la página en vivo cuando hacemos cualquier modificación de la página, permitiendo que cada modificación quede plasmada instantáneamente en la web para ir comprobando su correcta ampliación o modificación.

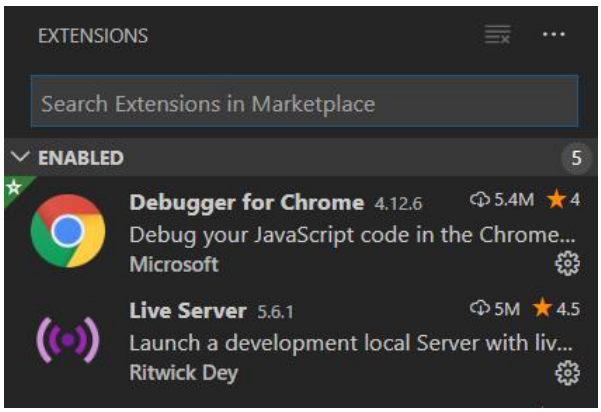

Figura 3.1: Extensiones de Visual Studio Code

Como hemos mencionado antes, utilizaremos el lenguaje de programación Python para descargar y procesar los datos de satélite a partir de la API que la ESA nos proporciona. Python es un lenguaje moderno con una comunidad activa por eso en el mercado hay varios IDE para Python, por ejemplo, PyCharm, Spyder, Thonny, Sublime Text, Eclipse+Pydev, etc. Después de estudiar un poco cada IDE, nos decantamos por la utilización de PyCharm[16] por sus múltiples funcionalidades y además por la experiencia que teníamos en desarrollar aplicaciones en Intellij. Ambas son desarrolladas por JetBrains y sus interfaces de usuario son parecidas. Esta ventaja nos permitiría tardar menos tiempo en aprender a utilizar ese entorno de desarrollo por su similaridad y familiaridad.

Después de instalar el entorno de desarrollo PyCharm, es necesario instalar también, para complementar el IDE, las librerías que hemos citado antes en la Sección 2.1. La librería NumPy y la librería pandas son dos librerías instaladas por defecto al instalar PyCharm.

PyCharm tiene un control de paquetes muy cómodo, en la pestaña File–Settings podemos ver todas las librerías que tenemos instaladas para consulta, actualización y para buscar cómodamente nuevas librería que podamos necesitar.(Figura 3.2)

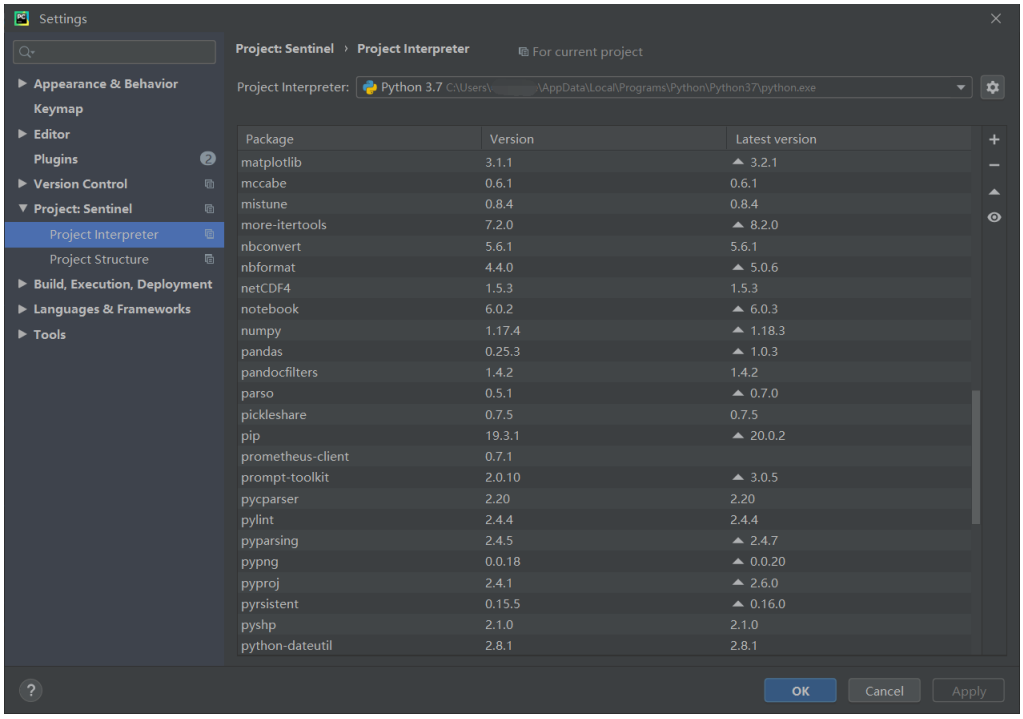

Figura 3.2: Control de paquetes de PyCharm

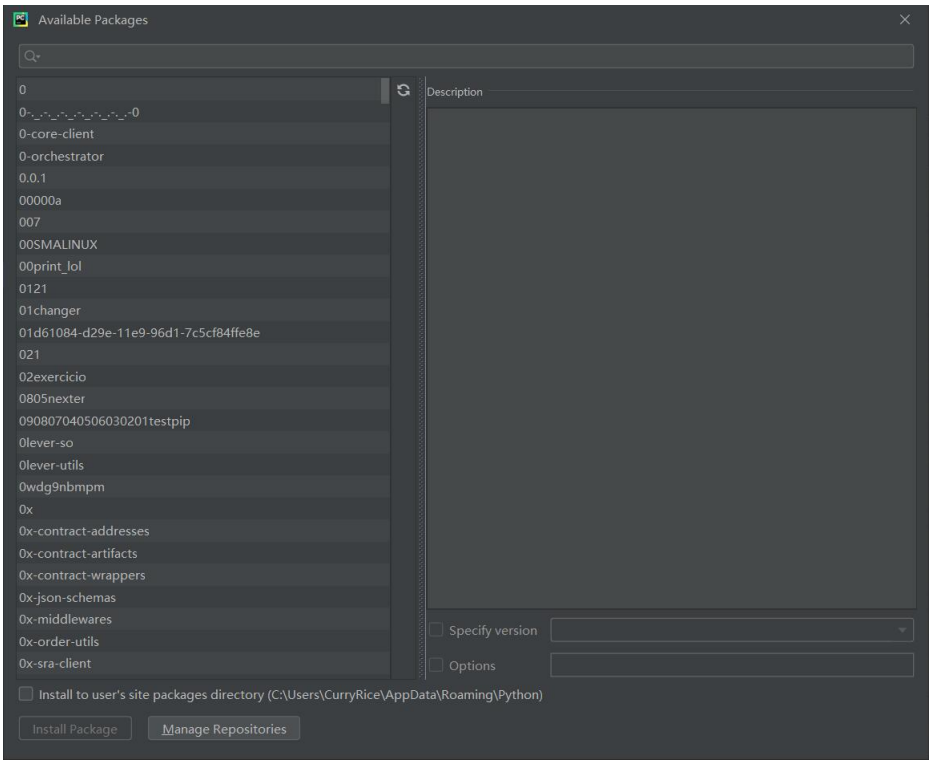

Figura 3.3: Añadir nuevas librerías

Para añadir nuevas librerías le damos al botón + que está en la parte derecha de la ventana y aparece una nueva ventana con las librerías disponibles. Estas librerías pertenecen al

repositorio oficial de Python (https://pypi.python.org/simple). Existe la posibilidad de añadir otros repositorios con el botón Manage Repositories, pero en nuestro caso no hace falta, las librerías que necesitamos están en el repositorio oficial. De entre todas buscamos e instalamos netCDF4 y GitPython.(Figura 3.3)

Con todo esto preparado, podremos empezar el desarrollo de una manera clara sencilla y eficiente.

3.4. Fase de Diseño

El diseño de esta aplicación consta de los siguientes elementos:

Obtención de los datos. Como hemos comentado en el apartado anterior, la ESA pone a nuestra disposición en su web los archivos de todas las misiones Sentinel para su descarga, por lo que lo primero que se comenzó a desarrollar es un programa en Python encargado de acceder al servidor y descargar el archivo que nosotros deseemos.

En este repositorio encontraremos archivos de las 5 misiones Sentinel, pero gracias a la investigación previa, tenemos la certeza que los datos sobre el NO2 que motivan este proyecto se encuentran en los datos que envía el Sentinel-5. Es esta misión la encargada de tomar las lecturas de gas por lo que nuestro programa se centrará en obtener únicamente estos ficheros.

Para filtrar por esta misión de Sentinel, el programa observará que el comienzo del nombre del archivo contenga S5P, esto denotará que pertenece a la dicha misión.

Una vez filtrado por esta nomenclatura, buscaremos los archivos que correspondan a la fecha deseada. La fecha aparece en el centro del nombre. La primera parte de este nombre de fichero es constante siempre, por lo que es sencillo aislar la parte que hace referencia a la fecha.

Un ejemplo de nombre fichero de la misión Sentinel-5 a fecha 17 de noviembre de 2019 es el siguiente:

S5P_NRTI_L2_CO__20191117T111514_20191117T112014_10854_01_010302_20191117T115316.nc

El programa una vez ha obtenido el nombre del archivo que nos interesa, lo descargará y almacenará para su posterior tratamiento.

Tratamiento de los ficheros de datos. El primer programa ha descargado y almacenado el archivo de un día concreto enviado por el satélite, pero el problema con el que nos enfrentamos ahora, es que el satélite no solo captura las lecturas de NO2 de la atmósfera, sino que en su archivo en formato .nc guarda una impresionante cantidad de variados datos captados por sus más de 15 instrumentos. Entre estos datos hay incluso fotografías en alta resolución de la orografía terrestre.

Toda esta gran cantidad de información hace que el archivo pese muchísimo, llegando incluso a ocupar gigas de memoria. Por lo tanto es un archivo inmanejable

por su tamaño y formato, además de la ineficiencia que supondría trabajar con tal cantidad de datos cuando únicamente nos interesa una pequeña parte.

Observado esto, se comenzara mediante otro programa a separar únicamente la parte que nos interesa que es la relativa al NO2 del conjunto total de información. Dentro del fichero hay una sección en la que consta en cabecera de la palabra NO2, por lo tanto recorremos el fichero hasta encontrarlo y tras ello iremos pasando esos datos a un fichero en formato csv. Este fichero es más manejable, y al tener únicamente los datos que deseamos, su tamaño en disco es muchísimo más pequeño.

El uso de archivos en formato csv El archivo generado tendrá la siguiente estructura. Cada línea en el archivo representará cada una de las lecturas realizadas por el satélite sobre la atmósfera, y que consiste en tres datos: Latitud,longitud y la concentración.

Tras diversas pruebas de rendimiento, observamos que la concentración enviada por el satélite utiliza una escala que hace que los datos tengan 6 o más decimales, por lo que para aumentar la eficiencia de este método, multiplicaremos la concentración por un millón según vamos tratando las líneas. De esta forma los datos en su mayoría corresponderán a números enteros más manejables para el sistema de representación de mapas de calor.

El segundo problema que observamos a la hora de la representación, es que entre el conjunto mayoritario de filas cuya concentración corresponde a un número natural coherente, algunas filas marcaban puntos con sus coordenadas, pero su concentración era tan ínfima en comparación con las demás, que las hacía prácticamente irrelevantes.

Estos datos aunque aparecen en baja proporción en el fichero, al disponer de más de 100.000 líneas por día, acaban siendo un conjunto de datos extenso que no solo no aporta ninguna información al usuario, sino que además ocupa espacio y confunde al mapa de calor mostrando puntos donde no debería haberlos. Por estos motivos decidimos poner una cota mínima de concentración, de forma que los puntos que no la superasen, no son almacenados en el fichero para evitar su posterior representación.

Una vez creado el fichero con todas las consideraciones antes mencionadas, ya disponemos de un archivo cuyo nombre es la fecha del día al que pertenecen los datos. Estos ficheros se almacenan en un repositorio a la espera de ser solicitado para la representación en la página web.

Creación de la interfaz de usuario. El objetivo de este proyecto es mostrar a un usuario interesado una interfaz que le permita apreciar las nubes de concentraciones de NO2, por lo que teniendo la parte de la obtención de la información y almacenamiento de dicha información, podremos proceder a crear la interfaz visual.

La interfaz visual la implementaremos sobre una dirección web, con el objetivo de llegar al mayor número de personas gracias a las posibilidades que ofrece internet.

La página web constará de los siguientes elementos:

- El mapa mundi sobre el que se representará el mapa de calor con las concentraciones. El mapa abarcará el planeta entero, ya que el satélite es de cobertura mundial y muestra información de cada rincón del planeta.

- Un selector de fecha. El satélite lleva 8 años transmitiendo información, por lo que será necesario que el usuario especifique la fecha que desea consultar. Esto nos ayudará a crear el programa de descargas con la capacidad de bajar el fichero de la ESA del día concreto que se ha solicitado.

La finalidad que tiene este sistema, es evitar almacenar en una base de datos extensa todos los archivos .csv con los datos de meses y meses. Así evitamos almacenar grandes volúmenes de datos, reduciéndose únicamente a los datos con las fechas solicitadas expresamente por el usuario.

Procesamiento de los .csv. Tras seleccionar la fecha por parte del usuario, comenzará el proceso de descarga y conversión en el fichero puente .csv. Todo este proceso es el descrito en los pasos anteriores.

Tras todo este proceso nos encontraremos con un .csv cuyo nombre es la fecha a la que corresponden sus datos, y será enviado al servidor para que lo muestre sobre el mapa. El programa irá recorriendo fila por fila el fichero puente y será ejecutado por una función en javascript que irá creando los puntos mediante la librería de Leaflet.

Cada uno de estos puntos con sus coordenadas y dato de concentración se irá representando sobre el mapa con la intensidad que corresponda a su lectura de concentración en ese punto de la atmósfera.

De esta forma tras haber seleccionado la fecha, el usuario observará un mapa de calor con toda la información de concentraciones de NO2 que el satélite Séntinel captó aquel día sobre la superficie terrestre.

3.5. Fase de implementación

3.5.1. Backend

Por razones de modularidad y mantenibilidad, dividimos el programa en siguientes ficheros de Python:

Nombre	Descripción
main.py	Programa principal
nc.py	Clase de NC donde se guarda información sobre fichero .nc
download.py	Descarga del fichero
read.py	Lectura de los datos que nos interesan y almacenamiento en un .csv
github.py	Envío de los datos al github
utils.py	Funciones útiles como por ejemplo, pasar el tamaño de String a Int
bot.py	Bot de telegram, para enviar mensajes al usuario

1. **main.py**

```
if len(sys.argv) > 1:
    try:
        date = datetime.datetime.strptime(sys.argv[1], '%Y/%m/%d')
    except ValueError:
        print("Date Format Error! (format: YYYY/MM/DD)")
        sys.exit()
else:
    date = datetime.datetime.today() - datetime.timedelta(1) # yesterday
```

Empezamos analizando la longitud del argumento. Si longitud es uno, significa que no hay argumento extra, entonces usaremos la fecha de ayer como la fecha que queremos descargar. Si la longitud es mayor que uno, pasaremos el String a tipo Date.

```
username = 's5pguest'
password = 's5pguest'
utils.get_cookies(username, password)
cookies = pickle.load(open("./cookies", 'rb'))
result = utils.get_files_by_date(date, cookies)

print("\n" + str(date.date()) + ": " + str(result['files'].__len__()) + "
    files with total size: " + utils.sizeof_fmt(result['total_size']))
```

Después, usaremos el nombre de usuario y la contraseña haciendo uso de las cookies de la página de Sentinel. Lo que hace get_cookies es, que con el nombre y la contraseña dada, almacena las cookies en un fichero local, y con la función pickle.load() carga las cookies como una variable de Pyhton. Esto lo empleamos así, ya que vamos a usar muchas veces las cookies durante la ejecución del programa. El siguiente paso, es hacer una solicitud de búsqueda con la fecha que nos interesa, lo que hace get_files_by_date() es que con la fecha dada, genera la solicitud, la envia, y procesa los resultados devolviendo una lista de arrays de la clase NC (esta función la vamos a explicar de forma más completa más adelante). Al final imprime un mensaje diciendo que en la fecha dada hay X ficheros con un tamaño total de T GiB.

```
file_path = "docs/data/" + str(date.date())
os.makedirs(file_path, exist_ok=True)
if os.path.exists(file_path + "/metadata.json"):
    with open(file_path + "/metadata.json") as f:
        try:
            data = json.load(f)
        except json.decoder.JSONDecodeError:
            data = {}
else:
    data = {}
```

Ahora necesitamos crear una carpeta cuyo nombre es la fecha de los datos. La función makedirs() nos puede funcionar, y exist_ok=True comprueba si ya existe la carpeta, si es así entonces no hace nada y devuelve True. Luego comprobará si existe metadata.json en dicha carpeta. Metadata.json es un fichero que guarda información sobre los fichero .nc que ya hemos descargado y procesado, y la información está estructurada en formato json. Entonces podremos cargar la función con json.load() de una manera muy sencilla. Si no existiese metadata.json significa que no hemos tenido información de ese día y se crearía un objeto vacío.

```
total_size = 0
files_to_download = []
for file in result['files']:
    if file.ncid not in data:
        files_to_download.append(file)
        total_size += file.size
print("Will download " + str(len(files_to_download)) + " files with " +
    utils.sizeof_fmt(total_size) + " (" + str(date.date()) + ")")
```

Lo que hace este bloque de código es recorrer la lista de ficheros .nc que hemos obtenido antes e ir comprobando si hay algún fichero que ya esté descargado y procesado. Si no es así lo añadiremos a la nueva lista files_to_download. Al final del procedimiento, se imprimirá un mensaje que nos dice cuantos ficheros tenemos que descargar y sus tamaños totales.

```
2020-05-03: 10 files with total size: 4.0GiB
Will download 10 files with 4.0GiB (2020-05-03)
```

Figura 3.4: Información sobre ficheros que va a descargar

```
MAX_RETRIES = 10
WAIT_SECONDS = 10
for file in files_to_download:
    for j in range(MAX_RETRIES):
        try:
            is_downloaded = download(file, cookies)
            break
        except requests.exceptions.ConnectionError:
            msg += "\n[" + datetime.datetime.now().strftime("%Y-%m-%d
                %H:%M:%S") + "](" + str(j) + " retry):<" + file.ncid +
                "> Connection Error."
        time.sleep(WAIT_SECONDS)
    else:
```

```
        is_downloaded = False
```

Descargar un fichero de internet es algo incontrolable, ya que podría estar afectado por muchísimos factores como, por ejemplo, inestabilidad de la red. Sería deseable que cuando algo falla en la fase de descarga, se vuelva a descargar de nuevo. En Python existe comando for-else que nos puede funcionar, ya que ejecuta el código de for primero y cuando ha terminado ejecuta el código de else. Si salimos de for con break entonces no ejecuta else. En este caso, cuando falla la descarga, capturamos la excepción, extendemos el mensaje de error, esperamos 10 segundos e intentamos de nuevo la descarga. Si después de 10 intentos sigue fallando, ponemos is_downloaded a falso, indicando que no podemos descargar este fichero. Si ha descargado con éxito con break salimos del bucle for.

```
if is_downloaded:
    i += 1
    print(" (" + str(i) + "/" + str(files_to_download.__len__()) + ")")
    read.save_to_csv("download/" + file.ncid + ".nc", file_path +
        "/data.csv")
    os.remove("download/" + file.ncid + ".nc")
    data.update({file.ncid: {"size": file.size}})
    metadata = open(file_path + "/metadata.json", "w")
    json.dump(data, metadata)
    metadata.close()
else:
    msg += "\n[" + datetime.datetime.now().strftime("%Y-%m-%d
        %H:%M:%S") + "](Download failed):" + file.ncid
```

Primero comprobamos que el fichero se ha descargado bien. Después con la función read.save_to_csv (que se explicará más adelante) sacamos los datos que nos interesan y se guarda en el fichero.

Con os.remove eliminamos el fichero .nc que hemos descargado, porque estos ficheros son enormes y para que no nos agote espacio de disco duro los eliminamos una vez que tenemos los datos necesarios.

```
try:
    github.push_to_github()
except Exception as e:
    msg += "\n[" + datetime.datetime.now().strftime("%Y-%m-%d
        %H:%M:%S") + "]" + str(e)

if not bot.send_message(msg):
    msg = "\n[" + datetime.datetime.now().strftime("%Y-%m-%d
        %H:%M:%S") + "]: Telegram bot can't send message." + msg

log = open("log.txt", "a+")
```

```
log.write("\n\n---" + date.strftime("%Y-%m-%d %H:%M:%S") +
    "---" + msg)
log.close()
```

Cuando ya tenemos los datos del día de interés, subimos a GitHub. Si algo falla añadimos el mensaje de error y lo enviamos por bot de telegram para que cuando no estamos delante del ordenador también podemos recibir mensaje desde móvil. Al final, guardamos el mensaje en un fichero log,txt por si algún fallo no lo hemos recibido como mensaje de telegram.

2. **nc.py**

```
class Nc(object):
    def __init__(self, title, ncid, link, size, date):
        self.title = title
        self.ncid = ncid
        self.link = link
        self.size = size
        self.date = date
    def __str__(self):
        return '\n' + self.title + '\n' + self.ncid + '\n' + self.link +
            '\n' + self.size + '\n' + str(self.date)
```

La clase Nc se encarga de guardar la información sobre el título, id, enlace de descarga, tamaño y fecha del fichero .nc. También cuenta con la sobrescritura del método __str__(self) para que cuando queremos imprimir las informaciones del fichero nc nos resulte más sencillo.

3. **download.py**

```
def download(nc, cookies):
    os.makedirs("download/", exist_ok=True)
    url = "https://s5phub.copernicus.eu/dhus/odata/v1/Products('" + nc.ncid
        + "')/$value"
    file_path = "download/" + nc.ncid + ".nc"

    r = requests.head(url, cookies=cookies, stream=True,
        headers={'Accept-Encoding': 'gzip', 'TE': 'gzip'})
    if 'content-length' in r.headers:
        total_size = int(r.headers['content-length'])
    else:
        total_size = nc.size

    if os.path.exists(file_path):
        temp_size = os.path.getsize(file_path)
    else:
```

```
15          temp_size = 0
```

Download asegura que existe la carpeta "download" y si no existe se crea. Luego se construye la url para hacer la descarga y la ruta donde se va a guardar el fichero. Después se hace un pedido de head sobre el fichero que queremos descargar. "Accept-Encoding" y "T" lo ponemos en gzip porque a veces el servidor nos devuelve "Transfer-Encoding: chunked" [17] para disminuir el tiempo de respuesta, pero este encoding no nos sirve porque no permite hacer pausa de la descarga, es decir, cuando perdemos la conexión con el servidor tenemos que descargar de nuevo, Los ficheros que tenemos que descargar normalmente son ficheros gigantes y es muy probable que perdamos la conexión durante la descarga. El siguiente paso es leer el tamaño del fichero, para poder indicar el porcentaje descargado. Pero a veces el header que nos devuelve no tiene atributo "content-length", entonces usamos el dato de la clase nc. El dato de la clase nc no es exacto, ya que está usando unidades como MiB y GiB, y se pierden las cifras pequeñas. Pero como solo lo usamos para mostrar el porcentaje de descarga, en general, no nos afecta mucho. Después comprobamos si existe el fichero que queremos descargar. Si existe leemos el tamaño del fichero y lo guardamos como tamaño temporal, sino el tamaño temporal sería cero.

```
1  headers = {'Range': 'bytes=%d-' % temp_size}
2  r = requests.get(url, stream=True, cookies=cookies, headers=headers,
       timeout=10)
3  if temp_size == 0:
4      f = open(file_path, "wb")
5  else:
6      f = open(file_path, "ab")
7
8  for chunk in r.iter_content(chunk_size=1024 * 1024):
9      if chunk:
10         f.seek(temp_size)
11         f.truncate()
12         f.write(chunk)
13         f.flush()
14         temp_size += len(chunk)
15
16         # progress bar
17         done = int(50 * temp_size / total_size)
18         sys.stdout.write("\r" + nc.ncid + ".nc(" +
               utils.sizeof_fmt(nc.size) + ")\t [%s%s] %d%%" % ('X' * done,
               ' ' * (50 - done), 100 * temp_size / total_size))
19         sys.stdout.flush()
20
21 f.close()
22 if utils.sizeof_fmt(temp_size) == utils.sizeof_fmt(total_size):
23     return True
24 else:
```

```
25      return False
```

Para empezar a descargar, primero vamos a construir el header de la petición. En el atributo “Range” ponemos el rango que queremos que nos devuelva, desde tamaño temporal hasta final del fichero. Con este header hacemos la petición y abrimos el fichero en modo añadir o sobrescribir, dependiendo si tenemos o no algo descargado anteriormente. Luego recibimos el fichero por chunk, el tamaño de cada chunk es 1024 * 1024 byte, es decir, 1 MiB. Este número no puede ser muy pequeño o muy grande. Si es muy pequeño, perdemos tiempo en seek() y flush() en masivos chunks pequeños, y si este número es muy grande perdemos tiempo en cuando perder los paquetes de datos y tenemos que solicitar de nuevo. Tras unas pruebas con diferentes números, 1 MiB es un número razonable. Después para cada chunk recibido, movemos el puntero de fichero a tamaño temporal, eliminamos los bytes después del tamaño temporal para evitar problemas, escribimos los datos de chunk, limpiamos el buffer de output y tamaño temporal sumarle el tamaño de chunk. Después imprimimos una barra de proceso y porcentaje de descarga. Repetimos todo este proceso hasta que recibimos el ultimo chunk, cerramos el fichero y comprobamos que el tamaño temporal es igual a tamaño total, para saber si este fichero se ha descargado bien.

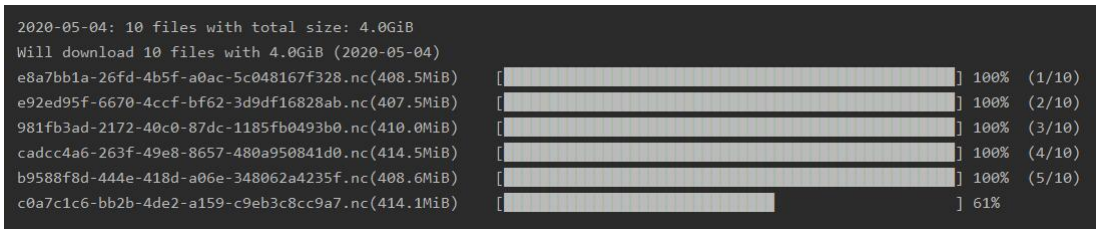

Figura 3.5: Información sobre la descarga

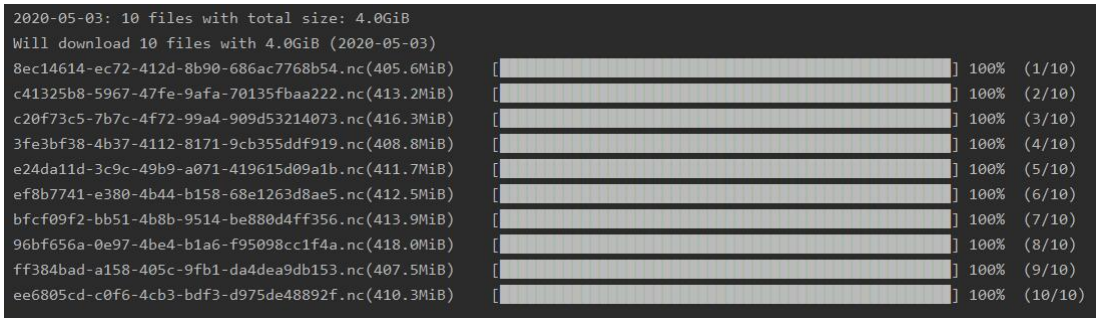

Figura 3.6: Información sobre la descarga terminado

4. **read.py**

```
1 def save_to_csv(file, save_path):
2     nc = Dataset(file, mode='r')
3     lons = nc.groups['PRODUCT'].variables['longitude'][:].flatten()
```

```
    lats = nc.groups['PRODUCT'].variables['latitude'][:][0, :,
        :].flatten()
    no2 = nc.groups['PRODUCT'].variables[
        'nitrogendioxide_tropospheric_column'][0, :, :].flatten()

    data = pandas.DataFrame({'Longitude': lons, 'Latitude': lats, 'Data':
        no2})
    data['Data'].replace('', numpy.nan, inplace=True)
    data.dropna(subset=['Data'], inplace=True)
    data["Data"] = data["Data"] * 1000000
    data = data[data.Data > 50]
    data.to_csv(save_path, index=False, mode='a', header=False)
    nc.close()
```

Lo que hace esta función, básicamente, es leer los datos del fichero nc, procesarlos y guardarlos en un fichero csv. Primero usamos Dataset de la librería netCDF4 en modo lectura para obtener una variable del tipo Dataset. Después en su grupo "PRODUCT" leemos las variables longitud, latitud y dióxido de nitrógeno. Guardamos estos datos en un DataFrame de la librería pandas, que es muy útil para gestionar los datos en forma de tabla. Tras esto descartamos las filas con datos incompletos, y lo multiplicamos por 1000000, porque los valores de dióxido de nitrógeno son números muy pequeños y no se diferenciarían en el mapa. Seguidamente descartamos las filas menores que 50. Este paso es necesario porque la mayoría de los datos son menores que 50, y eso aumenta muchísimo el tamaño del fichero y el tiempo de procesamiento para mostrar el mapa. Al final guardamos los datos que tenemos en un fichero csv y cerramos el fichero nc.

5. **github.py**

```
def push_to_github():
    git_ssh_identity_file = os.path.expanduser('~/.ssh/id_rsa')
    git_ssh_cmd = 'ssh -i %s' % git_ssh_identity_file
    with Git().custom_environment(GIT_SSH_COMMAND=git_ssh_cmd):
        repo = Repo(".")
        git = repo.git
        remote = repo.remote()
        remote.pull()

        git.add("docs")
        git.commit("-m", "update by Python script " +
            datetime.datetime.now().strftime(" %Y-%m-%d %H:%M:%S"))

        remote.push()
```

GitHub nos permite dos formas de autenticar, una es usar el nombre de usuario y contraseña y otra es la que usamos en el proyecto, conectar con GitHub mediante SSH keys [3]. Hacer esto, nos puede evitar que cada vez que se hace push haya

que introducir nombre de usuario y contraseña. En git bash ejecutamos el siguiente comando para generar SSH key:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

> Generating public/private rsa key pair.
> Enter a file in which to save the key (/home/you/.ssh/id_rsa):
> Enter passphrase (empty for no passphrase):
> Enter same passphrase again:
```

Passphrase lo tenemos que dejar en vacío, porque sino nos la va a pedir cada vez que hacemos push. Una vez tenemos SSH key, pegamos el contenido de id_rsa.pub a github [1]. Cuando SSH key funciona correctamente, ya podemos hacer push usando la librería gitPyhton. Primero indicamos donde está el fichero de key private, después obtenemos el objeto del repositorio, hacemos pull para asegurarnos que todo esta sincronizado, y añadimos todos los cambios de la carpeta "docs" que es la raíz de la nuestra página. Al final, hacemos el commit y push.

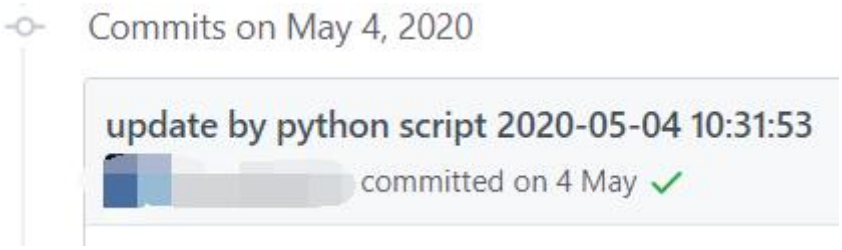

Figura 3.7: Automatización de push

6. **utils.py**

```
units = {"B": 1, "KB": 10 ** 3, "MB": 10 ** 6, "GB": 10 ** 9, "TB":
    10 ** 12}
# file size string to int
def parse_size(size):
    number, unit = [string.strip() for string in size.split()]
    return int(float(number) * units[unit])

# file size int to string
def sizeof_fmt(num, suffix='B'):
    for unit in ['', 'Ki', 'Mi', 'Gi', 'Ti', 'Pi', 'Ei', 'Zi']:
        if abs(num) < 1024.0:
            return "%3.1f%s%s" % (num, unit, suffix)
        num /= 1024.0
    return "%.1f%s%s" % (num, 'Yi', suffix)
```

Estas dos funciones son muy útiles, la primera sirve para pasar el tamaño de una cadena de carácter a números enteros, y la otra para lo inverso. Hemos usado dos

listas de unidades porque el servidor de Sentinel nos devuelve en megabyte, gigabyte, y nosotros como estudiantes de informática, queremos usar mebibyte, gibibyte.

```
def get_cookies(username, password):
    login = requests.post("https://s5phub.copernicus.eu/dhus/login",data= {
        'login_username': username, 'login_password': password})
    cookies = {'dhusAuth': login.cookies['dhusAuth'], 'dhusIntegrity':
        login.cookies['dhusIntegrity']}
    f = open('./cookies', 'wb')
    pickle.dump(cookies, f)
    f.close()
```

Lo que hace la función get_cookies es hacer un pedido de login con el nombre de usuario y la contraseña que nos ha pasado por parámetro, y leemos dos atributos de cookies que nos han devuelto "dhusAuth" y "dhusIntegrity". El servidor de Sentinel nos identifica con estos dos campos de cookie. Con la librería pickle podemos guardar una variable en un fichero, para luego cuando quereamos usarla en múltiples ejecuciones, solo tenemos que cargar esta variable desde el fichero y no hace falta repetir todo el proceso anterior.

```
def get_search_result(date, cookies):
    date = str(date.date())
    url = 'https://s5phub.copernicus.eu/dhus/search?start=0&rows=100&q=(
        footprint:"Intersects(POLYGON((-29.812190777585087
        26.577078786569615,69.10491090874537
        26.577078786569615,69.10491090874537
        71.10236152833656,-29.812190777585087
        71.10236152833656,-29.812190777585087 26.577078786569615)))" ) AND
        ( (platformname:Sentinel-5 AND producttype:L2__NO2___)) AND (
        beginPosition:[' + date + 'T00:00:00.000Z TO ' + date +
        'T23:59:59.999Z] AND endPosition:[' + date + 'T00:00:00.000Z TO ' +
        date + 'T23:59:59.999Z] )'

    text = requests.get(url, cookies=cookies)
    tree = ET.ElementTree(ET.fromstring(text.text))
    root = tree.getroot()
    entries = root.findall("{http://www.w3.org/2005/Atom}entry")
    return entries
```

La función get_search_result()hace una petición de búsqueda con la fecha dada. Primero pasamos la fecha del tipo date al tipo string, y construimos una query. Lo que pide esta query es la información del polígono que hemos definido de la plataforma"Sentinel-5", con tipo de producto "L2__NO2___" y la fecha seleccionada. El resultado lo pasamos a una estructura de árbol de XML, y buscamos el nodo de las entidades y lo devolvemos.

```
def get_files_by_date(search_date, cookies):
    result = {'files': [], 'total_size': 0}

    entries = get_search_result(search_date, cookies)
    for entry in entries:
        date = entry.find('{http://www.w3.org/2005/Atom}date
            [@name="beginposition"]').text.split("T")[0]
        date = datetime.datetime.strptime(date, '%Y-%m-%d')
        title = entry.find('{http://www.w3.org/2005/Atom}title').text
        ncid = entry.find('{http://www.w3.org/2005/Atom}id').text
        link =
            entry.find('{http://www.w3.org/2005/Atom}link').attrib['href']
        size = entry.find('{http://www.w3.org/2005/Atom}str
            [@name="size"]').text
        size = parse_size(size)

        nc = Nc(title, ncid, link, size, date)
        result['files'].append(nc)
        result['total_size'] += size

    return result
```

Para cada entidad que obtenemos con la función anterior, sacamos fecha, nombre, id, enlace de descarga y tamaño. Lo guardamos en una lista de clase nc y la retornamos.

7. **bot.py**

```
token = 'XXX'
my_id = "XXX"
def send_message(message, chat_id=my_id):
    url = "https://api.telegram.org/bot" + token + "/sendMessage?chat_id="
        + chat_id + "&text=" + message
    response = requests.get(url)
    if not response.ok:
        return False
    else:
        return True
```

El objetivo de esta función es mandar un mensaje a nuestro telegram, usando telegram bot [2]. Así podemos ejecutar la aplicación en un servidor con crontab, especificar una hora de ejecución para todos los días y recibir resultados de ejecución desde móvil sin necesidad de estar frente al ordenador.

Figura 3.8: Mensaje recibido desde la aplicación Telegram

3.5.2. Frontend

En la parte de demostración utilizamos GitHub Pages, ya que es un servicio de GitHub gratis y es suficiente para nuestra necesidad. Para el mapa hemos usado "Leaflet", una librería de mapa escrita en javascript y de código abierto. La estructura de nuestra web tiene la siguiente forma:

```
|   index.html
|
+---css
|       index.css
|       lightpick.css
|
+---data
|   |   metadata.json
|   |
|   +---2020-03-01
|   |       data.csv
|   |       metadata.json
|   |
|   +---2020-03-02
|   |       data.csv
|   |       metadata.json
|   |
|   +---2020-03-03
|   |       data.csv
|   |       metadata.json
|   |
|   ...
|
+---img
```

```
|       calendar.png
|
\---js
        leaflet-heat.js
        lightpick.js
        moment.min.js
```

index.html es la página principal, la carpeta "data" contiene los datos que hemos descargado y procesado, la carpeta "img" es la carpeta donde se guardan las imágenes que se utilizan en la página. Finalmente todas las librerías de javascript las guardamos en la carpeta "js".

```
$.getJSON("data/metadata.json", function (metadata) {
            let disableDates = [];
            let start = new Date(metadata[0]);
            let end = new Date(metadata[metadata.length - 1]);
            let temp = start;
            while ((end - temp) > 0) {
                if (!metadata.some(d => {
                    d = new Date(d);
                    if (d.getDate() != temp.getDate())
                        return false;
                    if (d.getMonth() != temp.getMonth())
                        return false;
                    if (d.getFullYear() != temp.getFullYear())
                        return false;
                    return true;
                })) {
                    disableDates.push(temp);
                }
                let nextDay = new Date(temp);
                nextDay.setDate(temp.getDate() + 1);
                temp = nextDay;
            }
```

En el código fuente de la página principal podemos encontrar este fragmento de código. Cuando la página está cargada, con jQuery hacemos una petición de "metadata.json" donde está la información de los días con información disponibles. Pero tenemos un problema porque "lightpick.js", la librería de calendario, solo nos permite inhabilitar días. Entonces tenemos que crear una lista de días inhabilitados y recorrer las fechas, metiendo los días que no están en la lista de metadata.

```
            picker = new Lightpick({
```

```
            field: document.getElementById('calendar'),
            format: "YYYY-MM-DD",
            onSelect: function () {
                $("p").first().text(picker.toString('YYYY-MM-DD'))
                getMetadata("data/" + picker.toString('YYYY-MM-DD'))
            },
            minDate: start,
            maxDate: end,
            disableDates: disableDates
        });
    });
```

Una vez tenemos la lista de días inhabilitados, ya podemos crear y configurar el selector de fechas. Para ello indicamos el ID del elemento DOM, el formato de las fechas, callback de selección, fecha inicio, fecha final y la lista de días inhabilitados.

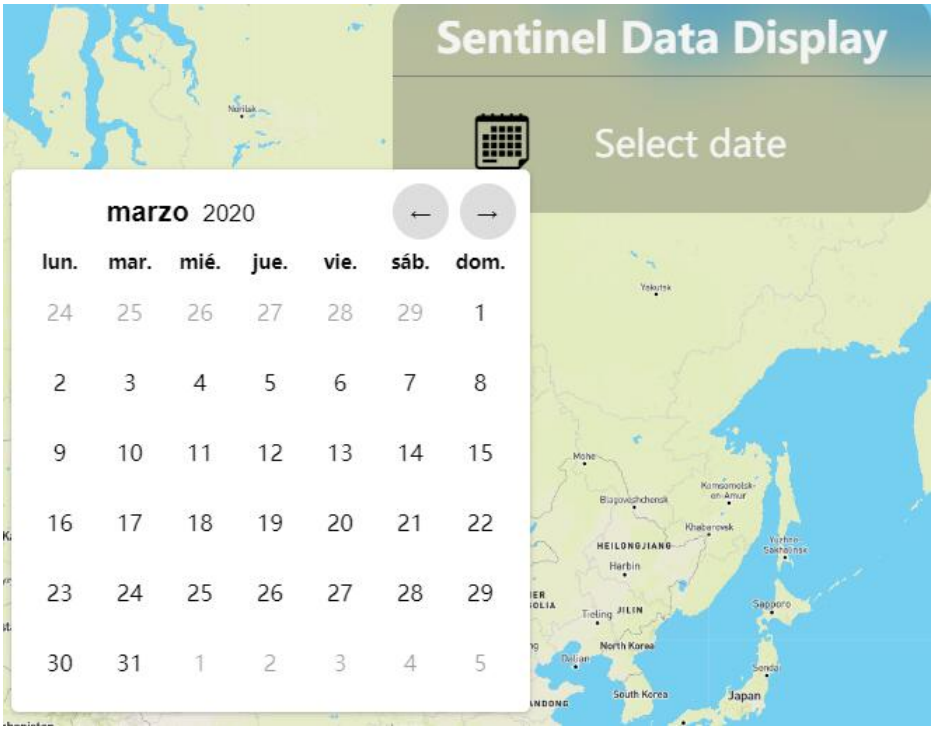

Figura 3.9: Selector de fechas

```
function getMetadata(path) {
    $.ajax({
        type: "GET",
        url: path + "/data.csv",
        dataType: "text",
        success: function (data) {
            processData(data);
```

```
        }
    });
    if (mymap.hasLayer(marker)) {
        mymap.removeLayer(marker)
    }
}
```

Esta función es callback de cuando seleccionamos una fecha, el parámetro que nos pasa es la fecha del tipo string, con esto hacemos una petición de los datos, y llamamos a la función "processData" cuando el pedido se haya realizado con éxito. Al final, eliminamos todos los elementos del mapa si estaba pintado anteriormente.

```
function processData(allText) {
    var allTextLines = allText.split(/\r\n|\n/);
    var tablaPuntos = [];

    for (var i = 0; i < allTextLines.length - 1; i++) {
        var data = allTextLines[i].split(',');
        var ptocalor = [data[1], data[0], data[2]]
        tablaPuntos.push(ptocalor)
    }
    marker = L.heatLayer(tablaPuntos, {
        scaleRadius: true,
        radius: 25,
        minOpacity: 0.2,
        blur: 50
    });
    marker.addTo(mymap)
}
```

El texto que nos pasan por parámetro lo dividimos con saltos de linea, y para cada linea lo dividimos a su vez con comas. Así tenemos longitud, latitud y valor de dióxido de nitrógeno. Esta información la metemos en una lista. Después creamos un Layer, le pasamos la lista y la configuración y añadimos este Layer al mapa que tenemos.

Todo el código se puede consultar en `https://github.com/CurryRice233/Sentinel`

La aplicación puede ejecutarse en `https://curryrice233.github.io/Sentinel/`

Capítulo 4

Caso de estudio

Una imagen vale más que mil palabras, en este apartado expondremos y explicaremos dos ejemplos ilustrativos para explicar y mostrar la utilidad y funcionamiento del proyecto desarrollado.

El año 2020 es un año muy diferenciado de todos los años recientes que preceden. El mundo por desgracia se ha visto envuelto este año en una grave crisis humanitaria por culpa de un virus surgido a principio de año denominado covid-19.

Este virus, para sorpresa de toda la humanidad, pasó de pequeños focos de contagio en la provincia china de Wuhan a, en apenas dos meses, convertirse en una grave pandemia global con muy graves consecuencias. Además de la crisis humanitaria y sanitaria, a la que ha sometido al planeta, se ha producido una crisis económica y un parón de la vida cotidiana tal como la conocíamos, como no ha habido precedente en la historia más reciente.

Por otra parte, deseamos resaltar en este proyecto, al tratarse de una aplicación de medición de la contaminación atmosférica global, que se ha producido un escenario idóneo para la aplicación y observación del funcionamiento del proyecto que hemos realizado, ya que nos proporciona datos de como las medidas que los países han ido tomando se han visto reflejados en la generación de gases contaminantes.

Europa es el continente que más ha sufrido por esta pandemia, y los gobiernos para evitar la transmisión ha decretado confinamientos como nunca antes se habían visto. Estos confinamientos han reducido drásticamente la emisión de gases contaminantes, tanto el generado en desplazamientos al verse restringidos, como en la actividad industrial y de transporte por el cierre de empresas y negocios no esenciales, con todo el movimiento de mercancías que ocasionaban.

Sobre este escenario vamos a realizar un estudio de la evolución de las nubes de gas contaminantes durante el mes de marzo. Este mes ha sido seleccionado porque es cuando los países europeos comenzaron a decretar sus respectivas medidas.

Este estudio se centrará en dos países europeos, España e Italia, por su cercanía y conocimiento de las medidas que se han ido aplicando. El objetivo de este estudio consiste en tomar diferentes días para ir viendo la reducción de contaminación que se encontraba en la atmósfera a medida que las normas de confinamiento se iban endureciendo.

4.1. Análisis de Italia

El primer caso de estudio lo realizaremos sobre Italia, ya que presenta una peculiaridad respecto a los demás países europeos, incluido el nuestro. El confinamiento se decretó en dos fases. Primero la parte de Lombardía y Venecia, pues el brote comenzó allí, y al extenderse más tarde ya se declaró el decreto de confinamiento para el país completo. En este caso, nos centramos en el norte de Italia, ya que la mayoría de las industrias del país están situadas allí, y podemos ver claramente el cambio de contaminación en la zona antes y después del confinamiento.

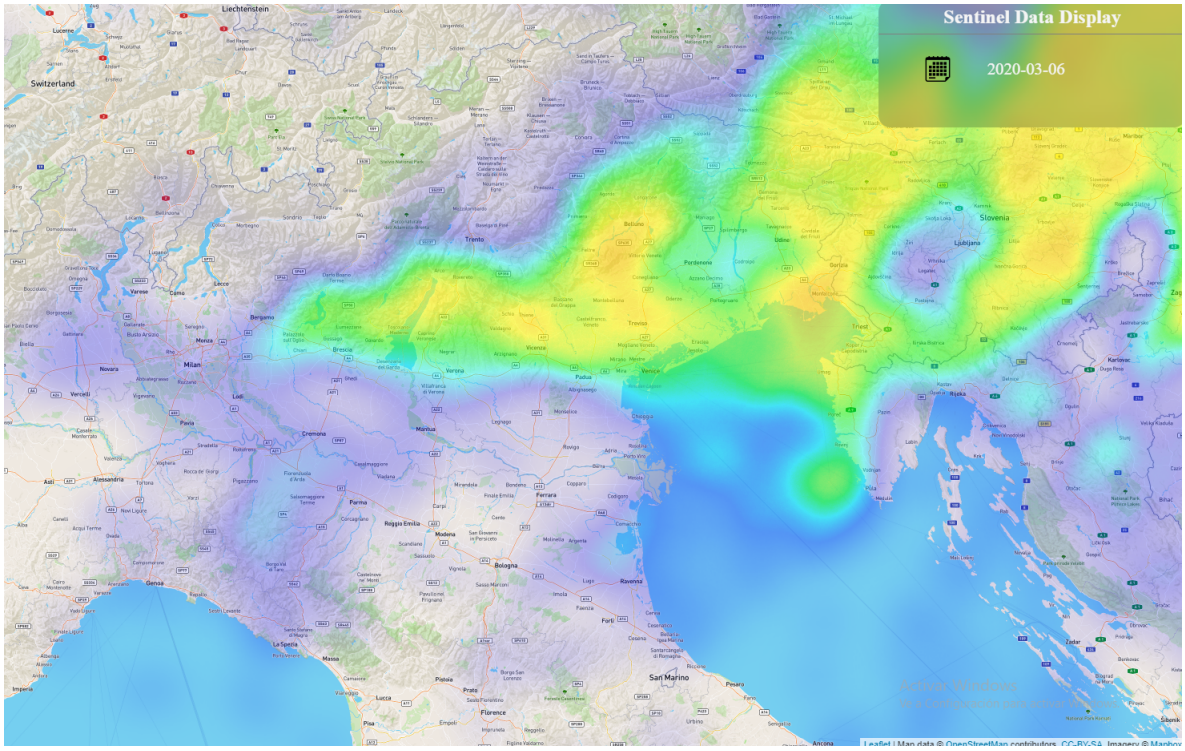

Figura 4.1: Demostración en Italia del día 6/03/2020

En la primera imagen (Figura 4.1) observamos las altas concentraciones de contaminación, ya que en esta fecha aún no se había decretado ningún confinamiento. Esta es la situación en la que se encontraba Europa antes de la llegada de la pandemia.

En la segunda imagen (Figura 4.2), observamos el impacto de las primeras medidas tomadas en el país italiano, por lo que en la zona de la Lombardía y Venecia ya ha quedado la contaminación muy reducida.

Por último (Figura 4.3), tras el descontrol de la pandemia, el país entero quedó confinado,

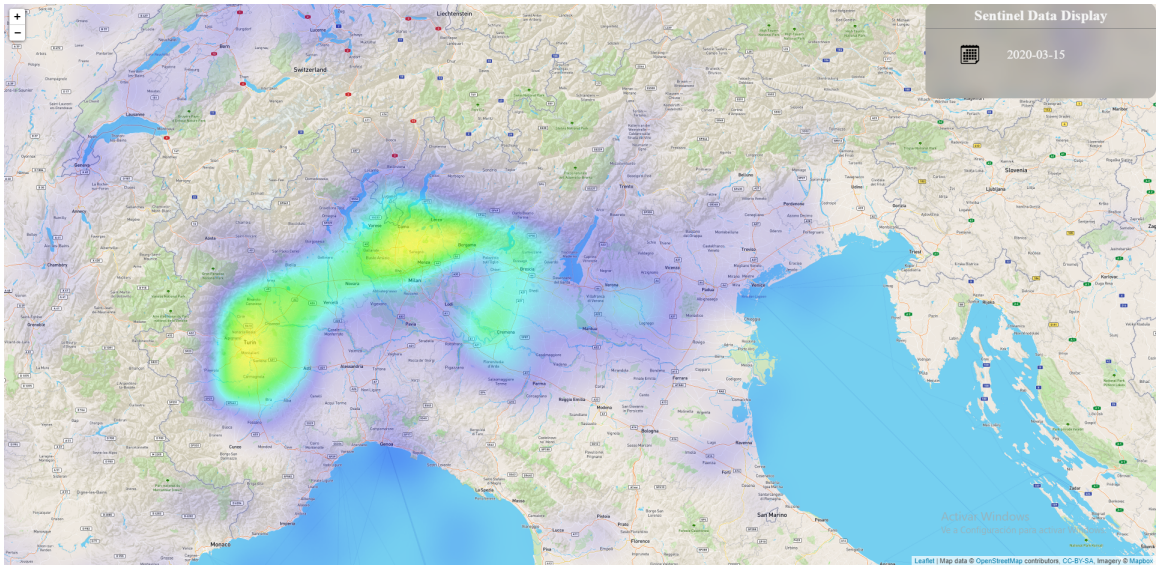

Figura 4.2: Demostración en Italia del día 15/03/2020

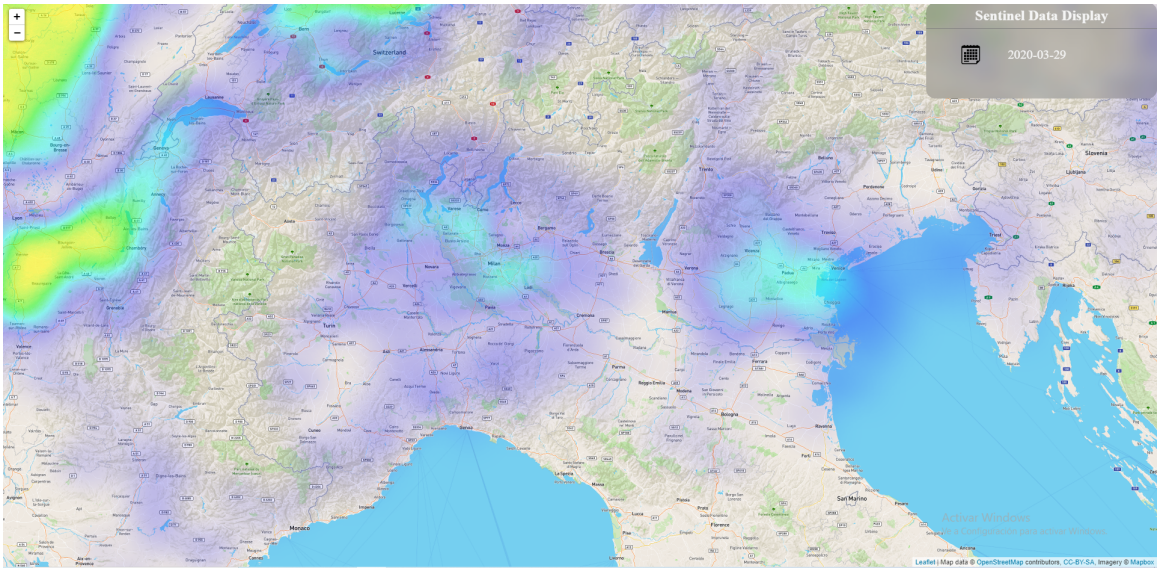

Figura 4.3: Demostración en Italia del día 29/03/2020

y la actividad reducida a únicamente actividades esenciales. Aquí podemos observar el impacto de las medidas en la amplia reducción de todo tipo de contaminación atmosférica.

La peculiaridad del país italiano en el caso de estudio, consiste en que el confinamiento, a diferencia de España, no fue decretadode forma homogénea sino que se dividió en dos fases, la zona norte del país y más tarde se extendió al resto de territorios.

4.2. Análisis España

En este segundo ejemplo observaremos la repercusión de las medidas adoptadas en nuestro país antes y después de la pandemia.

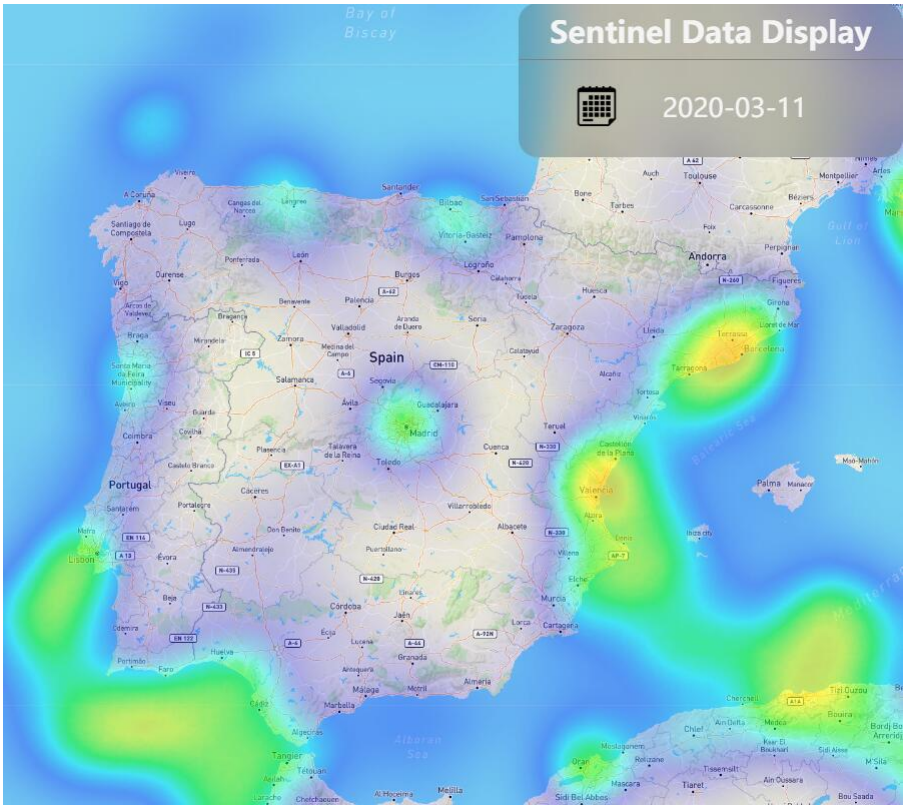

Figura 4.4: Demostración del día 11/03/2020

La contaminación en España y Portugal era muy amplia y centrada en los focos industriales principalmente(Figura 4.4).

Tras el confinamiento, la reducción de la contaminación es inmensa, quedando únicamente pequeños focos en las grandes ciudades (Figura 4.5).

Tras el estudio, podemos concluir, que estamos ante una situación inédita. Jamás nos habíamos ni acercado a los niveles actuales de bajas emisiones, pues la reducción de la actividad económica alcanza una cota sin precedentes. Confiamos que este respiro al planeta, sirva para que al retomar la actividad comiencen a implantarse las medidas anticontaminación de la que tantos años se lleva hablando, pero sin que nadie las aplique firmemente.

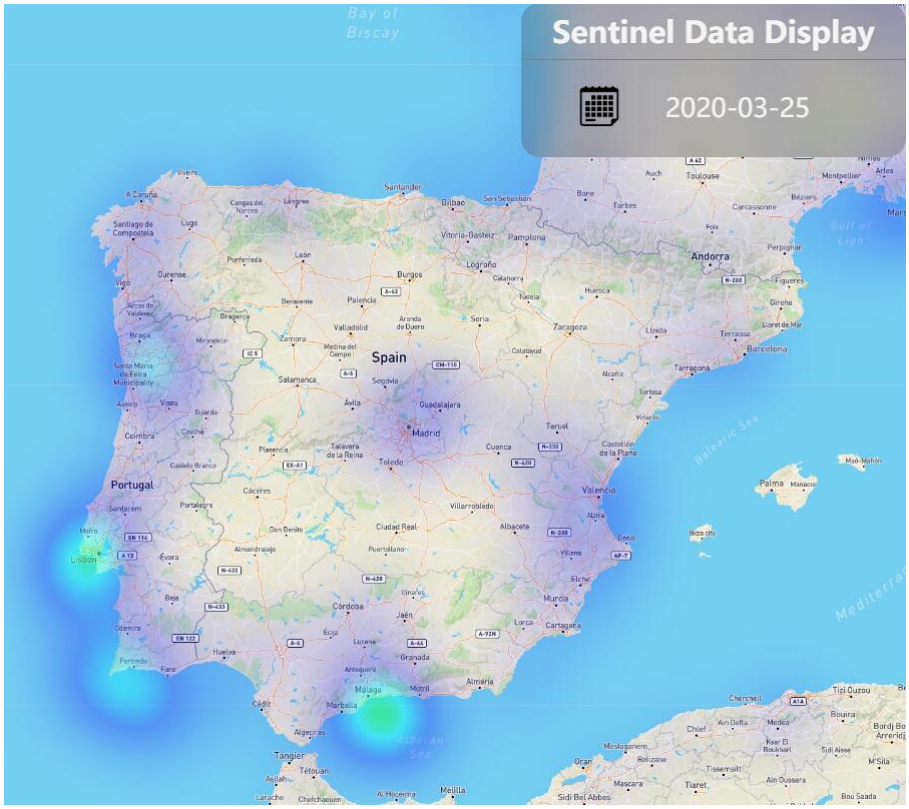

Figura 4.5: Demostración del día 25/03/2020

Capítulo 5

Contribuciones al Proyecto

En este capítulo expondremos las contribuciones de cada uno nosotros, los integrantes del proyecto, para la obtención final del proyecto. Cada uno se ha especializado en una sección para que a la hora de integrarlo el resultado sea un software de alta calidad.

5.1. Elam Uceda Herrero

Mi contribución al proyecto, abarca los siguientes puntos:

1. La fase de análisis de los requisitos junto a mi compañero, ha sido la fase inicial que comenzamos a realizar, y de forma conjunta fuimos estudiando cada requisito solicitado y estudiando los diferentes métodos que tenemos a nuestra disposición para resolverlos.

   Concretamos en esta fase, que el lenguaje de programación sería el Python debido a su forma simple y rápida de desarrollar código unido con su flexibilidad con los detalles y su portabilidad a la hora de ejecutarlo en diferentes sistemas operativos. Esta cualidad es importante pues los usuarios que deseen usar nuestra aplicación no se encontrarán con estas restricciones.

   En cuanto a la página web la desarrollaríamos en HTM5 por nuestro conocimiento del lenguaje, su claridad y comodidad a la hora de desarrollar páginas web junto con su estilo con el lenguaje css.

2. En la fase de investigación estudié las diferentes misiones Sentinel existentes, buscando la más apropiada para nuestro trabajo. La misión Séntinel-5 coincidió con lo que buscábamos, ya que se encarga de medir las lecturas de NO2 en la atmósfera.

   Con esta información pudimos buscar en la página de la ESA el repositorio para empezar con la obtención de datos.

Además de estas investigaciones me dispuse a buscar las alternativas ya existentes para obtener la información que íbamos a generar. Y así recopilé información de las distintas plataformas de pago que existen para acceder a los datos del Séntinel.

3. En la fase de diseño, mi contribución se centra en la representación del mapa de calor para la visualización del NO2.

   Para empezar busqué información sobre diferentes librerías compatibles con javascript para incluir el mapa del mundo y que me permitiese crear puntos de calor para representar cada coordenada enviada por el Satélite.

   Me decanté por Leaflet, ya que es gratuita a diferencia de Google Maps y por su comodidad y simplicidad a la hora de ir leyendo del csv y generando por cada fila un punto con su concentración.

Por lo tanto, se ha centrado mi aportación principal en la sección del mapa mundi, y la representación de todos los puntos obtenidos del satélite.

5.2. Jiali Zheng

Mi contribución al proyecto, abarca los siguientes puntos:

Como ha dicho mi compañero, en la fase de análisis de los requisitos, trabajábamos de forma conjunta a la hora de estudiar los requisitos, y después del estudio, decidimos utilizar Python para descargar datos y HTML + CSS + JavaScript para la parte de demostración.

En la fase de investigación, con la información de que los datos que nos interesa están en la misión Séntinel-5, me puse a investigar el formato de fichero que nos ofrece. Observando la estructura del fichero y cómo sacar los datos que nos interesa. Después de esta investigación añadimos un proceso más para obtener los datos solo que nos interesa y descartando informaciones de baja utilidad. De esta forma, redujimos el tamaño de los ficheros para que ocuparan menos memoria y fueran más manejables.

Con la fase anterior, sabemos que el formato de los ficheros que nos ofrece ESA es netCDF4. En esta fase, la fase de instalación y configuración, estudié la librería netCDF4, para saber cómo instalar en PyCharm y cómo leer los datos en Python. Otro problema al que me enfrenté, fue automatizar el git, subir los ficheros que hemos procesado al GitHub. Después de probar varias formas y varias librerías, la mejor solución que discerní fue usar la librería GitPython con la forma de login ssh.

En la fase de implementación, mi contribución se centra en desarrollo el programa de Python para descargar y procesar los datos de Sentinel y guardarlos en fichero de .csv.

Capítulo 6

Trabajo Futuro

Este capítulo se centra en explicar todo el trabajo que podría desarrollarse a partir de nuestro proyecto con el fin de extender su funcionamiento, usándolo como base para la realización de proyectos mucho más complejos

Este proyecto, como hemos ido contando, ha tenido un objetivo muy específico, la obtención de los datos de la contaminación de la atmósfera a partir del satélite Séntinel. No obstante este satélite ofrece una increíble cantidad de datos captada por todos sus sensores, por lo que el trabajo en el futuro se dividiría en dos ramas.

La primera sería la ampliación del trabajo existente en la búsqueda de concentraciones de NO2 y la segunda sería la obtención y análisis de otros archivos enviados por el satélite.

6.1. Primera rama

La primera posibilidad para realizar el trabajo en el futuro consiste en la realización de múltiples mejoras a la aplicación ya existente, lo cual daría mayor versatilidad a la herramienta y haría que el usuario pueda obtener información más analizada y eficaz para sus estudios.

1. La primera de las posibles mejoras consistiría en alojar la web en un servidor propio más sofisticado que permitiese mucha más flexibilidad a la página web.

   Con un servidor así se podrá tener una base de datos que primeramente permitiese alojar grandes cantidades de ficheros con las lecturas del satélite, lo que nos brindaría la posibilidad de tener un extenso historial de los datos que el satélite envió cada día, permitiendo que los usuarios consultasen y realizasen sus estudios con información que pueda remontarse incluso años atrás.

   Esta mejora sería muy importante ya que la mayoría de estudios de contaminaciones estudian variaciones a lo largo de los años para comprobar el impacto de las

medidas que van surgiendo para reducir las mismas, incluyendo la observación de las variaciones de contaminaciones por estaciones o fechas señaladas.

Todas estas posibilidades que se añadirían gracias a un gran banco de datos mejorarían sustancialmente la calidad de la información suministrada a los interesados

2. La segunda mejora a la actual versión podría consistir, gracias al servidor más sofisticado, en la posibilidad de añadir instrucciones que se ejecuten en el servidor. Esto solucionaría el problema actual de nuestro servidor que permite alojar la página en la nube pero no permite la ejecución de instrucciones por parte del servidor.

   La mejora y apertura de la posibilidad de dicha ejecución de instrucciones nos permitiría crear programas en javascript que, por ejemplo, una vez capturados los datos por el sistema no solamente se encargue de mostrarlos sino que pueda trabajar con ellos, crear gráficas y en general, trabajar con la información ofreciendo múltiples opciones que no contemplen únicamente la muestra de datos.

   El usuario sería el gran beneficiario de dicha mejora pues tendría a su disposición múltiples estudios ya realizados y la posibilidad de visualizar unos datos ya preprocesados, obteniendo información mucha más precisa y detallada.

3. La tercera mejora que se nos ha ocurrido es la creación de una secuencia dinámica de la visualización de los datos a lo largo de varios días.

   El usuario podría seleccionar un intervalo de fechas y el programa realizaría una animación que vaya recorriendo esos días mostrando una especie de película de todas las lecturas de datos. De esta forma quedaría plasmado de manera muy visual la evolución de las nubes de concentraciones de gas, proporcionando interesantes conclusiones e información de la evolución de estas nubes de contaminación.

4. Por último, para finalizar las posibilidades de mejora de la aplicación existente, se podría añadir un delimitador de área geográfica.

   Se implementaría la posibilidad de marcar unos puntos en el mapa mundi, registrando el sistema las coordenadas y generando una figura geométrica que una todos esos puntos. Una vez indicada el área a estudiar los datos mostrados corresponderían solo a dicha área.

   Esto permitirá al interesado centrarse en una área específica de estudio, como puede ser un país o un área geográfica o continente.

6.2. Segunda rama

La segunda rama de mejoras, consistiría en la ampliación del conjunto de datos. Como al principio hemos mencionado, este proyecto gira en torno a un Satélite, el Sentinel-5 para ser concretos, el cual está centrado en el estudio completo de atmósfera y superficie terrestre. De esa extensa cantidad de mediciones tomadas, el proyecto que nos atañe se centra en la obtención exclusiva de los datos de NO2 pero el abanico es tan extenso que con la base ya realizada, se podría ir generando diversos extractores centrados en los demás datos proporcionados.

El Séntinel-5 es un satélite muy sofisticado y a continuación procedo a enumerar los diversos datos que seríamos capaces de obtener de este satélite y manteniendo toda la estructura de obtención y extracción de datos actual.

Servicios de monitoreo Atmosférico (CAMS). En este apartado se incluye toda la información obtenida sobre las múltiples variables atmosféricas con la finalizar de apoyar las políticas climáticas de los gobiernos europeos para la mejora de la calidad de vida de todos los ciudadanos europeos y mundiales.

Está centrada en el estudio de la calidad del aire junto con su interacción con el clima. Proporciona los datos de los siguientes gases:

1. OZONO
2. UV de la superficie
3. NO2
4. SO2
5. HCHO
6. CHOCHO
7. Aerosoles

Además de esto, proporciona información sobre los gases estratosféricos de cobertura global y posee espectrómetros de alta resolución. Por último estudia segmentos de tierra proporcionando imágenes y estudios hídricos y climáticos del área estudiada.

Por estos motivos creemos que el proyecto que presentamos es muy modular, ya que es posible únicamente desarrollando un extractor de datos, la creación de nuevos apartados que presenten concentraciones de los múltiples gases que hemos enumerado.

La creación de estos módulos dará un enorme abanico de posibilidades de estudios atmosféricos a nuestra aplicación y por lo tanto todo esto mejorará la sensación final del usuario con la aplicación.

Capítulo 7

Conclusiones

La cuestión climática, desde prácticamente los años 70, ha ido cobrando importancia. El rapidísimo aumento de las emisiones por la multiplicación de la sociedad de consumo y el mal llamado bienestar ha ido destruyendo el planeta, hasta que en la fecha actual nos encontramos en el último aviso antes de un punto de no retorno.

Es nuestro deber actuar a tiempo para evitar la degradación sin retorno de nuestro hogar, y que las generaciones venideras no tengan que sufrir nuestra irresponsabilidad. Aún está en nuestras manos ser dignos habitantes de este bello y frágil planeta.

Entre todas las actividades humanas que más nos hacen peligrar se encuentran las emisiones generadas por quemar combustibles fósiles, ya que generan CO2, principal culpable del efecto invernadero, y NO2, un gas muy tóxico y contaminante.

La problemática ambiental está en boca de todos pero gracias a este proyecto, hemos conocido el alcance real de esta contaminación, ya que a veces no somos conscientes de la realidad a la que nos enfrentamos. Las concentraciones son muy amplias y están extendidas por todo el mundo.

La conclusión, que podemos obtener del proyecto que presentamos, es la necesidad de comenzar a revertir esta situación en cada rincón del planeta, de forma inmediata y contundente, con el objetivo de evitar la degradación irreversible. Estas medidas únicamente serán eficaces si son planteadas con todas las evidencias científicas, empleando todo el abanico de medidores atmosféricos para saber dónde y de que forma actuar en cada caso.

Los satélites como el Sentinel, serán de una enorme utilidad para todos los estudios de medidas a tomar y la evolución de las políticas que vayan surgiendo, por ello deseamos que nuestro proyecto ayude a personas como nosotros, haciéndoles ver el alcance y gravedad de la problemática climática. Además de que científicos y estudiosos puedan emplear la aplicación para las mediciones y observación de la evolución atmosférica.

Capítulo 8

Conclusions

The climate issue, since practically the 1970s, has been gaining importance, the rapid increase in emissions due to the multiplication of the consumer society and welfare has been destroying the planet. Today we are at the last warning before a point of no return.

It is our duty to act in time to avoid the irreversible degradation of our planet and that future generations do not have to suffer our irresponsibility; it is still in our hands to be worthy inhabitants of this beautiful and fragile planet.

Among all the human activities that endanger us the most, are the emissions generated by burning fossil fuels, since they generate CO2, the main generator of the greenhouse effect and NO2, a very toxic and polluting gas.

The environmental problem is on everyone's lips but thanks to this project, we have learned the real extent of this pollution. Sometimes we are not aware of the reality we are facing. The concentrations are very wide and are spread all over the world.

The conclusion that we can obtain from this project is the need to start reversing this situation in every corner of the planet, immediately and decisively, in order to avoid irreversible degradation. These measures will only be effective if they are proposed with all the scientific evidence, using the full range of atmospheric meters to know where and how to act in each case.

Satellites such as the Sentinel will be extremely useful for the studies of measures to be taken and the evolution of policies that may arise. We want our project helps people like us, making them to see the scope and severity of the climate problem as well as scientists and scholars to use the application for the measurement and observation of atmospheric evolution.

Índice de figuras

Bibliografía

[1] Adding a new ssh key to your github account. https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account.

[2] Bots: An introduction for developers. https://core.telegram.org/bots.

[3] Connecting to github with ssh. https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh.

[4] Debugger for chrome - visual studio marketplace. https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome.

[5] European space agency. http://www.esa.int/.

[6] Gitpython is a python library used to interact with git repositories, high-level like git-porcelain, or low-level like git-plumbing. https://gitpython.readthedocs.io/en/stable/.

[7] jquery: The write less, do more, javascript library. https://jquery.com/.

[8] Leaflet:a javascript library for interactive maps. https://leafletjs.com/.

[9] Lightpick: Javascript date range picker - lightweight, no jquery. https://wakirin.github.io/Lightpick/.

[10] Live server - visual studio marketplace. https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer.

[11] Moment: parse, validate, manipulate, and display dates and times in javascript. https://momentjs.com/.

[12] netcdf4-python is a python interface to the netcdf c library. https://unidata.github.io/netcdf4-python/netCDF4/index.html.

[13] No2, el peligroso contaminante que afecta a la capacidad de atención de los niños. https://www.lavanguardia.com/natural/20170226/42280025427/no2-dioxido-de-nitrogeno-peligroso.html.

[14] Numpy is the fundamental package for scientific computing with python. https://numpy.org/.

[15] pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the python programming language. https://pandas.pydata.org/.

[16] Pycharm: the python ide for professional developers by jetbrains. https://www.jetbrains.com/pycharm/.

[17] Transfer-encoding - http — mdn. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Transfer-Encoding.

[18] Visual studio code - code editing. redefined. https://code.visualstudio.com/.

PASCAL
ENERO 2018
Ult. actualización 26 de junio de 2020
LaTeX lic. LPPL & powered by TEFLON CC-ZERO