UNIVERSIDADE DE SANTIAGO DE COMPOSTELA


ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

# Seguimento de contaminación de cidades 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 *(Seguimento de contaminación decidades 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 titoría 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 (Data):

Titor/a, Alumno/a,

(Joaquín Ángel Triñanes Fernández) (Hugo Gómez Sabucedo)

# Agradecementos

Se se quere pór algún agradecemento, este vai aquí.

# Resumo

Breve resumo das principais contribucións do traballo.

# Índice xeral

# Índice de figuras

# Índice de cadros

# 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.

Todos estes axentes afectan non só ó medio ambiente, senón ós seres humanos. Por iso, monitorizar a súa evolución é algo crucial, especialmente en entornos especialmente críticos, coma poden ser as cidades, nas cales a conxunción dunha gran densidade de poboación e movementos en áreas, polo xeral, pequenas, fai que sexa crucial coñecer cal é o nivel destes axentes contaminantes. Desta forma, as autoridades poden adoptar medidas, tanto no eido da saúde pública, para previr enfermidades asociadas a unha elevada presenza de ditos contaminantes, como de proteción do medio ambiente, para evitar que se causen danos ó entorno no que vivimos.

Así, dende a Axencia Espacial Europea (ESA) lanzouse, no ano 2017, un satélite de observación terreste nomeado Sentinel-5 Precursor (ou Sentinel-5P). Dito satélite, que se enmarca dentro do Programa Copernico, 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 propor-

ciona 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. Construíndo unha query, e empregando a librería `requests`, descagaremos 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 posteriomente 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 empregando `leaflet`, unha libraría de código aberto de `JavaScript` que permite crear mapas web de forma sinxela. Isto permitirá consultar a evolución de cada un dos diferentes parámetros (escollendo un deles) ó longo do tempo.

Todo isto implementarémolo 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 cumplir 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 obexectivo 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 asumiráo 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 periodo 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

## 2.3. Requisitos do produto

## 2.4. Requisitos de rendemento

# 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 ????¿¿¿¿?¿MÓDULO??? `cron` de Ubuntu, definiremos unha tarefa a executarse automáticamente 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 e JavaScript. 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, **?¿?¿?¿?¿aqui comentar movidas de copernicus e api**.

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

#### Script principal e automatización

or unha parte, temos o script principal, `app.py`. Este é o script dende o que se executa ?¿?¿?¿?¿?¿. Nel, establécense os parámetros cos cales se executaran 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).

En segundo lugar, temos o script de descarga de datos.

Por outra parte, temos o script que se encarga de procesar os datos. Nel, o módulo principal que empregamos é HARP que, como se explica en **??**, 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 POLLUTANT. 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 centenares de arquivos cada mes, con tamaños de entre 500MB e 1GB, os cales, unha vez procesados, non son de interese nin utilidade).

Malia que cada unha das funcións de procesado

Por último, 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 **??**??¿¿¿?, 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.

### 3.1.2. Middleware

### 3.1.3. Frontend

A descrición destes compoñentes, xunto coas funcionalidades que desenvolven cada un dos arquivos, explícase máis en detalle no anexo A.

## 3.2. Tecnoloxías empregadas

### 3.2.1. Anaconda

### 3.2.2. HARP

## 3.3. Desenvolvemento do sistema

## 3.4. Implementación do sistema

# Capítulo 4

# Probas

# Capítulo 5

# Conclusións e posibles ampliacións

# Apéndice A

# Manuais técnicos

En función do tipo de Traballo e metodoloxía empregada, o contido poderase dividir en varios documentos. En todo caso, neles incluirase toda a información precisa para aquelas persoas que se vaian encargar do desenvolvemento e/ou modificación do Sistema (por exemplo código fonte, recursos necesarios, operacións necesarias para modificacións e probas, posibles problemas, etc.). O código fonte poderase entregar en soporte informático en formatos PDF ou postscript.

# Apéndice B

# Manuais de usuario

Incluirán toda a información precisa para aquelas persoas que utilicen o Sistema: instalación, utilización, configuración, mensaxes de erro, etc. A documentación do usuario debe ser autocontida, é dicir, para o seu entendemento o usuario final non debe precisar da lectura doutro manual técnico.

# Bibliografía


[1] Nvidia CUDA programming guide. Versión 2.0, 2010. Dispoñible en *http://www.nvidia.com*.

[2] Acceso múltiple por división de código. Artigo da wikipedia (*http://es.wikipedia.org*). Consultado o 2 de xaneiro do 2010.

[3] R.C. Gonzalez e R.E. Woods, *Digital image processing*, 3ª edición, Prentice Hall, New York, 2007.

[4] P. González, J.C. Cartex e T.F. Pelas, "Parallel computation of wavelet transforms using the lifting scheme", *Journal of Supercomputing*, vol. 18, no. 4, pp. 141-152, junio 2001.