

UNIVERSIDADE DE SANTIAGO DE  
COMPOSTELA



ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

# Seguemento de contaminación de ciudades españolas a partir de datos de TROPOMI

*Autor/a:*

**Hugo Gómez Sabucedo**

*Titores:*

**Joaquín Ángel Triñanes Fernández**

**Grao en Enxeñaría Informática**

**Febreiro 2024**

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría  
da Universidade de Santiago de Compostela para a obtención do Grao en  
Enxeñaría Informática





**D. Joaquín Ángel Triñanes Fernández**, Profesor/a do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela,

INFORMA:

Que a presente memoria, titulada (*Seguemento de contaminación de cidades españolas a partir de datos de TROPOMI*), presentada por **D. Hugo Gómez Sabucedo** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo nosa tutoría no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 1 de febreiro de 2024:

Titor/a,

Alumno/a,

(Joaquín Ángel Triñanes Fernández)    (Hugo Gómez Sabucedo)



# Agradecementos

Quero dar as grazas, en primeiro lugar, á miña familia, por apoiarme sempre ó longo de tódolos meus estudos. Sen o voso apoio e o voso amor, nada disto sería posible.

Tamén quero agradecer ós meus amigos, ós que estaban e ós que viñeron. Grazas por estar sempre aí, polo apoio e por servir de distracción, moitas veces, nos momentos malos.



# Resumo

A presenza de diferentes axentes contaminantes na atmosfera é a causa non só do cambio climático, senón de moitos problemas de saúde para as persoas. Actualmente, dispoñemos de múltiples satélites que nos permiten monitorizar os niveis destes parámetros de forma remota. Porén, analizar multitude de arquivos e datos, para diversos parámetros e días, pode ser unha tarefa tediosa.

É por iso que se propón este traballo, que ten como obxectivo deseñar un sistema para monitorizar a variabilidade e tendencia de diferentes parámetros obtidos a través de datos de satélite. Mediante a descarga destes datos e a súa transformación e agregación, poderemos reducir significativamente a cantidade de arquivos que será preciso analizar, o cal facilita o seguimento da contaminación por parte dos usuarios finais. O desenvolvemento deste sistema farase de forma que se poida acceder a el facilmente, e tendo en conta tamén a interoperabilidade do mesmo.





# Índice xeral

<b>1. Introducción</b>	<b>1</b>
1.1. Obxectivos do traballo . . . . .	2
1.2. Estrutura da memoria . . . . .	2
1.3. Descrición do sistema . . . . .	3
<b>2. Especificación de Requisitos</b>	<b>5</b>
2.1. Identificación dos usuarios . . . . .	5
2.2. Requisitos funcionais . . . . .	6
2.2.1. Obter datos de satélite da ESA [RF1] . . . . .	6
2.2.2. Transformar e agregar datos de satélite [RF2] . . . . .	6
2.2.3. Implementar un servidor ERDDAP local [RF3] . . . . .	6
2.3. Requisitos do produto . . . . .	7
2.3.1. A aplicación deberá ser rápida e eficiente [RNF1] . . . . .	7
2.3.2. Eficiencia de almacenamento [RNF2] . . . . .	7
2.4. Requisitos de rendemento . . . . .	7
2.4.1. Tempo de resposta razoable [RNF3] . . . . .	7
<b>3. Deseño</b>	<b>9</b>
3.1. Componentes do sistema . . . . .	9
3.1.1. Backend . . . . .	10
3.1.2. Middleware . . . . .	12
3.1.3. Frontend . . . . .	12
3.2. Tecnoloxías empregadas . . . . .	12
3.2.1. Anaconda . . . . .	12
3.2.2. HARP . . . . .	13
3.2.3. ERDDAP . . . . .	13
3.3. Desenvolvemento do sistema . . . . .	13
3.3.1. Introducción . . . . .	13
3.3.2. Descarga de datos . . . . .	15
3.3.3. Procesamento de datos . . . . .	18
3.4. Implementación do sistema . . . . .	22
3.4.1. Configuración da máquina de AWS . . . . .	22
3.4.2. Configuración de servidor ERDDAP . . . . .	23

3.4.3. Realización da interface . . . . .	24
<b>4. Probas</b>	<b>27</b>
4.1. Probas unitarias . . . . .	27
4.2. Probas de integración . . . . .	30
4.3. Probas de rendemento . . . . .	30
<b>5. Conclusións e posibles ampliacións</b>	<b>31</b>
<b>A. Manuais técnicos</b>	<b>35</b>
A.1. Instalación . . . . .	35
A.2. Mantemento . . . . .	37
<b>B. Manuais de usuario</b>	<b>39</b>
<b>Bibliografía</b>	<b>41</b>

# Índice de figuras

3.1. Área de interese sobre a que se realiza a busca. . . . .	16
3.2. Exemplo da interface para NO2 en novembro de 2023. . . . .	25
3.3. Exemplo da interface de ERDDAP co mapa de NO2 centrado en Madrid. . . . .	26
3.4. Exemplo da interface de ERDDAP cos valores de NO2 de Madrid. . . . .	26
4.1. Exemplo de pantalla de inicio de ERDDAP. . . . .	28
4.2. Exemplo da lista de datasets dispoñibles. . . . .	28
4.3. Exemplo da información e arquivos dun dataset. . . . .	29
4.4. Exemplo da mensaxe de erro que se amosa na interface se introducimos un mes non dispoñible. . . . .	29
B.1. Exemplo da interface que se mostra ó acceder á aplicación. . . . .	39



# Capítulo 1

## Introdución

O cambio climático é un asunto que, actualmente, está á orde do día. O aumento das temperaturas a nivel global é un asunto que preocupa non só ós científicos, senón tamén á poboación en xeral. As causas do cambio climático son moitas, pero podemos estar de acordo en que unha gran parte é debida á presenza de múltiples gases ou substancias químicas que afectan á atmosfera.

A presenza destes axentes contaminantes tamén afecta en gran medida ós seres humanos. Moitas enfermidades presentes na sociedade de hoxe en día son debidas a un exceso de contaminación atmosférica. Doenzas respiratorias, coma a asma, ou alerxias son problema de saúde que están cada vez máis presentes, e que afectan negativamente á calidade de vida das persoas.

Por iso, monitorizar a evolución destes axentes é algo crucial, sobre todo en entornos especialmente críticos, como poden ser as cidades, nas cales a conxunción dunha gran densidade de poboación e de movementos en áreas, polo xeral, pequenas, fai que sexa crucial coñecer cal é o nivel destes axentes. Desta forma, as autoridades poden adoptar medidas, tanto de saúde pública (como poden ser plans de prevencións de diferentes enfermidades), coma de protección do medio ambiente (limitando, por exemplo, a circulación de vehículos), co obxectivo de evitar que se causen danos ó entorno en que vivimos e reducir a mortalidade debida a estas enfermidades.

Así, dende a Axencia Espacial Europea (ESA) lanzouse, no ano 2017, un satélite de observación terrestre nomeado Sentinel-5 Precursor (ou Sentinel-5P). Dito satélite, que se enmarca dentro do Programa COPERNICUS, emprega un instrumento denominado espectrómetro, co obxectivo de monitorizar a contaminación atmosférica. Desta forma, TROPOMI proporciona observacións de forma diaria de gases atmosféricos que son chave á hora de monitorizar a calidade do aire e realizar predicións sobre a evolución da mesma, coma poden ser o metano ( $CH_4$ ), monóxido de carbono ( $CO$ ), formaldehído ( $HCHO$ ), dióxido de nitróxeno ( $NO_2$ ), ozono ( $O_3$ ), aerosol ou dióxido de xofre ( $SO_2$ ).

Este satélite proporciona datos a dous niveis distintos: L1, con datos crus; e L2, con variables xeofísicas derivadas dos datos de nivel 1. Existe outro nivel, o

Nivel 3 (L3), no cal as variables están mapeadas en escalas regulares. Será este o nivel que nos interese alcanzar nos nosos datos, para poder obter con eles un agregado mensual que nos permita ver a media ou crear climatoloxías para poder observar anomalías nos mesmos.

O obxectivo deste traballo é, por tanto, empregar os datos que nos proporciona este satélite para, mediante a realización de diferentes operacións sobre os mesmos, poder facer un seguimento da evolución ó longo do tempo da contaminación en diversas cidades de España. Decidiuse escoller este ámbito debido a que, ó tratarse dun área relativamente reducida, o procesamento dos datos será máis sinxelo, posto que será preciso procesar menos arquivos de menor tamaño.

## 1.1. Obxectivos do traballo

Os obxectivos deste traballo de fin de grao son os seguintes:

1. Mapear a contaminación atmosférica a partir de datos de sensores remotos, con especial relevancia en entornos urbanos
2. Monitorizar a variabilidade e tendencia dos diferentes parámetros satelitais relacionados coa contaminación do aire.
3. Desenvolver un visor online que permita representar de forma dinámica a información.
4. Implementar un sistema interoperable de distribución dos datos e produtos.

## 1.2. Estrutura da memoria

A presente memoria estrutúrase do seguinte xeito:

- **Capítulo 2:** Especificación de requisitos. Este capítulo describe detalladamente os requisitos e criterios que debe cumprir o proxecto.
- **Capítulo 3:** Deseño. Aquí explicaremos todo o relativo ó proceso de deseño do software ata acadar a súa versión final. Tamén se comenta a metodoloxía empregada para o seu desenvolvemento, así como as diferentes tecnoloxías das que se fixo uso no mesmo.
- **Capítulo 4:** Probas. Esta sección trata sobre as probas realizadas, así como os resultados das mesmas.
- **Capítulo 5:** Conclusións e posibles ampliacións. Neste último capítulo preséntanse as conclusións obtidas após realizar o proxecto, así como propostas de melloras sobre o mesmo ou posibles ampliacións futuras.

## 1.3. Descrición do sistema

Como se explicou nos obxectivos do traballo, o sistema que deseñemos terá como finalidade poder monitorizar a variabilidade e tendencia de diferentes parámetros, obtidos a través de datos de satélite, os cales se empregan para medir a contaminación do aire.

Será preciso, por tanto, deseñar un módulo mediante o cal poidamos descargar os datos de TROPOMI, obtidos a través da ESA, para poder traballar con eles, realizando as transformacións anteriormente mencionadas. Así, empregando as APIs proporcionadas, poderemos deseñar unha función que busque un determinado produto (por exemplo,  $NO_2$ ) no catálogo de COPERNICUS, filtrando os resultados para centrarnos nun área de interese (neste caso, a Península Ibérica) e datas concretas. Construindo unha query, e empregando a librería `requests`, descargaremos os arquivos para, posteriormente, transformalos. Ademais, os arquivos veñen en formato `.zip`, polo que será preciso, neste módulo, descomprimilos para poder obter os arquivos de formato `.nc` cos que debemos traballar. Este módulo será desenvolvido na linguaxe Python.

Por outra parte, deseñaremos outro módulo, tamén en Python, no cal poidamos traballar cos arquivos descargados. Os datos que descargamos son de nivel 2, polo que temos que realizar operacións cos mesmos para transformalos en datos agregados por mes. Para isto, será preciso empregar `HARP`, unha ferramenta que permite ler e procesar datos de satélite, permitindo a inxesta de arquivos e o traballo cos mesmos. Así, importaremos tódolos arquivos descargados e realizaremos as transformacións anteriormente citadas, para posteriormente exportar o resultado nun arquivo, tamén de formato `.nc`, nun directorio específico.

Os datos transformados almacenaranse nun servidor ERDDAP, que empregaremos como middleware. Este servidor proporciona unha forma sinxela de acceder a datasets científicos en formatos de arquivos comúns, facilitando a xeración de gráficos e mapas. Unifica diferentes tipos de servidores con diferentes estruturas e devolve os datos nun formato de arquivo común, así como estandariza as datas entre os mesmos. Ademais, é posible configurar un servidor ERDDAP propio e empregar os nosos propios datos. Esta será a opción que elixamos, xa que nos permite empregar os datos que nós mesmos transformamos.

Será preciso establecer, no servidor onde corra a aplicación, unha tarefa automática, para que, segundo se publiquen datos novos, estes estean dispoñibles na nosa aplicación. Esta tarefa deberá executarse tódolos meses, tendo en conta que os datos das medicións de satélite tardan uns días en estaren dispoñibles, e deberá descargar os datos dos parámetros que empregue a nosa aplicación, transformalos e almacenalos no directorio establecido.

Por último, deseñaremos unha interface web que permitirá consultar a evolución de cada un dos diferentes parámetros (escollendo un deles) ó longo do tempo.

Todo isto implementarémoslo nunha máquina de AWS, de forma que o sistema

se administrará dende a mesma. Así, poderemos acceder á interface web de forma sinxela, a través de internet, e toda a xestión da aplicación estará tamén centralizada. Nesta máquina almacenaremos os datos transformados que obteñamos do módulo de Python, e nela teremos tamén o servidor ERDDAP co cal acceder a ditos datos. Accedendo á máquina poderemos visualizar a interface web que mencionamos.



# Capítulo 2

## Especificación de Requisitos

Neste capítulo trataremos en profundidade os principais requisitos que debe cumprir o sistema e o traballo; isto é, as funcionalidades que se require que teña o sistema. Especificaremos os usuarios que empregarán o sistema, as funcionalidades que se lle atribúen a cada un e os compoñentes do sistema, así como os requisitos funcionais e non funcionais do mesmo.

### 2.1. Identificación dos usuarios

O obxectivo deste proxecto é proporcionar información de diferentes parámetros relacionados coa contaminación atmosférica, co obxectivo de poder realizar un seguimento de como evolucionou ó longo do tempo a contaminación en diferentes puntos da Península Ibérica. Temos, por tanto, un perfil de usuario claro: aquel que consultará os datos, ou usuario final [U1]. Por outra parte, identificaremos outro usuario, administrador [U2] que será o encargado de realizar o mantemento de todo o sistema, e que se encargará de arranxar os distintos problemas que poidan surxir co uso do mesmo.

- O **usuario final** ([U1]) é aquela persona á que vai dirixida principalmente o sistema, e que ten interese en poder monitorizar a evolución da contaminación por diferentes parámetros ó longo do tempo. Trátase dun perfil de persoa variada, dende un científico, un traballador público que queira elaborar plans para a redución da contaminación nas cidades, ou calquera cidadán con interese polo tema. Porén, as tarefas que se lle atribúen son sinxelas:
  1. **Consultar o nivel de contaminación.** Este usuario consultará, na interface web do servidor en que se desenvolva o sistema, o valor dun parámetro en concreto, de entre tódolos que estean dispoñibles, para un mes concreto.

- **O administrador do sistema [U2]** é a persoa que se encargará do mantemento do mesmo, monitorizando o funcionamento do servidor para que, en caso de que se produza algún erro, poder solventalo. O rol de administrador do sistema asumirao o alumno encargado de desenvolver este traballo. As súas principais tarefas serán:
  1. **Configurar o sistema.** Este usuario será o encargado de realizar a configuración inicial do sistema, implementando os diferentes módulos que o compoñen e descargando e procesando datos durante un período de tempo o suficientemente longo, que permita a correcta utilización do sistema.
  2. **Monitorear o sistema.** Como administrador, deberá encargarse de monitorizar o correcto funcionamento do sistema, vixiando os posibles problemas que poidan xurdir no mesmo. Especialmente, deberá facer énfase en controlar que, cada mes, se descarguen e procesen correctamente os arquivos correspondentes, para asegurar a máxima calidade nos datos que consulte o usuario final.

## 2.2. Requisitos funcionais

### 2.2.1. Obter datos de satélite da ESA [RF1]

A aplicación que se desenvolva debe ser capaz de obter, de forma automática, os datos de satélite dispoñibles no dataspace de COPERNICUS, na web da Axencia Espacial Europea. Isto farase empregando unha API de entre aquelas que ofrece a propia ESA. Estes datos deberán limitarse a unha área xeográfica de interese, que no noso caso será a Península Ibérica, e a un rango de datas determinados.

### 2.2.2. Transformar e agregar datos de satélite [RF2]

Os datos que se descarguen dende a web da ESA deberán ser transformados, de forma que serán agregados mensualmente para poder facer un seguemento da evolución da contaminación. Os arquivos resultantes deberán manter unicamente a información necesaria.

### 2.2.3. Implementar un servidor ERDDAP local [RF3]

No servidor onde se implemente este sistema, implementarase un servidor ERDDAP local, de forma que se facilite o acceso ós datos. Deberase configurar un *dataset* por cada un dos parámetros, e almacenar nel os diferentes arquivos.

## **2.3. Requisitos do produto**

### **2.3.1. A aplicación deberá ser rápida e eficiente [RNF1]**

A aplicación deberá ser capaz de, nun tempo razoable dende que estean dispoñibles, descargar e procesar os datos de satélite, de forma que os usuarios poidan acceder a estos datos o antes posible.

### **2.3.2. Eficiencia de almacenamento [RNF2]**

A aplicación debe ser eficiente en canto ó uso do almacenamento se refire. Tendo en conta que, aproximadamente, cada mes se descargarán en torno a 50GB de novos datos por cada produto que se procese, e que após o procesamento teremos datos cun tamaño resultante darredor de 500MB por parámetro, é crucial que a aplicación almacene só os datos estritamente necesarios.

## **2.4. Requisitos de rendemento**

### **2.4.1. Tempo de resposta razoable [RNF3]**

A aplicación debe ter un tempo de resposta razoable, de forma que os usuarios non deban agardar moito tempo para teren os resultados que desexan.



# Capítulo 3

## Deseño

Neste capítulo, explicaremos os diferentes compoñentes que forman o sistema e as tecnoloxías empregadas, así como o proceso de desenvolvemento e implementación do mesmo.

### 3.1. Compoñentes do sistema

No que respecta á arquitectura do sistema software desenvolvido, podemos identificar tres grandes compoñentes:

- **Backend.** O backend da aplicación está formado polos diferentes scripts de Python que se encargan da descargar os datos, mediante a API, da web de COPERNICUS, da súa descompresión, o procesamento e o almacenamento no directorio apropiado para que se poida acceder a eles. Ademais, cun script de Python e o programa `cron` de Ubuntu, definiremos unha tarefa a executarse automaticamente de forma mensual, de forma que, segundo teñamos dispoñibles novos datos, estes se descarguen e sexan accesibles ó usuario, para que teña sempre os datos máis recentes.
- **Middleware.** O middleware está formado polo servidor ERDDAP, que serve como unha capa intermedia entre o backend de Python e o frontend que verá o usuario final. Ademais, como comentamos na introdución, proporciona unha forma sinxela de acceder a datasets científicos en formatos de arquivos comúns, facilitando a xeración de gráficos e mapas. Coa implementación do noso propio servidor ERDDAP, poderemos almacenar nel, ademais dos datasets que xa veñen por defecto, uns cos nosos arquivos.
- **Frontend.** O frontend desenvolverase empregando HTML. Será un frontend sinxelo, de forma que, cando se acceda á aplicación na URL do servidor, o usuario vexa un mapa, acoutado á zona na que se realiza o estudo dos datos, e que poida elixir, por unha parte, un parámetro de entre os catro

que se proporcionan, e un mes no rango de meses dispoñibles no dataset, actualizándose a información de forma dinámica.

### 3.1.1. Backend

Como xa comentamos, o backend da aplicación desenvolveuse en Python. A elección desta linguaxe de programación para este módulo é sinxela: é a linguaxe de programación máis axeitada para traballar con grandes cantidades de datos. Ademais, é a linguaxe que ten maior compatibilidade con tódalas ferramentas que deberemos empregar para o procesado dos datos, o que fai que sexa, practicamente, a única opción a ter en conta.

Así, o backend de Python divídese en catro scripts diferentes, cada un cunha funcionalidade determinada.

#### Script principal e automatización

Por unha parte, temos o script principal, `app.py`. Este é o script dende o que se executa o programa. Nel, establécense os parámetros cos cales se executarán os módulos de descarga e procesado dos arquivos, comprobando tamén a súa validez (como, por exemplo, que a data de inicio da busca sexa anterior á data de fin). Pódese executar tanto por liña de comandos coma importando o arquivo e chamando á función.

Por outra parte, temos o script `auto.py`, o cal se encarga da automatización da descarga e procesado dos datos. Este será o script que executaremos con `cron` de forma mensual, para manter os datasets actualizados. Este script obtén, a partir da data actual, as datas para as que realizaremos a busca, que serán entre o primeiro e último día do mes anterior. Desta forma, executaremos o `job` de `cron`, cunha frecuencia mensual, o día 10 de cada mes. Polo tanto, no caso da execución que se realizaría o día 10 de xaneiro de 2024, as datas que se tomarían como referencia serían o 1 de decembro de 2023 como data de inicio, e o 31 de decembro de 2023 como data de fin. Decidiuse elixir o décimo día de cada mes para a actualización dos datos debido a que, como se explica en [1], o procesado dos arquivos resultantes das medicións dos satélites ten unha latencia de 5 días. Así, asegurámonos de que, cando se execute o script, tódolos arquivos que se deben ter en conta estarán dispoñibles.

#### Script de descarga de datos

En segundo lugar, temos o script de descarga de datos. A súa función principal é `obtenArquivos`, que recibe como parámetros a área de interese da busca, as datas de inicio e de fin, o parámetro para o que se realizará a busca e o directorio onde se almacenarán os arquivos. Se este directorio non existe, créao.

A continuación, empregando a API de oData, realiza unha consulta para obter os arquivos que se descargarán. Iterando por estes arquivos, obtén o seu Id e o nome e, se estes non existen (tanto en formato zip, por estaren comprimidos, ou en formato zip, por estaren descomprimidos), realiza a súa descarga, na función `descargaArquivo`. Posteriormente, descomprime o arquivo coa función `descomprimeArquivo`, os cales quedan dentro dunha carpeta co seu nome. Por tanto, unha vez descargados e descomprimidos tódolos arquivos, móveos á carpeta principal e elimina tódalas subcarpetas.

No que respecta á función de descarga de arquivos, en primeiro lugar chama a unha función para obter o token de autorización preciso para descargar os datos. A continuación, crea a url para a descarga e, mediante a librería `Session`, descarga o arquivo, sempre que a resposta do servidor sexa positiva. Por outra parte, a función de descompresión dos arquivos comproba, en primeiro lugar, que o ficheiro zip exista. Se é así, empregando a librería `zipfile`, extrae os arquivos no directorio (imprimindo unha mensaxe de erro en caso de que o arquivo sexa inválido) e, unha vez feito, bórrao.

### **Script de procesado de datos**

Por último, temos o script que se encarga de procesar os datos. Nel, o módulo principal que empregamos é `HARP` que, como se explica en 3.2.2, é unha ferramenta que permite a lectura e procesado de datos de satélite. Neste módulo temos diferentes funcións de procesado, unha para cada un dos diferentes parámetros que se ofrecen na aplicación, xa que as operacións deben personalizarse en función do parámetro. Así, establécese unha función xenérica que, en función do parámetro que se estea tratando, invoca a unha función ou outra e, unha vez procesados tódolos arquivos e exportados os datos, elimina os arquivos que se descargaron, co obxectivo de optimizar o almacenamento do servidor (posto que estamos falando de centenas de arquivos cada mes, con tamaños de entre 500MB e 1GB, os cales, unha vez procesados, non son de interese nin utilidade). Ademais, temos unha función denominada `obtenListaProdutos`, para obter os arquivos a procesar.

Malia que cada unha das funcións de procesado son bastante semellantes, posto que realizan unha serie de operacións similares entre elas, foi preciso particularizalas, xa que o nome das variables que exportaremos difire en función do parámetro. Estas funcións reciben unha lista de arquivos a procesar e un directorio onde almacenar o arquivo final procesado. Empregando a función `import_product` de `harp`, importamos os arquivos, realizando con eles unha serie de operacións para filtrar os datos segundo a súa calidade, como se indica nos diferentes manuais. Unha vez temos tódolos produtos importados, realizamos unha serie de operacións, propias para cada produto, para quedarnos cos datos dunha zona e obter con eles unha cuadrícula; e unha serie de post-operacións, globais a tódalas funcións, para realizar a agregación dos datos. A continuación, definimos o nome do arquivo a exportar (que segue o patrón `parametro_AAAAMM.nc`), e gardámolo.

### 3.1.2. Middleware

O middleware que empregamos será ERDDAP. Isto é un servidor de datos que proporciona unha forma simple e consistente de acceder e descargar múltiples datasets científicos, de diferentes fontes, nun formato común, facilitando a elaboración de gráficos e mapas [2]. Ademais, permite obter os datos en diversos formatos de arquivo, ademais de proporcionar funcións para estandarizar as datas dos resultados ou proporcionar os mesmos en formatos de imaxe personalizados. A instalación de ERDDAP de referencia é aquela feita pola National Oceanic and Atmospheric Administration (NOAA, Oficina Nacional de Administración Oceánica e Atmosférica) que, ademais de ter máis de 200 datasets, permite crear un servidor local para almacenar nel datos propios.

Para realizar esta instalación, só será necesario seguir os pasos que se indican en [3], o que explicaremos máis detalladamente en 3.3 e 3.4. En resumo, é preciso:

1. Dispoñer dunha versión de Java 17.
2. Configurar Tomcat.
3. Descargar os arquivos de configuración de ERDDAP e realizar os cambios precisos para adaptalo ó sistema.
4. Instalar o arquivo `.war` de ERDDAP.
5. Iniciar o servidor e configurar os datasets que queremos que sexan visibles no mesmo.

### 3.1.3. Frontend

Para o frontend do proxecto, crearemos unha aplicación sinxela en Tomcat, empregando HTML. Facendo uso da funcionalidade que nos proporciona ERDDAP para xerar imaxes a partir dos datos, crearemos unha interface na que teñamos, por unha parte, un mapa cos datos que queiramos visualizar e, por outra parte, dous selectores, un para seleccionar o parámetro a consultar e outro para seleccionar o mes e ano.

## 3.2. Tecnoloxías empregadas

### 3.2.1. Anaconda

Anaconda é unha distribución de Python, que simplifica a xestión e instalación de paquetes. Isto permitiu unha instalación rápida de Python, así coma dos diferentes paquetes empregados. A versión de Python que empregamos é Python 3.11.6. Entre os principais paquetes que se instalaron están: `os`, para empregar funcionalidades do sistema operativo; `shutil`, para traballar con arquivos;



`requests`, para poder realizar peticións HTTP de forma sinxela; `zipfile`, para traballar cos arquivos zip; `pandas`, para manexar datasets; ou `harp`, para empregar as funcións de HARP que mencionaremos en 3.2.2.

### 3.2.2. HARP

HARP é unha ferramenta para ler, procesar e comparar datos de satélite. Ademais de ferramentas para executarse por liña de comando, tamén contén unha interface de Python moi sinxela de usar. Esta ferramenta permite preprocesar diferentes datasets de forma que poidamos facer que teñan a mesma cuadrícula espacio-temporal, o mesmo formato de datos ou a mesma estrutura. Para empregala, deberemos importar os produtos que queiramos transformar, aplicando diferentes operacións para filtrar os datos en base á súa calidade, excluír certas variables, transformalos a cuadrículas regulares, etc.

### 3.2.3. ERDDAP

Como comentamos en 3.1.2, ERDDAP é un servidor de datos que nos permite acceder de forma sinxela a múltiples datasets científicos, tanto dispoñibles na rede como localmente, facilitando a elaboración de diferentes gráficos. Será esta a funcionalidade que empregaremos para elaborar a interface da aplicación, coa cal interactuarán os usuarios.

## 3.3. Desenvolvemento do sistema

### 3.3.1. Introducción

Para poder comezar a desenvolver o sistema, primeiro foi preciso realizar unha introdución á detección mediante satélite de diferentes datos e ó funcionamento dos satélites de TROPOMI. Para isto, foron de utilidade os seminarios dispoñibles en [5], de monitorización de alta resolución de  $NO_2$  dende o espazo con TROPOMI; [6], de ferramentas para analizar datos de alta calidade de satélite; e [7], de aplicacións de observacións de satélites para a calidade do aire e a exposición á saúde. Neles, explícase o funcionamento básico dos satélites e como miden e procesan os datos. Tamén se explican os diferentes niveis de datos que existen, como mencionamos na introdución. Dipoñemos de 4 niveis principais de datos:

1. **Nivel 0 (L0):** estes datos son datos crus, tal e como se obteñen das medicións do satélite, a máxima resolución.
2. **Nivel 1 (L1):** son os datos de nivel L0, pero con referencias temporais e información dos sensores empregados para as medicións, coma coeficientes de calibración radiométrica e xeométrica.

3. **Nivel 2 (L2):** os datos de nivel L1, pero con variables xeofísicas derivadas na mesma resolución e ubicación que os datos de nivel L1.
4. **Nivel 3 (L3):** as variables obtidas dos datos de nivel L2, mapeadas en forma de cuadrículas espazo-temporais, completas e con consistencia.

Empregar datos de nivel L3 ten diversas vantaxes:

- Podemos dispoñer dun único arquivo por día (ou por mes, ou o período de tempo que nos interese). Polas características do satélite, o seu ciclo orbital é de 16 días, realizando 14 órbitas por día, ou 227 órbitas por ciclo [9]. Polo tanto, para dous días distintos, é posible que un arquivo non cubra a totalidade da área de interese. Ó realizar as operacións de agregación dos datos, aseguramos ter un único arquivo por día, facendo a manipulación dos datos moito máis sinxela.
- Datos dispostos en cuadrículas uniformes. Coa agregación dos datos, podemos crear tamén unha cuadrícula para os mesmos, dispoñéndoo de forma regular no mapa ou área que nos interese estudar.
- Arquivos de menor tamaño. Os arquivos de nivel L2 conteñen unha maior cantidade de datos, unha gran parte dos cales pode non ser relevante para o noso sistema. O seu procesamento a nivel L3 permite eliminar estes datos e variables innecesarias, minguando o tamaño dos arquivos.
- Maior calidade nos datos. Nos diferentes manuais dos produtos, establécense diversos criterios de calidade asociados ós datos, que podemos empregar para descartar medicións de certas áreas pequenas do espazo en caso de que non teñan unha calidade suficiente. Desta forma, ó procesar os datos, podemos eliminar estas medicións de menor calidade e, como crearemos agregados por grandes períodos de tempo, evitaremos que queden vacíos.

O satélite SENTINEL 5P proporciona datos, cunha resolución espacial de  $7 \times 7 \text{ km}^2$ , de seis parámetros moi relacionados coa contaminación atmosférica: ozono ( $O_3$ ), tanto a columna troposférica (aquela que se forma en capas máis altas da atmosfera) coma a columna total; dióxido de nitróxeno ( $NO_2$ ), tanto a columna troposférica coma a total; dióxido de xofre ( $SO_2$ ), a columna total; monóxido de carbono ( $CO$ ), a columna total; metano ( $CH_4$ ), a columna total; e formaldehído ( $HCHO$ ), a columna total. Tratar todos estes parámetros era difícil de abordar, polo que foi preciso descartar dous deles para non telos en conta. Baseándonos na información dispoñible en [10], decidimos elixir os seguintes:  $NO_2$ ,  $CO$ ,  $SO_2$  e  $O_3$ . Escolléronse estes parámetros por seren os que máis problemas de saúde adoitan causar, ademais de por empregárense máis a miúdo nos diferentes mapas de contaminación.

### 3.3.2. Descarga de datos

Unha vez decididos os parámetros, podemos comezar coa descarga dos datos. Para isto, no Dataspace COPERNICUS temos dispoñibles múltiples APIs que podemos empregar para acceder e descargar os datos. Estas APIs poden ser de tres tipos: *Catalogue APIs*, que permiten buscar no catálogo de produtos dispoñibles e descargar os arquivos; *Streamlined Data Access (SDA) APIs*, que permiten obter datos de observación terrestre (EO) xunto con ferramentas de procesamento e análise; e outras APIs, que permiten, por exemplo, o acceso a datos EO empregando *buckets* de S3. No noso caso, elixiremos unha API do primeiro tipo, xa que é preciso poder buscar e filtrar os produtos para descargar aqueles que nos interesen.

Empregaremos a API de OData, que está baseada en APIs REST e permite, empregando mensaxes HTTP sinxelas, publicar e editar recursos identificados con URLs. A documentación completa da API atópase en [11]. En xeral, unha query OData contén a URL base xunto cunha serie de opcións, entre as que se atopan: *filter*, *orderBy*, *top*, *skip*, *count* e *expand*. Dentro da opción de *filter*, podemos filtrar por:

1. **Nome do produto:** *filter=Name*. Buscando un produto específico polo seu nome.
2. **Colección de produtos:** *filter=Collection/Name*. Buscar produtos dunha colección concreta (SENTINEL-1, SENTINEL-2, SENTINEL-3, SENTINEL-5P, SENTINEL-6 ou SENTINEL-1-RTC).
3. **Data de publicación:** *filter=PublicationDate*. Filtrando entre un intervalo de datas, que pode incluír (con **ge** e **le**) ou non (con **gt** e **lt**) as datas que se empreguen.
4. **Data de detección:** *filter=ContentDate/Start / filter=ContentDate/End*. Filtrando pola data de detección dos datos.
5. **Área xeográfica:** *filter=OData.CSC.Intersects(area=geography'SRID=4326;)*. Pódense filtrar os datos para elixir aqueles que crucen cun punto (**POINT(x,y)**) ou con polígono (**POLYGON()**, cunha lista de puntos).

Por outra parte, existen as seguintes opcións, que deben ir separadas dos filtros e entre elas empregando o símbolo \$:

- **orderBy:** para ordenar de forma descendente (**desc**) ou ascendente (**asc**), que é a opción por defecto. Pódese filtrar por *ContentDate/Start*, *ContentDate/End*, *PublicationDate* ou *ModificationDate*.
- **top:** para limitar o número máximo de ítems que devolverá a query. Por defecto é 20, e pode establecerse como un número enteiro entre 0 e 1000.

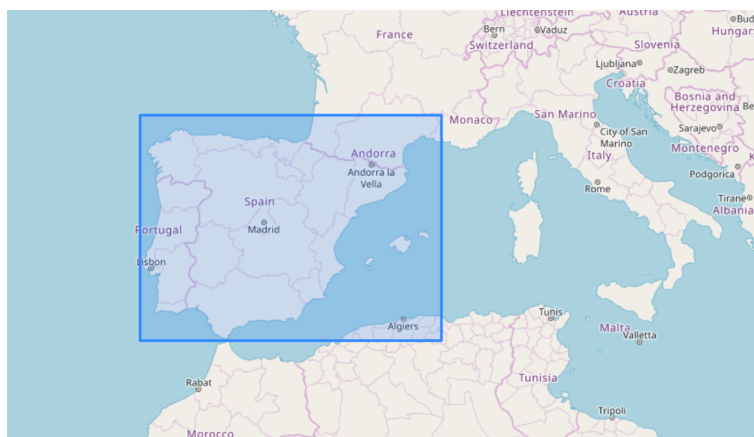


Figura 3.1: Área de interese sobre a que se realiza a busca.

- **skip**: permite saltar un certo número de resultados para, por exemplo, poder paxinar en caso de que a query obteña moitos resultados. O valor por defecto é 0, e pode tomar un valor entre 0 e 1000.
- **count**: para obter o número exacto de produtos que coincidan coa query. Por defecto, está establecida a **False**, deshabilitada.
- **expand**: permite ver os metadatos completos de cada un dos resultados que obtivo a query. Acepta como argumentos *Attributes* ou *Assests*.

Para realizar a busca dos produtos, unicamente é preciso empregar unha librería que nos permita facer peticións HTTP. Neste caso, elixiuse a librería **requests** de Python, xa que permite realizar estas peticións dunha forma moi sinxela. A URL base dende a que obteremos os datos é <https://catalogue.dataspace.copernicus.eu/odata/v1/Products>, engadiremos os seguintes filtros:

- **Área xeográfica**: limitaremos a busca á Península Ibérica, para o que definiremos o seguinte polígono: `POLYGON((-9.647785 35.912135,-9.647785 44.230843,4.916224 44.230843, 4.916224 35.912135,-9.647785 35.912135))`. Esta área é a que se mostra na imaxe 3.1
- **Colección de produtos**: filtramos os produtos da colección **SENTINEL-5P**.
- **Nome de produto**: empregamos o filtro `contains(Name, 'S5P_OFFL_{parametro}')`, para obter os produtos do parámetro que nos interese.
- **Data de detección**: coas datas que se lle pasan á función dende o módulo principal da aplicación, filtramos en base á data de detección dos datos xa que, como comentamos en 3.1.1, os datos tardan varios días en publicarse dende que se miden.

- Aplicamos ademais a opción `top` para obter un número maior de resultados, que estableceremos a 100. Isto débese a que temos, de media, dous produtos por cada día que se busca e, no noso caso, ó facer buscas mensuais, o número de arquivos que se descargan será en torno a 60, moi superior ó límite de 20 que establece por defecto a API.

Os resultados obtidos desta petición almacénanse nunha variable que se convertirá a un `DataFrame` de `pandas`, que contén, entre outros, o nome do arquivo, o seu identificador, a data de detección e publicación ou a pegada xeográfica. Así, iremos iterando sobre o `DataFrame` para, empregando o Id do produto e o seu nome, descargar os diferentes arquivos.

Para poder descargar os produtos, é preciso xerar un token de autenticación, xa que é preciso estar rexistrado. O proceso de rexistro é moi sinxelo, e pode facerse dende a páxina principal de COPERNICUS. No noso caso, decidiuse elixir o correo corporativo da universidade, xunto cun contrasinal xenérico. Foi preciso tamén crear unha función, `get_keycloak`, que, recibindo como parámetros o usuario e o contrasinal, devolve o token de acceso. É preciso ter en conta as limitacións e cuotas que se explican en [12], onde se explica que cada token de acceso é válido durante 10 minutos, podendo ter un máximo de 100 sesións activas simultaneamente. Por iso, foi preciso xerar o token de acceso na propia función de descarga, e non antes de realizar a mesma xa que, desta forma, corríase o risco de que a sesión expirase antes de poder descargar tódolos arquivos, o cal provocaría un fallo.

Unha vez temos o token de autenticación, creamos un obxecto de tipo `Session`, no cal estableceremos, mediante o método `.headers.update()`, o token de autenticación. A continuación, establecemos a url na que se atopa o arquivo, que será `https://catalogue.dataspace.copernicus.eu/odata/v1/Products(id)/$value`, onde id é o identificador do produto que vamos descargar. Antes de descargar o arquivo, debemos comprobar o código de estado de HTTP da resposta, para comprobar que todo sexa correcto. Desta forma, descargaremos o arquivo se o código de estado é 200 (OK) ou 308 (Permanent redirect). Se o estado é dos tipos 30X, significa que precisamos accións adicionais para completar a petición, polo que volveremos a repetila. Se o estado é correcto, escribiremos o contido do arquivo nun ficheiro `.zip`, no directorio que corresponda.

Inmediatamente despois de descargar o arquivo, temos que descomprimilo posto que, como vimos de comentar, é preciso descargalo en formato ZIP. Para isto, temos unha función moi sinxela, denominada `descomprimeArquivo`, que comproba que o arquivo exista e emprega a librería `zipfile` para extraelo, comprobando tamén que sexa un arquivo zip válido (capturando a excepción `zipfile.BadZipFile`). Unha vez descomprimido, eliminaremos o ficheiro zip, posto que xa non será necesario. Debemos ter en conta que o ficheiro final que queremos obter, en formato `netCDF` (ou `.nc`) non se atopará directamente no directorio en que se extrae, senón que se atopa dentro dunha subcarpeta co nome do arquivo

que acabamos de descargar. Por tanto, unha vez descargamos e descomprimos tódolos arquivos, deberemos mover os ficheiros .nc ó directorio de descarga, podendo eliminar posteriormente tódolos subdirectorios.

### 3.3.3. Procesamento de datos

Como xa comentamos, para procesar os datos empregaremos, principalmente, HARP 3.2.2, unha ferramenta de lectura e procesado de datos de satélite, que se pode empregar en Python de forma moi sinxela. A idea básica é importar unha serie de produtos, ós cales se lles aplican unha serie de operacións e postoperacións para transformalos en arquivos en cuadrículas regulares, e posteriormente exportalos. Así, deseñamos unha función xenérica `transformL3`, que recibe a ruta onde están os arquivos a procesar e o tipo de produto dos mesmos, e a ruta onde se exportar á o arquivo resultante. Así, podemos optimizar o código, realizando dende esta función a chamada a cada unha das funcións deseñadas para cada un dos parámetros. Unha vez se procesa o arquivo en cuestión, elimínanse tódolos arquivos que se empregaron nesta operación, para cumprir co requisito [RNF2].

A continuación, explícarase máis detalladamente o proceso de transformación de cada un dos produtos.

#### Transformación dos datos de CO

A función de procesado dos datos de monóxido de carbono recibe, ó igual que as demais, unha lista de arquivos e unha ruta onde almacenar os arquivos procesados. Con esta lista de arquivos, podemos iterar sobre ela, de forma que importamos os arquivos de un en un. Isto faise así, e non todos de golpe, para que o proceso de importado sexa máis eficiente e previr fallos ó ter que importar numerosos arquivos simultaneamente. Desta forma, empregamos a función `import_products` de HARP, a cal nos devolve un produto de HARP que engadiremos a unha lista. Esta función pode recibir unha serie de operacións que se aplicarán ós arquivos á hora de importalos. A lista das posibles operacións que se poden empregar en HARP atópase en [13]. No caso do monóxido de carbono, temos as seguintes operacións:

1. `CO_column_number_density>50` : con isto, filtramos os datos en base á calidade dos mesmos. Segundo as indicacións do manual de produto de CO [18], recoméndase ignorar os datos cun valor `qa_value` menor que 0.5. Porén, como se explica en [14], a variable `qa_value` é mapeada como `CO_column_number_density_validity`.
2. `derive(CO_column_number_density [Pmolec/cm2])`: a operación `derive` serve para cambiar o tipo de datos ou unidade dunha variable. Neste caso, cambiaremos o valor da columna de monóxido de carbono (que no arquivo

original se denomina `carbonmonoxide_total_column`) a petamoléculas por centímetro cadrado, xa que esta vén en moles por metro cadrado, para traballar con valores numéricos máis lexibles.

3. `keep(latitude, longitude, latitude_bounds, longitude_bounds, CO_column_number_density)`: con esta operación, indicamos que queremos incluír estas variables no produto.

Unha vez importamos tódolos produtos, empregaremos a función `execute_operations` para realizar operacións sobre os mesmos. Ó pasar unha lista de produtos á función, estes agruparanse despois de realizar as operacións sobre os mesmos. Neste caso, as operacións a realizar son:

1. `bin_spatial(2001, 35, 0.005, 3001, -10, 0.005)`: esta operación mapea tódalas observacións nunha cuadrícula espacial de latitude/longitude. Neste caso, creamos unha cuadrícula de 2001 puntos de alto, comezando no paralelo 35°, cun *offset* de 0.005°; e 3001 puntos de ancho, comezando no meridiano -10°, cun *offset* de 0.005.
2. `exclude(latitude_bounds_weight, longitude_bounds_weight, weight, latitude_weight, longitude_weight)`: excluimos estas variables do produto final.
3. `derive(latitude{latitude})` e `derive(longitude{longitude})`: derivamos as coordenadas de latitude e longitude da nova cuadrícula.

Por último, aplicamos unha serie de post-operacións, que definimos de forma global nunha variable, xa que serán as mesmas para tódolos produtos:

1. `bin()`: realiza un promedio de tódalas variables do produto na dimensión temporal.
2. `squash(time, (latitude, longitude))`: elimina a dimensión *time* para unha lista de variables (neste caso, *latitude* e *longitude*).

Unha vez temos o arquivo resultante, almacenámolo no directorio indicado. Para obter o nome do arquivo, seguiremos a expresión `CO_AAAAMM`, onde AAAA é o ano e MM é o mes do arquivo que exportamos. Isto obtémolo directamente do nome de calquera un dos arquivos que empregamos.

### Transformación dos datos de $NO_2$

Neste caso, o proceso é igual ó anterior. Os nomes das variables mapeadas atópanse dispoñibles en [15]. As operacións que aplicamos ó importar os produtos son:

1. `tropospheric_NO2_column_number_density_validity>75`: neste caso, o manual de dióxido de nitróxeno ([19]) indícanos que debemos considerar os datos cun valor de `qa_value` maior a 0.75.
2. `derive(tropospheric_NO2_column_number_density [Pmolec/cm2])`: derivamos a variable que orixinalmente se denomina `nitrogendioxide_tropospheric_column` das súas unidades orixinais ( $\text{mol}/\text{m}^2$ ) a  $\text{Pmolec}/\text{cm}^2$ .
3. `keep(latitude, longitude, latitude_bounds, longitude_bounds, tropospheric_NO2_column_number_density)`: incluimos as variables no produto.

As operacións que se executan sobre os datos son:

1. `bin_spatial(2001, 35, 0.005, 3001, -10, 0.005)`: creamos unha cuadrícula de 2001 puntos de alto, comezando no paralelo  $35^\circ$ , cun *offset* de 0.005°; e 3001 puntos de ancho, comezando no meridiano  $-10^\circ$ , cun *offset* de 0.005.
2. `exclude(latitude_bounds_weight, longitude_bounds_weight, weight, latitude_weight, longitude_weight)`: excluimos estas variables do produto final.
3. `derive(latitude{latitude})` e `derive(longitude{longitude})`: derivamos as coordenadas de latitude e lonxitude da nova cuadrícula.

Aplicamos como post-operacións as mesmas que no anterior caso. O arquivo resultante almacénase no directorio indicado, e o seu nome segue a expresión `NO2_AAAAMM`, onde AAAA é o ano e MM é o mes o arquivo que exportamos.

### Transformación dos datos de $O_3$

Para os datos do ozono, aplicamos as seguintes operacións ó importar os produtos:

1. `O3_column_number_density_validity>50`: neste caso, o manual do ozono ([20]) indícanos que debemos considerar os datos cun valor de `qa_value` maior que 0.5.
2. `derive(O3_column_number_density [Pmolec/cm2])`: derivamos a variable que orixinalmente se denomina `ozone_total_vertical_column` das súas unidades orixinais ( $\text{mol}/\text{m}^2$ ) a  $\text{Pmolec}/\text{cm}^2$ .
3. `keep(latitude, longitude, latitude_bounds, longitude_bounds, O3_column_number_density)`: incluimos as variables no produto.

Unha vez importados, executamos as seguintes operacións:



1. `bin_spatial(2001, 35, 0.005, 3001, -10, 0.005)`: creamos unha cuadrícula de 2001 puntos de alto, comezando no paralelo  $35^\circ$ , cun *offset* de 0.005°; e 3001 puntos de ancho, comezando no meridiano  $-10^\circ$ , cun *offset* de 0.005.
2. `exclude(latitude_bounds_weight, longitude_bounds_weight, weight, latitude_weight, longitude_weight)`: excluímos estas variables do produto final.
3. `derive(latitude{latitude})` e `derive(longitude{longitude})`: derivamos as coordenadas de latitude e lonxitude da nova cuadrícula.

As post-operacións que se aplican son as mesmas que nos anteriores casos. O arquivo resultante almacénase no directorio indicado, e o seu nome segue a expresión `03_AAAA_MM`, onde AAAA é o ano e MM é o mes o arquivo que exportamos.

### Transformación dos datos de $SO_2$

Por último, para o caso do dióxido de xofre, seguimos a mesma lóxica que nos casos anteriores. Os nomes dos mapeos das variables atópanse en [17]. Aplicamos sobre os datos que importamos as operacións:

1. `S02.column_number_density_validity>50`: neste caso, o manual do dióxido de xofre ([21]) indícanos que debemos considerar os datos cun valor de `qa_value` maior que 0.5.
2. `derive(03.column_number_density [Pmolec/cm2])`: derivamos a variable que orixinalmente se denomina `sulfurdioxide_total_vertical_column` das súas unidades orixinais ( $mol/m^2$ ) a  $Pmolec/cm^2$ .
3. `keep(latitude, longitude, latitude_bounds, longitude_bounds, S02.column_number_density)`: incluímos as variables no produto.

Sobre os datos importados, executamos as seguintes operacións:

1. `bin_spatial(2001, 35, 0.005, 3001, -10, 0.005)`: creamos unha cuadrícula de 2001 puntos de alto, comezando no paralelo  $35^\circ$ , cun *offset* de 0.005°; e 3001 puntos de ancho, comezando no meridiano  $-10^\circ$ , cun *offset* de 0.005.
2. `exclude(latitude_bounds_weight, longitude_bounds_weight, weight, latitude_weight, longitude_weight)`: excluímos estas variables do produto final.
3. `derive(latitude{latitude})` e `derive(longitude{longitude})`: derivamos as coordenadas de latitude e lonxitude da nova cuadrícula.

As post-operacións que se aplican son as mesmas que nos anteriores casos. O arquivo resultante almacénase no directorio indicado, e o seu nome segue a expresión `S02_AAAA_MM`, onde AAAA é o ano e MM é o mes o arquivo que exportamos.

## 3.4. Implementación do sistema

Unha vez comprobado que tódolos scripts funcionan correctamente, comezamos coa implementación do sistema deseñado na máquina de AWS que actuará como servidor. Os pasos a seguir serán: configurar por completo a máquina, instalando todo o software e paquetes precisos; instalar e configurar o servidor ERDDAP, para que lea os datos que se almacenan na máquina; e a implementación da interface á que accederán os usuarios.

### 3.4.1. Configuración da máquina de AWS

Como se comentou anteriormente, empregaremos os Servizos Web de Amazon para configurar unha instancia de EC2 e empregala como servidor, instalando nela o servidor ERDDAP e procesando os datos. Desta forma, facilítase que calquera usuario poida acceder dunha forma sinxela á aplicación dende a web.

A instancia que se elixiu é de tipo `t3.xlarge`, que dispón de 4 CPUs virtuais con 16 GiB de memoria, cunha AMI (Amazon Machine Image) de Ubuntu Jammy 22.04. A esta instancia acoplóuselle un volume de almacenamento SSD de tipo `gp2`, con 250 GB de almacenamento, que permite 750 IOPS (*Input/Output Operations per Second*). Ademais, obtívose unha dirección IP elástica, que se asociou á máquina, de forma que garantizamos que a dirección IP non vai cambiar co tempo, e a aplicación será accesible sempre dende a mesma dirección. Esta dirección IP é a `13.53.65.240`. Todos estes recursos están asignados á rexión `eu-north-1` de Europa (Estocolmo), na zona de dispoñibilidade `eu-north-1a`.

Para poder acceder correctamente á máquina, debemos configurar as regras de entrada e saída do grupo de seguridade asociado á mesma. Neste caso, configuramos as regras de saída para permitir o tráfico HTTP e HTTPS (portos 80 e 443) a tódolos destinos. No que ás regras de entrada se refire, permitimos tamén a entrada de todo o tráfico HTTP e HTTPS (portos 80 e 443); o tráfico SSH (porto 22), tamén dende tódolos orixes, para poder conectarnos remotamente á máquina empregando unha terminal; e o tráfico TCP personalizado ó porto 8080, xa que é o porto que emprega Tomcat, onde estará o servidor ERDDAP e a aplicación web.

Unha vez que temos a máquina funcionando, podemos instalar todo o software que será preciso para executar a aplicación. En primeiro lugar, instalamos Anaconda, unha distribución de Python que simplifica a xestión e instalación de paquetes. A súa instalación é moi sinxela, pois só é preciso descargar o instalador para Ubuntu e executalo, seguindo os pasos que indica. Unha vez instalado Anaconda, crearásenos un entorno, que será dende o cal descarguemos os paquetes e executemos a aplicación. A lista de paquetes completa atópase no anexo A. A continuación, debemos instalar Java, na versión 17. Debemos empregar esta versión, ou superior, para poder configurar correctamente o servidor ERDDAP. Tamén deberemos instalar `nodejs`, xunto con `npm` (o xestor de paquetes de `nodejs`), para

poder executar a interface web.

O seguinte paso será instalar e configurar Tomcat. Debemos crear un usuario específico, `tomcat`, sen privilexios, para facer máis segura a instalación. Posteriormente, descargamos e extraemos o *core build* de Tomcat nun directorio (que, no noso caso, será `/opt/tomcat`), sobre o cal podemos conceder permisos ó usuario `tomcat`. Tamén debemos configurar os administradores, para poder acceder ás páxinas de *manager* e *host-manager*. O último paso será crear un servizo en `systemd`, para que Tomcat poida executarse en segundo plano. Nel, configuraremos as variables de entorno coma `JAVA_HOME`, `JAVA_OPTS`, `CATALINA_BASE` e `CATALINA_HOME`. Recargamos `systemd` e xa poderemos iniciar o servizo de Tomcat, o cal permitiremos que se execute co inicio do sistema. Por último, debido a que Tomcat emprega o porto 8080, debemos permitir o tráfico nese porto tamén dende a máquina, empregando o comando `sudo ufw allow 8080`.

Neste punto, empregando os scripts que deseñamos en 3.3, executamos o módulo principal (`app.py`), decargando os datos para os catro parámetros que teremos dispoñibles para un período de tempo razoable. No noso caso, decidimos descargar tódolos datos do ano 2023, dende xaneiro ata decembro. Posteriormente, este rango temporal poderase ampliar. Configuraremos tamén `cron`, para poder executar o script de forma periódica. Como comentamos en 3.1.1, deseñamos un script que, cando se executa, obtén as datas de inicio e fin coma o primeiro e último día do mes anterior, e descarga o produto. Este será o script que empregaremos no job de `cron`. Executando `crontab -e`, poderemos ver tódolos *jobs activos*, así como definir novos. O que definiremos terá a seguinte estrutura: `0 1 10 * * /home/ubuntu/anaconda3/bin/python /home/ubuntu/TFG/auto.py &>/home/ubuntu/TFG/pylogs.txt`. Isto executará, o día 10 de cada mes, á 01:00, o script `auto.py`, e redirixe o `stdout` e `stderr` ó arquivo `pylogs.txt`, para que o administrador o poida consultar e comprobar a correcta execución.

Finalmente, con Tomcat executándose e os datos dispoñibles no servidor, podemos comezar coa configuración do servidor ERDDAP.

### 3.4.2. Configuración de servidor ERDDAP

Antes de comezar a instalar o servidor ERDDAP, debemos facer pequenos cambios nos arquivos de configuración de Tomcat. No arquivo `server.xml` (`/opt/tomcat/conf/server.xml`), debemos cambiar o valor do parámetro `connectionTimeout` dentro da etiqueta `<Connector>` do porto 8080. No arquivo `context.xml` (`/opt/tomcat/conf/context.xml`), debemos cambiar o valor de `cacheMaxSize` da etiqueta `<Resources>` a 80000.

A continuación, debemos descargar dende un repositorio de GitHub, proporcionado en [3], un arquivo zip cos arquivos de configuración de ERDDAP. Debemos descomprimir este arquivo na carpeta de Tomcat, creando o directorio `/opt/tomcat/content/erddap`. Dentro dos arquivos desta carpeta, atopamos

`setup.xml`, o arquivo coa configuración que especifica como se comporta o servidor ERDDAP. Nel, debemos cambiar: `< bigParentDirectory >`, que no noso caso establecemos como `/home/ubuntu/erddap/`; `< baseUrl >`, que é a URL dende a que se accederá ó servidor; `< emailEverythingTo >`, que é a dirección de correo á que se envían mensaxes de erro; e tódolos campos con etiquetas `< email* >` e `< admin*`, con información relativa ó administrador do sistema (por exemplo, `< adminIndividualName >`, co noso nome; `< adminCity >`, coa cidade; ou `< adminEmail >`, co noso email).

Con isto listo, podemos instalar o arquivo war de ERDDAP. Para isto, podemos descargalo directamente dende a URL que se indica en [3], descargándoo directamente no directorio `/opt/tomcat/webapps`. Con isto, teríamos configurado xa o servidor ERDDAP cunha instalación básica, con tódolos datasets que veñen por defecto. Este está accesible na url `http://13.53.65.240:8080/erddap/index.html`. Para poder incluír os nosos propios datasets, deberemos modificar o arquivo `/opt/tomcat/content/erddap/datasets.xml`, seguindo os consellos que se indican en [4]. Os creadores de ERDDAP proporcionan, ademais, unha ferramenta denominada **GenerateDatasetsXml**, o cal permite facilitar este proceso.

Para cada datasets, crearemos unha etiqueta `< dataset >`, na cal configuraremos parámetros coma o `type` (EDDGridFromNcFiles), o `datasetId`, o seu estado (`active=true` ou `active=false`), cada canto se recarga e se actualiza (`< reloadEveryNMinutes >`, que estableceremos a 10080 (7 días); e `< updateEveryNMillis >`, que será 60000 (1 minuto)), o directorio onde se atopan os arquivos (no caso de, por exemplo, o monóxido de carbono, será `/home/ubuntu/TFG/data/CO/`), ou a expresión regular no nome dos arquivos (que é `.*\nc`). Podemos engadir atributos, coma o nome e título do dataset, un resumo dos datos que ten ou as variables do mesmo. Tamén engadiremos tres **axisVariable**: unha para o tempo, que podemos derivar do nome do arquivo, xa que este é unha expresión regular; a latitude, que se obtén directamente do arquivo; e a lonxitude, que tamén se obtén do arquivo. Por último, engadiremos unha **dataVariable**, que será a variable que se mostrará no mapa. Engadiremoslle atributos coma o tipo de dato (`double`), o nome, o valor mínimo e o valor máximo. A configuración completa do arquivo `datasets.xml` está no anexo A.

### 3.4.3. Realización da interface

Para crear a interface, crearemos unha aplicación nova en Tomcat, para illala do servidor ERDDAP. Crearemos unha nova carpeta no directorio `/opt/tomcat/webapps`, denominada TROPOMIVisor. Este será o nome da aplicación, que teremos que poñer na URL para acceder á mesma. Desta forma, a url para acceder á aplicación sería `http://13.53.65.240:8080/TROPOMIVisor/`. O deseño da interface será sinxelo, xa que ERDDAP permite acceder a un gráfico dos datos mediante unha URL específica. Por tanto, faremos uso desta función para o deseño. No corpo do código HTML estableceremos un selector con catro op-

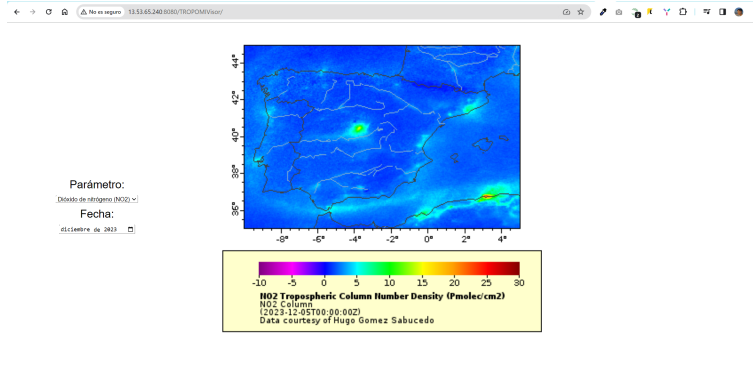


Figura 3.2: Exemplo da interface para NO2 en novembro de 2023.

cións, unha por cada un dos parámetros. Tamén estableceremos un `input` de tipo `month`, para poder seleccionar o mes e ano, que comece no primeiro mes do cal temos datos (xaneiro de 2023). Todo isto meterémolo dentro dun contedor, para que se posicionen xuntos. Definiremos tamén unha imaxe, a cal situaremos centrada na páxina, e co `source` por defecto coma unha imaxe do mapa de monóxido de carbono, que se actualizará automaticamente mediante código de JavaScript.

Coa etiqueta `< script >`, introduciremos o código de JavaScript. Definiremos unha función actualizar, na cal obteremos o valor dos selectores, para poder obter os parámetros que son precisos para parametrizar a query. Con isto, cambiaremos o `source` da imaxe, para que se poida actualizar. Engadiremos un `addEventListener` ós selectores, que se actualice cando se produza algún cambio, chamando á función actualizar.

Un exemplo de como resultaría esta interface atópase na figura 3.2, onde se ven os dous selectores definidos. O selector dos parámetros conta cos catro parámetros definidos, e o selector de data permite elixir, en primeiro lugar, o ano desexado e, en segundo lugar, o mes. Resulta especialmente interesante observar os valores obtidos no caso do dióxido de nitróxeno, xa que se aprecia con claridade tanto a resolución dos datos, como o efecto que se comentou na introdución da concentración da contaminación en áreas urbanas, coma poden ser Madrid ou Barcelona, observando mesmo altos valores de contaminación na zona do norte de Arxelia.

Ademais, malia que esta funcionalidade non se proporciona directamente na interface da aplicación, podemos empregar a interface de ERDDAP (accedendo a ela en <http://13.53.65.240:8080/erddap/index.html>) para ver outro tipo de gráficos. Así, podemos centrar o mapa nunha zona xeográfica específica, por exemplo Madrid, como se pode ver na figura 3.3. Ademais, tamén podemos crear gráficos de liñas, para ver os distintos valores dun parámetro ó longo dos meses de forma máis gráfica. Isto é o que se ve na figura 3.4, onde se observa a evolución da contaminación por dióxido de nitróxeno nesta área metropolitana, e como o

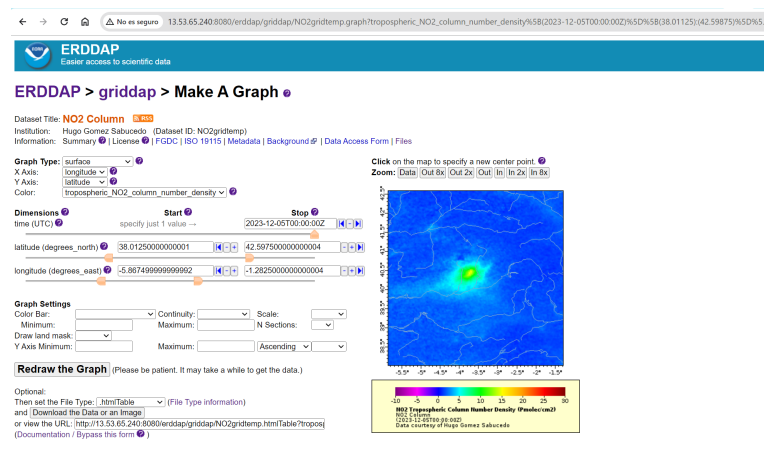


Figura 3.3: Exemplo da interface de ERDDAP co mapa de NO2 centrado en Madrid.

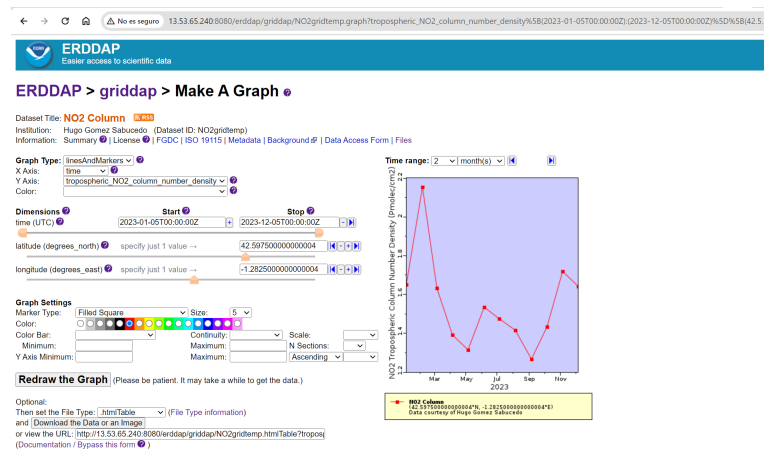


Figura 3.4: Exemplo da interface de ERDDAP cos valores de NO2 de Madrid.

seu valor é menor nos meses de verán, cando hai máis xente de vacacións.

# Capítulo 4

## Probas

As probas que detallaremos a continuación buscan identificar posibles fallos ou funcionamentos inadecuados do sistema, para garantir que este funciona de forma óptima. Así, conseguiremos o obxectivo de ter un sistema robusto e fiable.

### 4.1. Probas unitarias

As probas unitarias son de vital importancia para comprobar o correcto funcionamento do código a nivel de funcións ou métodos. Estas probas consisten en comprobar compoñentes individualmente, garantindo que funcionan de maneira axeitada.

Estas probas unitarias fixéronse de maneira manual, comprobando as diferentes funcións de extracción e transformación de datos. Comprobouse, por exemplo, o funcionamento da función de descarga de datos, revisando que a mesma devolve os datos no formato agardado (arquivos netCDF) e que, en caso de producirse un erro na descarga, o programa pode continuar descargando o resto de arquivos correctamente, minimizando a perda de datos. Así, se a descarga é satisfactoria, mostraranse os nomes dos arquivos que se descargaron.

Tamén se realizaron probas sobre as funcións que importan e transforman os arquivos a datos agregados de nivel L3. Neste caso, comprobouse que a función importa correctamente os arquivos, e que os transforma empregando as operacións definidas, producindo o arquivo resultante tamén en formato netCDF e co patrón de nome desexado.

Unha vez que se implementou o servidor ERDDAP, tamén se realizaron probas sobre o mesmo, para comprobar que todo estivera en orde. Xa que o servidor se implementou de forma incremental, engadindo os diferentes datasets e arquivos paulatinamente, puidemos comprobar non só o funcionamento do servidor en si, senón tamén que, conforme se xeraban novos arquivos, estes aparecerían dispoñibles no servidor tras o período de tempo que determinamos en 3.4.2. O servidor ERDDAP tamén dispón da súa propia interface, a cal é acce-

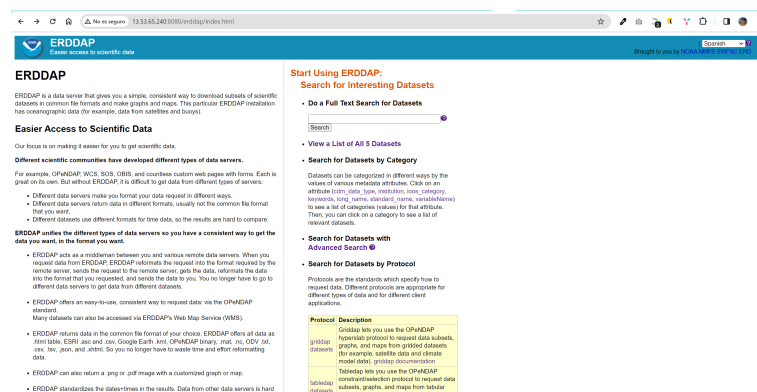


Figura 4.1: Exemplo de pantalla de inicio de ERDDAP.

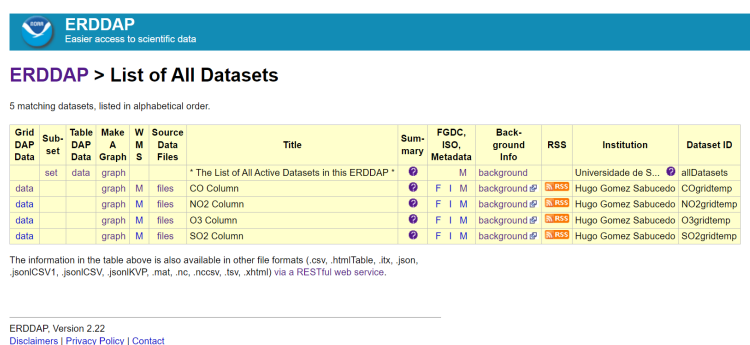


Figura 4.2: Exemplo da lista de datasets dispoñibles.

sible en <http://13.53.65.240:8080/erddap/index.html>. Ó acceder á mesma, preséntasenos a pantalla que se amosa na figura 4.1. Nela, móstrase unha mensaxe co número de datasets dispoñibles, sobre a cal podemos facer clic para ver unha lista cos mesmos, como a que se ve na figura 4.2. Dentro desta lista, premendo en **files**, poderemos ver a pantalla da figura 4.3 que mostra, para o dataset de monóxido de carbono, os arquivos dispoñibles no mesmo, así como a información sobre o propio dataset

Por último, comprobamos o correcto funcionamento da interface web, accesible en <http://13.53.65.240:8080/TROPOMIVisor/>. Ó acceder á mesma, vemos a pantalla da figura B.1, e podemos seleccionar, por unha parte, o parámetro para o que queremos consultar os datos e, por outra, o mes e ano. No caso de que elixamos un mes que non está dispoñible (por exemplo, febreiro de 2024), amosarase a mensaxe de error que se ve na figura 4.4.

Con todas estas probas, puidemos validar o cumprimento dos requisitos funcionais [RF1], [RF2] e [RF3].



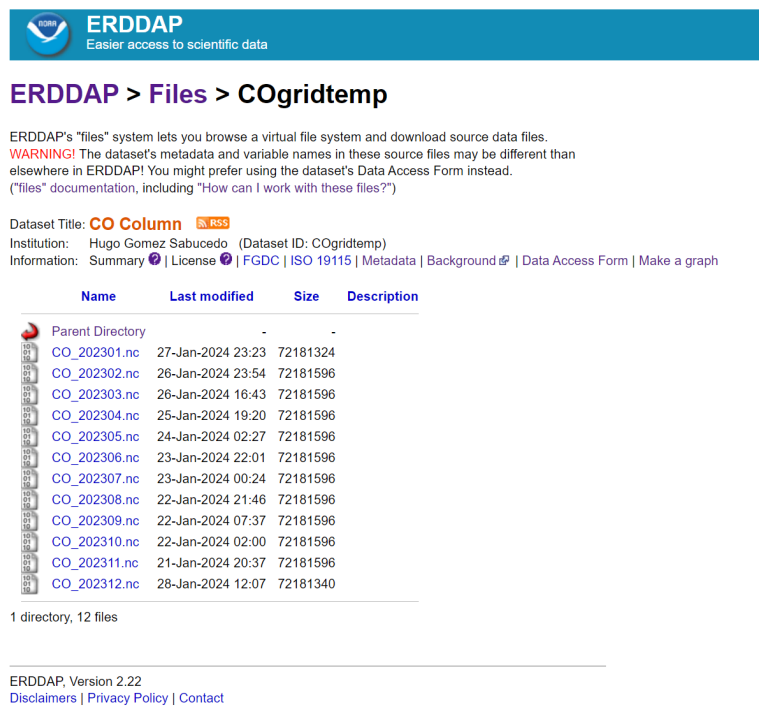


Figura 4.3: Exemplo da información e arquivos dun dataset.

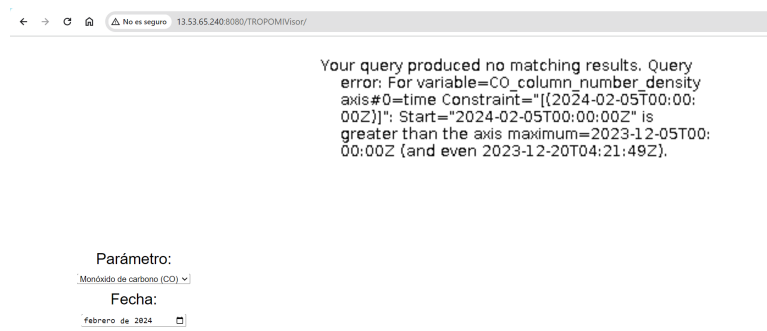


Figura 4.4: Exemplo da mensaxe de erro que se amosa na interface se introducimos un mes non dispoñible.

## 4.2. Probas de integración

Tamén se realizaron probas de integración da aplicación, comprobando o correcto fluxo da mesma, e asegurando que os datos se descargan, procesan e almacenan correctamente, puidendo ser visualizados posteriormente dende a interface. Tamén se comprobou que os cambios nos datasets se reflexaran correctamente. Ademais, comprobouse o correcto funcionamento da interface de usuario, revisando a correcta actualización dos gráficos ante un cambio na selección dos parámetros por parte do usuario.

## 4.3. Probas de rendemento

Por último, realizáronse probas xerais de rendemento, co obxectivo de validar o cumprimento dos requisitos non funcionais [RNF1] e [RNF3]. Así, validouse que a aplicación se execute razoablemente rápido no que a descarga e procesamento de datos se refire. Se ben a descarga dos datos require unha cantidade de tempo elevada, isto é o agardable, posto que estamos a descargar aproximadamente 50GB de datos por produto, o cal non é algo inmediato. O posterior procesamento dos datos si que é moito máis rápido, polo que podemos considerar que se cumpre o requisito de rapidez e eficiencia na aplicación, así coma o do tempo de resposta razoable.

Para verificar o requisito [RNF2], podemos ver canto ocupan finalmente os directorios onde se almacenan os diferentes arquivos agregados. Así, podemos ver que, para cada un dos parámetros, o seu directorio correspondente ocupa arredor de 800MB, o cal, tratándose dos datos de todo un ano, é bastante razoable. Podemos dar por cumprido tamén o requisito de eficiencia no almacenamento.

## Capítulo 5

# Conclusiones e posibles ampliacións

O obxectivo principal deste traballo era deseñar un sistema para a monitorización da variabilidade e tendencia de diferentes parámetros empregados para medir a contaminación do aire, con datos obtidos mediante medicións de satélite. Así, buscábase aproveitar os datos dispoñibles na web de COPERNICUS para traballar con eles, creando agregados mensuais, de forma que os usuarios os puideran empregar para seguir a evolución da contaminación atmosférica.

Este sistema ofrece a posibilidade de consultar, para a Península Ibérica, os valores de catro parámetros diferentes ó longo do ano 2023. Estes parámetros son o monóxido de carbono ( $CO$ ), o ozono ( $O_3$ ), o dióxido de nitróxeno ( $NO_2$ ) e o dióxido de xofre ( $SO_2$ ). Mediante unha interface sinxela, podemos visualizar nun mapa o valor destes parámetros, que se mapearon a unha cuadrícula regular que abarca toda a área de interese.

O usuario ten, desta forma, fácil acceso ós datos de satélite, podendo visualizar os mesmos dunha forma sinxela. Posto que os arquivos individuais de cada un dos parámetros teñen un gran tamaño, mediante a agregación dos mesmos de forma mensual podemos reducir drasticamente as necesidades de almacenamento, o cal resulta realmente beneficioso tendo en conta que, para un ano, poderíamos estar falando de cantidades en torno a 300GB de datos para cada parámetro por ano, mentres que o noso sistema precisa menos de 1GB por parámetro.

Desta forma, podemos considerar que se satisfíxeron os catro obxectivos que se definiron en 1.1. Logrouse mapear a contaminación atmosférica mediante os datos de sensores remotos, mediante a transformación dos datos dispoñibles no dataspace COPERNICUS da ESA e a súa transformación a nivel L3. Conseguimos monitorizar a variabilidade e tendencia dos parámetros relacionados coa contaminación do aire, tanto de forma xeográfica, visualizándoos nun mapa, como a través de gráficos de liñas. En relación con isto, tamén desenvolvimos un sinxelo visor online, para representar a información de forma gráfica e dinámica. Por último, implementouse un sistema interoperable para distribuír os datos,

posto que o propio servidor ERDDAP permite compartir e distribuír os datos con outros sistemas, dunha forma uniforme.

Para rematar, propóñense unha serie de melloras que se poderían aplicar ó sistema para facelo máis completo e adaptable ás necesidades dos usuarios:

1. **Mayor dispoñibilidade de datos.** Os parámetros dispoñibles no dataspaces de COPERNICUS inclúen, ademais dos analizados neste traballo, o metano ( $CH_4$ ), formaldehído ( $HCHO$ ), a fracción de nubes ou o índice UV. Estes parámetros tiveron que descartarse do proxecto por falta de tempo, polo que podería ser interesante incluílos en ampliacións futuras. Asimesmo, os datos están dispoñibles dende mediados do ano 2018, polo que tamén sería factible ampliar a dispoñibilidade temporal dos mesmos, que actualmente é dende xaneiro de 2023 ó presente. Tamén se podería ampliar a área de análise dos datos a todo o continente europeo, ou incluso a todo o mundo.
2. **Capacidade de automatización.** Podería ser útil integrar utilidades de automatización no sistema, de forma que os usuarios puideran recibir, de forma automática, algúns gráficos que eles mesmos elixan segundo teñamos novos produtos dispoñibles, para facilitármolles a análise dos mesmos.
3. **Mellora da experiencia e interface de usuario.** Unha forma de mellora da interface de usuario sería crear unha interface de JavaScript empregando leaflet. Isto non puido levarse a cabo neste proxecto, pero podería ser práctico en caso de aumentar tamén a dispoñibilidade de datos, xa que nos permite unha maior interacción cos mapas. Tamén poderían implementarse visualizacións en forma de gráficos 3D.
4. **Creación dun *dashboard* para visualizar os parámetros en distintas zonas da Península.** Proponse a creación dun taboleiro que permita, a partir dos datos dos que se dispón, centrarse en diferentes áreas urbanas da Península (por exemplo, a área metropolitana de Madrid ou a de Barcelona), para analizar máis en profundidade o valor e a evolución dos diferentes parámetros. Tamén se podería introducir, neste taboleiro, un apartado mediante o cal visualizar as zonas quentes; é dicir, aquelas zonas con valores especialmente altos de diferentes parámetros nun mes en concreto, ou de forma prolongada no tempo.
5. **Integración con Machine Learning.** Como un campo que está actualmente en auge, o machine learning resulta especialmente relevante no ámbito da monitorización ambiental, xa que nos permitiría, a partir dos datos dos que dispoñemos, identificar patróns nos mesmos co obxectivo de elaborar predicións. Desta forma, poderíamos crear, por exemplo, agregacións

semanais no canto de mensuais, o que nos permitiría dispoñer de información máis pormenorizada e realizar mellores análises. Isto podería empregarse en colaboración con diferentes cidades, para saberen cando activar, por exemplo, os seus plans anticontaminación.

6. **Análises estatísticos.** Por último, propónse a incorporación de ferramentas estatísticas para mellorar a experiencia dos usuarios á hora de analizar os datos. Por exemplo, poderían implemantarse algoritmos para a detección de anomalías, ou para realizar directamente análises temporais e seguementos dos cambios dos diferentes parámetros ó longo do tempo.



# Apéndice A

## Manuais técnicos

Este manual técnico explica como realizar a instalación do sistema e o seu mantemento por parte dos administradores.

### A.1. Instalación

No repositorio de GitHub <https://github.com/hugogs02/TROPOMIVisor> dispoñemos de todo o código fonte da aplicación, tanto do backend de Python, coma da interface e os arquivos de configuración do servidor ERDDAP. É preciso recalcar que, no caso de querer instalar o sistema no ordenador propio, este deberá empregar o sistema operativo Ubuntu. Para a súa instalación, é preciso seguir os seguintes pasos:

1. **Instalar Python.** O sistema require **Python 3.11.6**, o cal se instalará mediante Anaconda, para facilitar tamén a xestión e instalación de paquetes. Para instalar Anaconda, debemos descargar o instalador dispoñible en <https://www.anaconda.com/download#downloads>. Deberase dar permiso de execución (`chmod o+rx [filename]`). Será necesario seguir os pasos que indica o instalador, entre os que se inclúen elixir o directorio de instalación, engadilo ó PATH e inicializar a instalación co comando `conda init`. A continuación, instalaremos os seguintes paquetes:

- calendar (vén preinstalado con Python).
- datetime (versión 5.4).
- dateutil.parser (versión 2.8.2).
- glob2 (versión 0.7).
- harp (versión 1.17). Instalado con conda-forge.
- os (vén preinstalado con Python).
- pandas (versión 2.1.4).

- requests (versión 2.31.0).
  - shutil (vén preinstalado con Python).
  - sys (vén preinstalado con Python).
  - time (vén preinstalado con Python).
  - zipfile36 (versión 0.1.3).
2. **Instalar Java.** A instalación de Java debe ser da versión 17 (ou superior). Instalar Java en Ubuntu é sinxelo, simplemente se debe executar o comando `apt install openjdk-17-jdk openjdk-17-jre`.
  3. **Configurar Tomcat.** Para configurar correctamente tomcat, débese crear un usuario tomcat (con `sudo useradd -m -d /opt/tomcat -U -s /bin/false tomcat`), e descargar o arquivo de Tomcat (versión 10) dende <https://tomcat.apache.org/download-10.cgi>. Este deberá ser descomprimido no directorio `opt/tomcat`. Accederase ó arquivo `/opt/tomcat/conf/tomcat-users.xml` para configurar os usuarios Manager e Host Manager. A continuación, creárase un servizo para Tomcat, mediante o comando `sudo nano /etc/systemd/system/tomcat.service`, engadindo as seguintes liñas:

```

1      [ Unit ]
2      Description=Tomcat
3      After=network.target
4
5      [ Service ]
6      Type=forking
7
8      User=tomcat
9      Group=tomcat
10
11     Environment="JAVA_HOME=/usr/lib/jvm/java-1.11.0-
        openjdk-amd64"
12     Environment="JAVA_OPTS=-Djava.security.egd=file
        ://dev/urandom"
13     Environment="CATALINA_BASE=/opt/tomcat"
14     Environment="CATALINA_HOME=/opt/tomcat"
15     Environment="CATALINA_PID=/opt/tomcat/temp/tomcat.
        pid"
16     Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -
        server -XX:+UseParallelGC"
17
18     ExecStart=/opt/tomcat/bin/startup.sh
19     ExecStop=/opt/tomcat/bin/shutdown.sh
20
```



```

21      RestartSec=10
22      Restart=always
23
24      [ Install ]
25      WantedBy=multi-user.target

```

A continuación, recargarase o *systemctl* con `sudo systemctl daemon-reload` e arrancarase o servizo de Tomcat con `sudo systemctl start tomcat`. Configurarase Tomcat para que arranque co sistema, mediante o comando `sudo systemctl enable tomcat`. Por último, executarase `sudo ufw allow 8080` para permitir tráfico polo porto 8080.

4. **Executar os scripts de Python para obtención de datos.** Empregando git, podemos descargar o código dispoñible no repositorio que indicamos anteriormente. Co código descargado, podemos executar o script principal para descargar os datos. O formato de execución será: `python app.py produto data_inicio data_fin ruta_descarga ruta_procesado`. “produto” deberá ser un entre os dispoñibles (L2\_CO\_\_\_\_, L2\_NO2\_\_\_\_, L2\_O3\_\_\_\_, L2\_SO2\_\_\_\_), e a data deberá ter o formato AAAA-MM-DD
5. **Configurar o servidor ERDDAP.** Para configurar o servidor ERDDAP, deberemos seguir as instrucións de <https://coastwatch.pfeg.noaa.gov/erddap/download/setup.html>, empregando os arquivos de setup.xml e datasets.xml que se baixaron do repositorio de GitHub.
6. **Configurar a interface web.** Para a interface web, crearase unha aplicación nova de Tomcat. Isto farase creando unha nova carpeta no directorio `/opt/tomcat/webapps/`, e copiando á mesma o arquivo HTML que se baixou do repositorio.

Unha vez seguidos todos estes pasos, o sistema estará listo para o seu uso.

## A.2. Mantemento

O mantemento do sistema é sinxelo, pois o administrador só precisa consultar o log de erros o día que se indicou no *job* de cron para executar a actualización dos datos, e comprobar que funcionou correctamente. En caso de que se rexistre algunha mensaxe de erro relativa á descarga dos arquivos, se esta se dá nun ou dous deles, pode ignorarse, posto que non afectara ó resultado. En caso de que se produzan máis erros, será preciso executar o script principal manualmente, para realizar a descarga de datos. O mesmo será preciso en caso de ter algún erro coa importación e transformación dos datos.

Tamén poden ocorrer erros ou problemas co servidor ERDDAP. Neste caso, o mellor é reiniciar Tomcat, empregando para iso os scripts `shutdown.sh` e `startup.sh`, ubicados en `/opt/tomcat/bin/`.



# Apéndice B

## Manuais de usuario

O manual de usuario proporciona a información necesaria para o usuario final que desexa empregar o sistema de forma autónoma; é dicir, que o usuario non precise ler outro manual para empregar a aplicación.

Para acceder á aplicación, o usuario debe acceder á seguinte URL: <http://13.53.65.240:8080/TROPOMIVisor/>, podendo ver a interface da figura B.1. Nela, preséntaselle un desplegable para elixir o parámetro, entre os 4 que hai dispoñibles (CO, SO<sub>2</sub>, NO<sub>2</sub> ou O<sub>3</sub>); e un calendario dende o que pode elixir o mes do que quere consultar os datos. Ó realizar un cambio en calquera destes dous parámetros, actualízase automaticamente o mapa, amosándose o gráfico correspondente. O usuario deberá ter en conta o rango de datas dispoñible para consultar os datos, que neste traballo é a partir de xaneiro de 2023. Ademais, deberá ter en conta que os datos do mes anterior (por exemplo, de xaneiro de 2024) non estarán dispoñibles ata aproximadamente o día 11 do seguinte mes (neste caso, o 11 de febreiro), debido á latencia á hora de procesar os datos tanto no servidor local como por parte de COPERNICUS. Se intenta visualizar o mapa para un mes no que non hai datos, a imaxe mostrará unha mensaxe de erro.

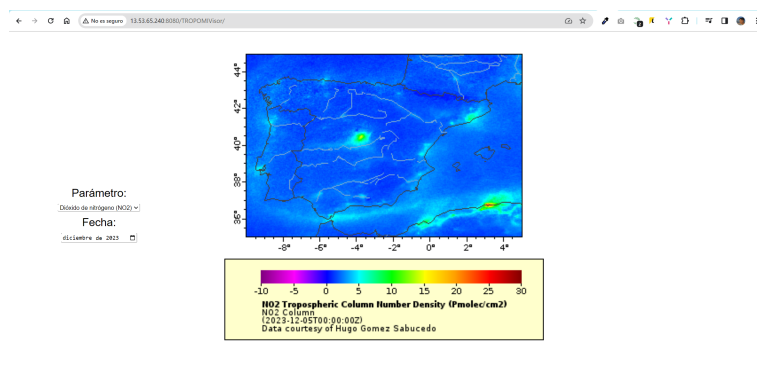


Figura B.1: Exemplo da interface que se mostra ó acceder á aplicación.



# Bibliografía

- [1] Data Products - Sentinel-5P Mission - Sentinel Online. *Sentinel Online*. [en línea] [sen data] Disponible en <https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-5p/data-products>.
- [2] ERDDAP - Home Page. *National Centers for Environmental Information (NCEI)*. [en línea] [sen data] Disponible en <https://www.ncei.noaa.gov/erddap/index.html>
- [3] Set Up Your Own ERDDAP. *ERDDAP*. [en línea] [sen data] Disponible en <https://coastwatch.pfeg.noaa.gov/erddap/download/setup.html>
- [4] Working with the datasets.xml File. *ERDDAP*. [en línea] [sen data] Disponible en <https://coastwatch.pfeg.noaa.gov/erddap/download/setupDatasetsXml.html>
- [5] ARSET - High Resolution NO2 Monitoring From Space with TROPOMI. *NASA Applied Sciences*. [en línea] [sen data] Disponible en <https://appliedsciences.nasa.gov/get-involved/training/english/arset-high-resolution-no2-monitoring-space-tropomi>
- [6] ARSET - Data Analysis Tools for High Resolution Air Quality Satellite Datasets *NASA Applied Sciences*. [en línea] [sen data] Disponible en <https://appliedsciences.nasa.gov/get-involved/training/english/arset-data-analysis-tools-high-resolution-air-quality-satellite>
- [7] ARSET - Applications of Satellite Observations for Air Quality and Health Exposure *NASA Applied Sciences*. [en línea] [sen data] Disponible en <https://appliedsciences.nasa.gov/get-involved/training/english/arset-applications-satellite-observations-air-quality-and-health>
- [8] Sentinel-5P L2. *Sentinel Hub*. [en línea] [sen data] Disponible en <https://docs.sentinel-hub.com/api/latest/data/sentinel-5p-l2/>
- [9] S5P Mission. *SentiWiki Home*. [en línea] [sen data] Disponible en [https://sentiwiki.copernicus.eu/web/s5p-mission#TROPOMI-Response-Functions-\(Acquisition-Modes\)](https://sentiwiki.copernicus.eu/web/s5p-mission#TROPOMI-Response-Functions-(Acquisition-Modes))

- [10] Types of air pollution. Which pollution is the most dangerous? — Air Quality Monitoring. Monitor in UK & Europe. Airly Data Platform and Monitors. *Air Quality Monitoring. Monitor in UK & Europe. Airly Data Platform and Monitors — Airly.* [en línea] [sen data] Disponible en <https://airly.org/en/the-most-dangerous-types-of-air-pollution/>
- [11] Documentation - OData. *Copernicus Dataspace* [en línea] [sen data] Disponible en: <https://documentation.dataspace.copernicus.eu/APIs/OData.html>
- [12] *Documentation - Quotas and Limitations.* Documentation [en línea] [sen data] Disponible en: <https://documentation.dataspace.copernicus.eu/Quotas.html>
- [13] HARP 1.20.2 documentation *HARP manual* [en línea] [sen data] Disponible en: <https://stcorp.github.io/harp/doc/html/index.html>
- [14] S5P\_L2\_CO — HARP 1.20.2 documentation. *HARP manual* [en línea] [sen data] Disponible en: [https://stcorp.github.io/harp/doc/html/ingestions/S5P\\_L2\\_CO.html](https://stcorp.github.io/harp/doc/html/ingestions/S5P_L2_CO.html)
- [15] S5P\_L2\_NO2 — HARP 1.20.2 documentation. *HARP manual* [en línea] [sen data] Disponible en: [https://stcorp.github.io/harp/doc/html/ingestions/S5P\\_L2\\_NO2.html](https://stcorp.github.io/harp/doc/html/ingestions/S5P_L2_NO2.html)
- [16] S5P\_L2\_O3 — HARP 1.20.2 documentation. *HARP manual* [en línea] [sen data] Disponible en: [https://stcorp.github.io/harp/doc/html/ingestions/S5P\\_L2\\_O3.html](https://stcorp.github.io/harp/doc/html/ingestions/S5P_L2_O3.html)
- [17] S5P\_L2\_SO2 — HARP 1.20.2 documentation. *HARP manual* ([en línea] [sen data] Disponible en: [https://stcorp.github.io/harp/doc/html/ingestions/S5P\\_L2\\_SO2.html](https://stcorp.github.io/harp/doc/html/ingestions/S5P_L2_SO2.html)
- [18] Sentinel-5P TROPOMI Level 2 Product User Manual - Carbon Monoxide. *Sentinel Online* [en línea] [sen data] Disponible en: [https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset\\_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-carbon-monoxide](https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-carbon-monoxide)
- [19] Sentinel-5P TROPOMI Level-2 Product User Manual - Nitrogen Dioxide. *Sentinel Online* [en línea] [sen data] Disponible en: [https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset\\_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-nitrogen-dioxide](https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-nitrogen-dioxide)

- [20] Sentinel-5P TROPOMI Level 2 Product User Manual - Ozone Total Column. *Sentinel Online* [en línea] [sen data] Disponible en: [https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset\\_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-ozone-total-column](https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-ozone-total-column)
- [21] Sentinel-5P TROPOMI Level 2 Product User Manual - Sulphur Dioxide SO<sub>2</sub>. *Sentinel Online* [en línea] [sen data] Disponible en: [https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset\\_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-sulphur-dioxide-so2](https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-5p-tropomi/document-library/-/asset_publisher/w9Mnd6VPjXlc/content/sentinel-5p-tropomi-level-2-product-user-manual-sulphur-dioxide-so2)