**Análisis del tráfico en la ciudad de valencia y su relación con la contaminación atmosférica. Almacenamiento de la información en MongoDB y consultas**

* **Nombre:** Alejandro Perez Peñalver

Paso 1. El alumno deberá pensar, en primer lugar, la estructura de los documentos que debe generar en la base de datos para guardar la información (tanto para los datos de tráfico como para los datos de las estaciones meteorológicas), esta estructura deberá optimizar la información lo máximo posible de manera que la base de datos no duplique información. Se recomienda consultar con el profesor la posible solución adoptada por cada alumno antes de empezar a materializar la base de datos. Se recomienda revisar la estructura de los datos para detectar alguna inconsistencia en los mismos (por ejemplo, puede haber identificadores duplicados, puntos sin coordenadas, ficheros vacíos, replicados…).

**Documento de trafico**

# agregación para limitar el arreglo 'valores' a solo un elemento en el primer documento

documents = db.trafico.aggregate([

{

"$project": {

"valores": { "$slice": ["$valores", 1] }, # se limita el arreglo 'valores' a un solo elemento

"id\_tramo":1, # Se agrega el campo '\_id'

"coordenadas":1,

}

},

{

"$limit": 1 # Se Limita los resultados a solo el primer documento

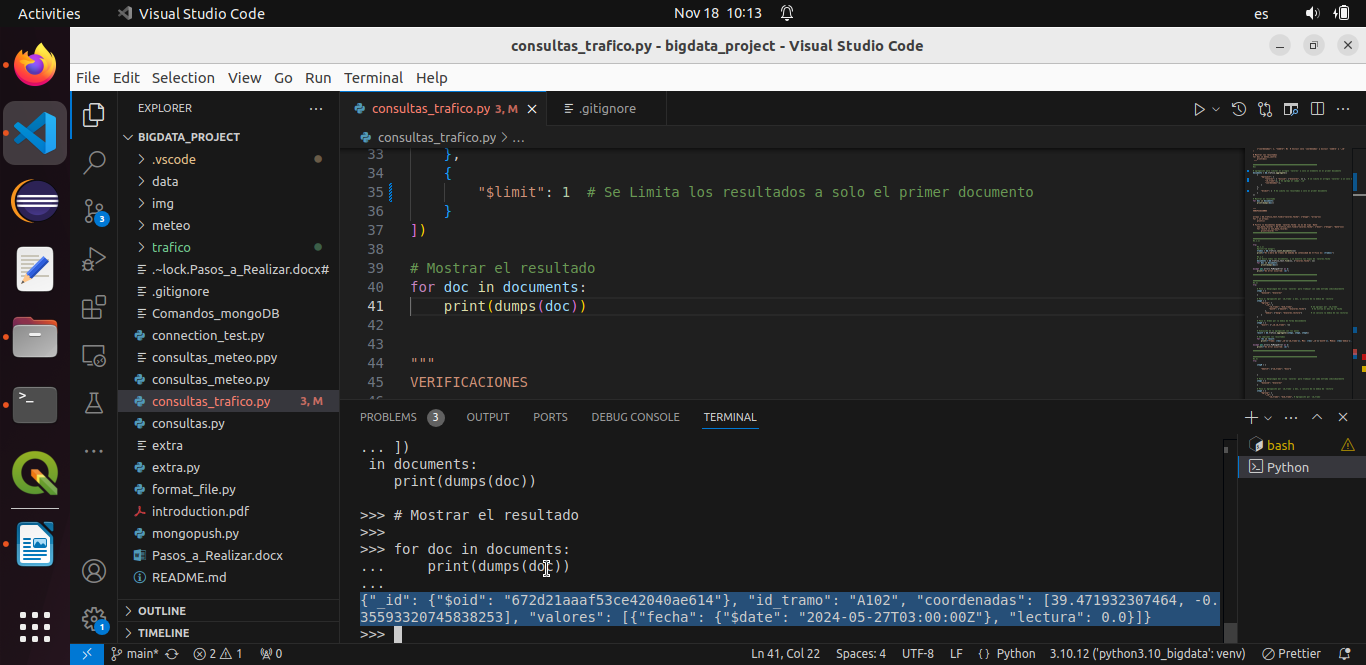
}

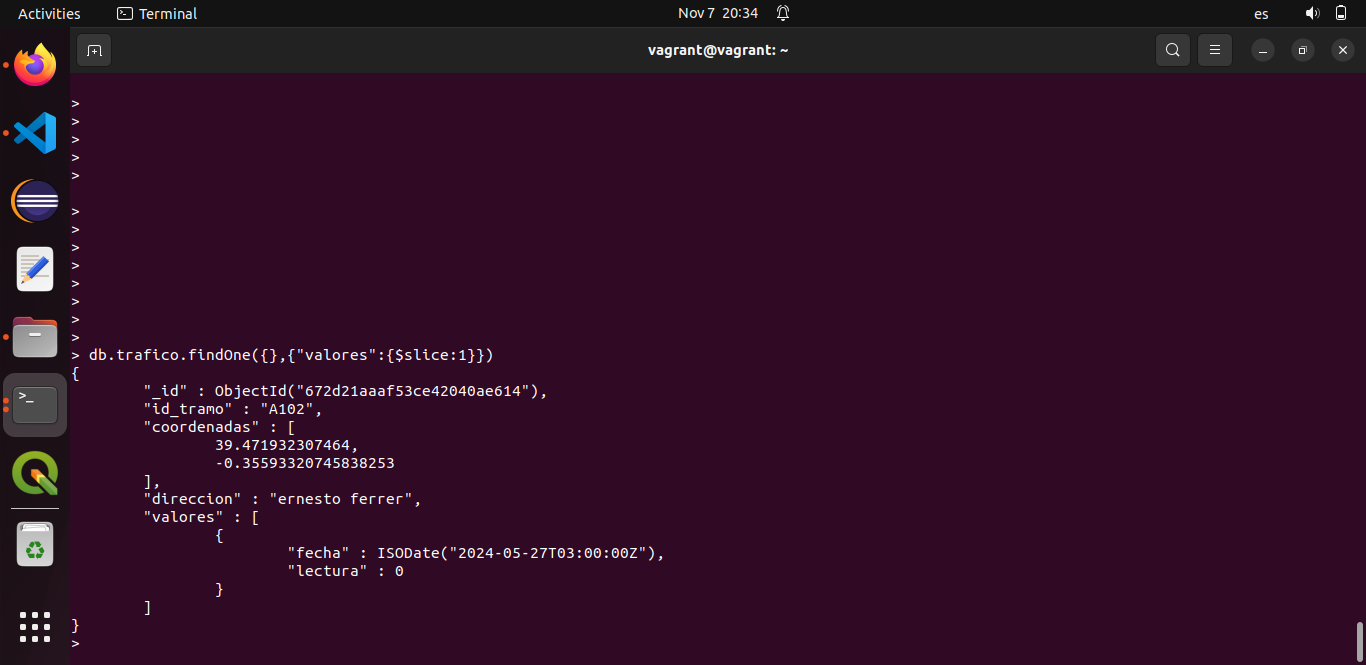
])

# Mostrar el resultado

for doc in documents:

print(dumps(doc))





**Documento de estaciones meteorológicas**

**Documentos de estaciones**

# agregación para limitar el arreglo 'valores' a solo un elemento en el primer documento

documents = db.trafico.aggregate([

{

"$project": {

"valores": { "$slice": ["$valores", 1] }, # se limita el arreglo 'valores' a un solo elemento

"id\_tramo":1, # Se agrega el campo '\_id'

"coordenadas":1,

}

},

{

"$limit": 1 # Se Limita los resultados a solo el primer documento

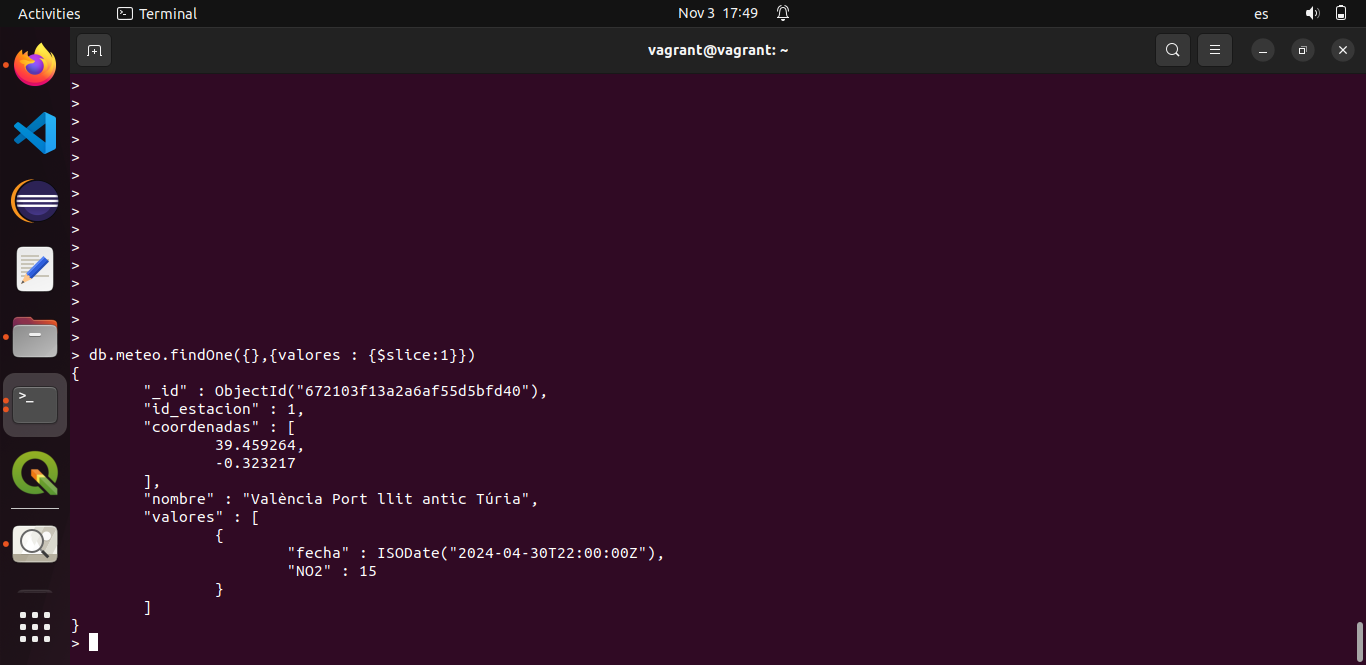
}

])

# Mostrar el resultado

for doc in documents:

print(dumps(doc))

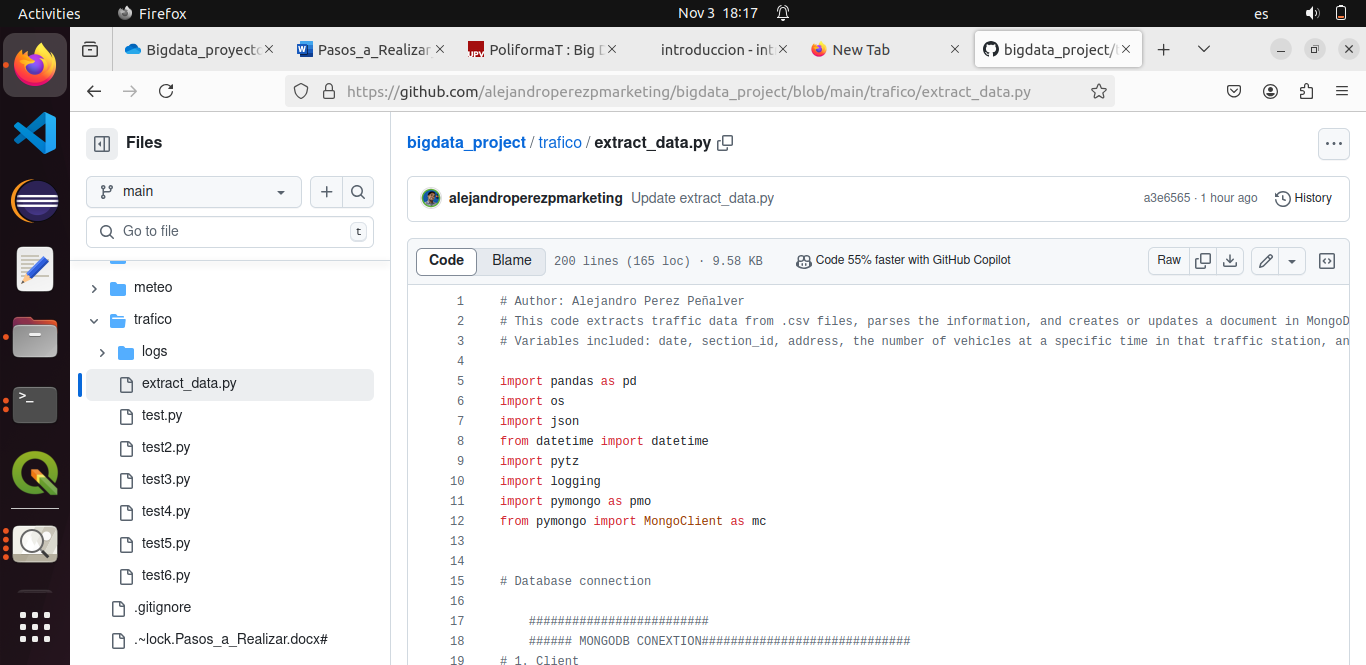


**Almacenamiento de los ficheros de tráfico (1 punto).** Se deberá realizar un programa en Python que lea los diferentes .csv originales del tráfico y los cargue en la base de datos. El documento en mongoDB debe contener el identificador el punto, la dirección, las coordenadas y los valores de lectura junto con la fecha.

Para la lectura y carga de la fecha se debe usar la librería datetime, de manera que se cargue la fecha en MongoDB en un formato reconocible por la base de datos. En caso de que la lectura sea de -1 o mayor de 5000 (datos erróneos) asignar un None a esa lectura. Tener en cuenta también la variable “estado” para cargar únicamente aquellas medidas válidas.

Los ficheros pueden presentar algún otro tipo de error o defecto en sus datos “confusión” por lo que el alumno debe tener en cuenta este tipo de errores o defectos a nivel de programación para generar una base de datos lo más consistente posible.

* **Código url:** https://github.com/alejandroperezpmarketing/bigdata\_project/blob/main/trafico/extract\_data.py

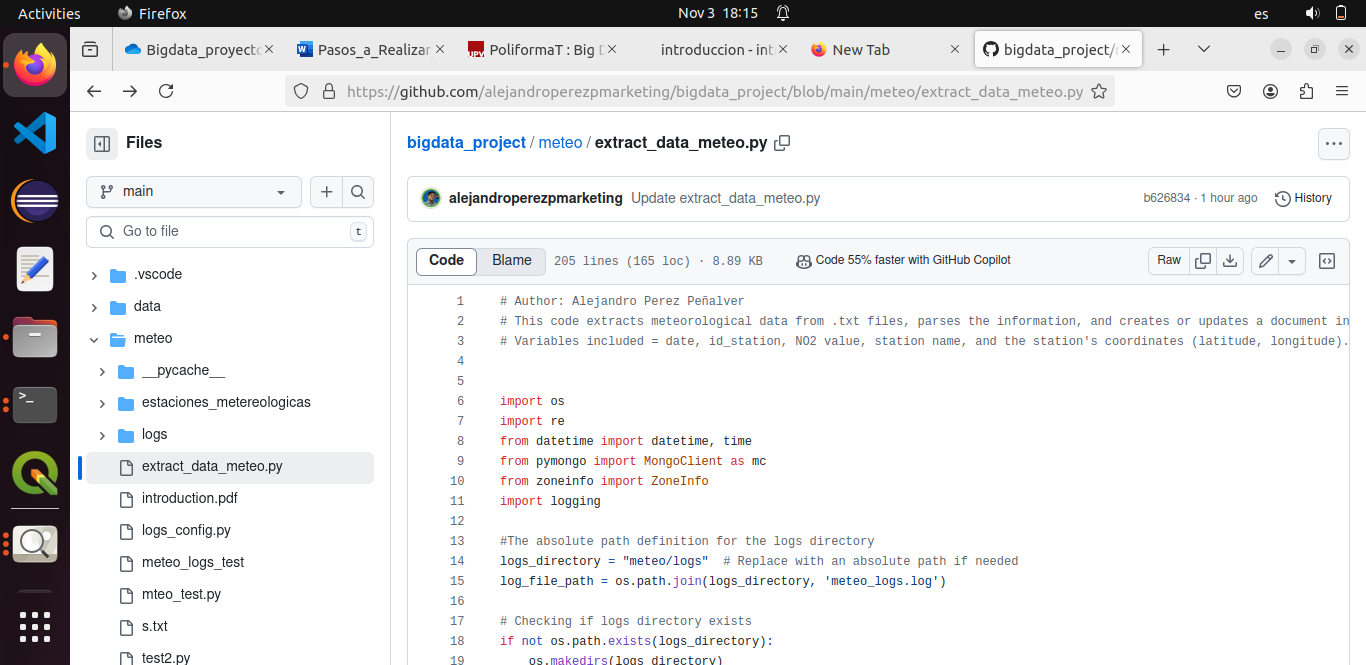


**Almacenamiento de las estaciones atmosféricas (1 punto):** Realizar un programa en Python para leer los ficheros .csv de cada estación meteorológica e importar la información a la base de datos generando una nueva colección. En este caso, el programa se deberá ejecutar una vez por cada estación meteorológica y deberá rescatar la información de NO2 únicamente para el mes de mayo y solo de aquellas estaciones que cuenten con esta información.

El programa debe buscar de forma automática la posición del dióxido de nitrógeno (NO2) dentro de la cabecera para cargar únicamente este valor en la base de datos.

En los documentos de la colección que generan estos ficheros deberán estar incorporadas las coordenadas de cada estación.

* **Código url**: https://github.com/alejandroperezpmarketing/bigdata\_project/blob/main/meteo/extract\_data\_meteo.py



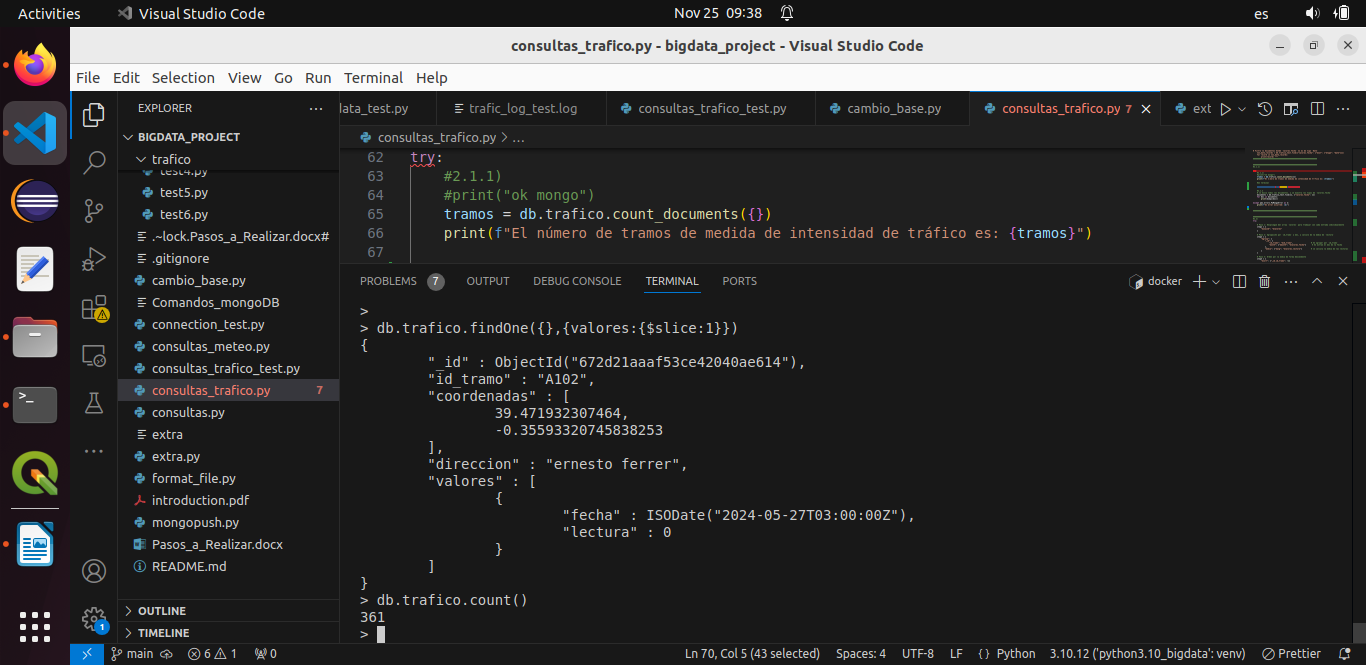
**Paso 2.** El alumno deberá contestar a las siguientes preguntas haciendo un copiar y pegar tanto de la consulta como del resultado obtenido para completar esta parte del proyecto.

**2.1 (0.5 puntos)** ¿Cuántos tramos de medida de intensidad de tráfico hay? ¿Cuál es la estructura del documento (usa la orden findOne() añadiendo el comando $slice en caso de que algún campo contenga mucha información)?

Cantidad de tramos de medida de intensidad de tráfico

#en terminal

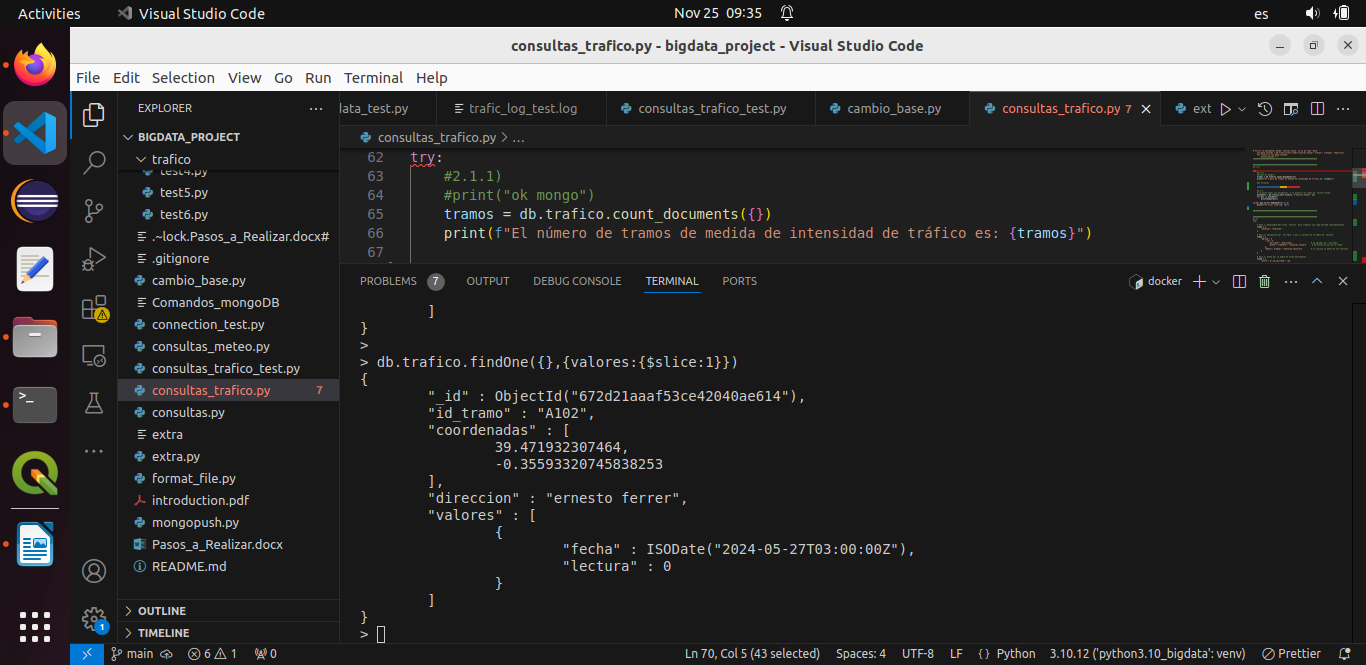
db.trafico.count()



Estructura del documento

# en terminal

db.trafico.findOne({},{valores:{$slice:1}})



import pymongo as pmo

from pymongo import MongoClient as mc

# Database connection

#########################

###### MONGODB CONEXTION#############################

# 1. Client

#client = mc('mongodb://vagrant:vagrant@localhost:27017/bigdata')

client = mc('mongodb://vagrant:vagrant@localhost:27017/bigdata?authSource=admin')

db = client.bigdata

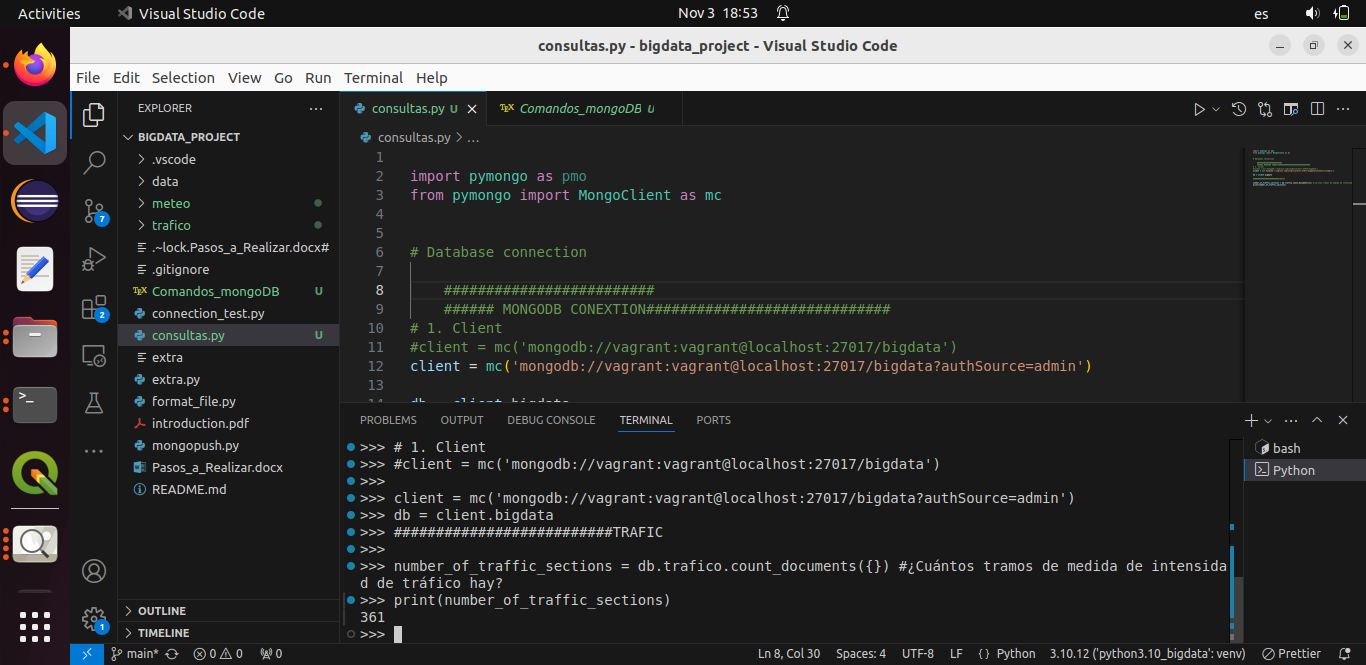
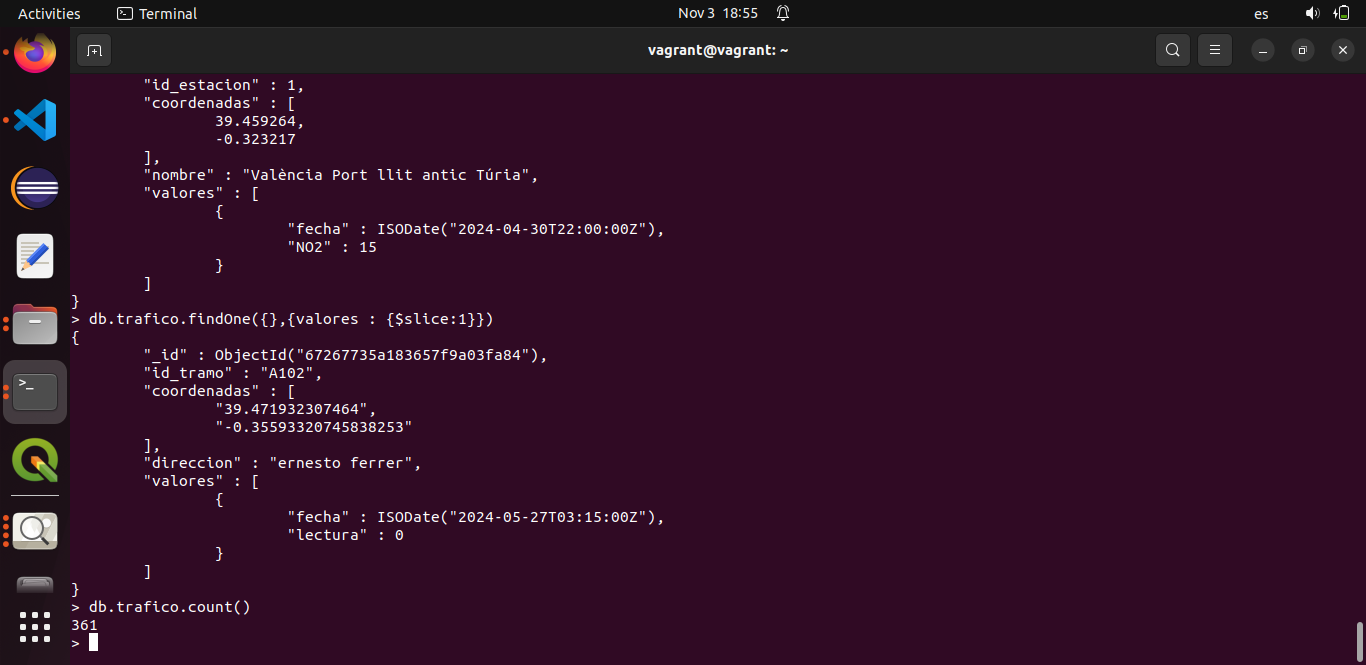
##########################TRAFIC

number\_of\_traffic\_sections = db.trafico.count\_documents({}) #¿Cuántos tramos de medida de intensidad de tráfico hay?

print(number\_of\_traffic\_sections)

#en terminal

db.trafico.count()



**Resultado: 361 documentos.**

**2.2 (1 punto).** Obtener el valor mensual medio de intensidad de tráfico para cada tramo, ordenando la salida de mayor a menor valor de intensidad de tráfico (operador $sort de la función aggregate). Como ejemplo, para comprobar la solución, para el tramo “A122” el resultado es 2420.4.

#### en terminal

db.trafico.aggregate([

{$unwind: "$valores"},

{$group: {

\_id: {

month: {$month:"$valores.fecha"},

id\_tramo: "$id\_tramo"

},

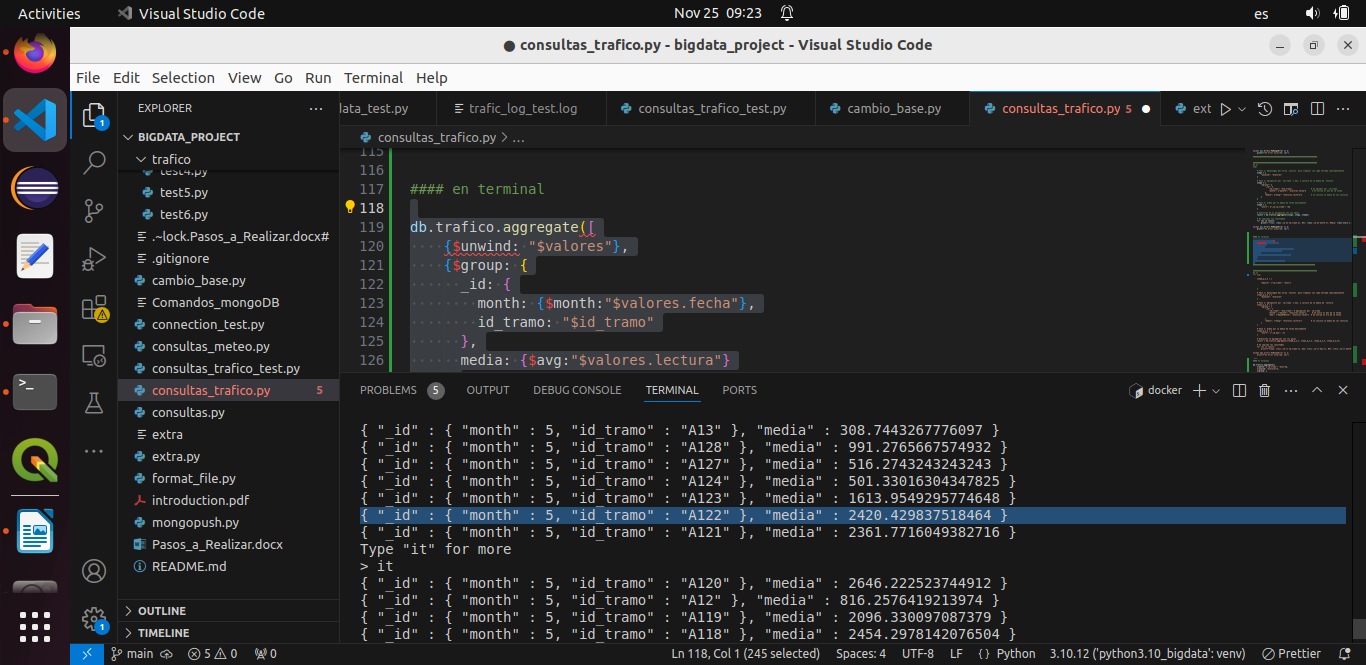
media: {$avg:"$valores.lectura"}

}

},

{$sort: {"\_id.id\_tramo":-1}}

])



**2.3 (0.5 puntos).** Obtener el valor medio de tráfico por día del mes para la espira con identificador igual a “A111”, ordenando la salida por día del mes. Como ejemplo, para comprobar la solución, para el primer día del mes el resultado es 975.7.

#2.3)

try:

step0 = {

"$match": {"id\_tramo": "A111"}

}

# Paso 1: Despliegue del array `valores` para trabajar con cada entrada individualmente

step1 = {

"$unwind": "$valores"

}

# Paso 2: Agrupación por `id\_tramo` y mes, y calculo de la media de `lectura`

step2 = {

"$group": {

"\_id": {

"id\_tramo": "$id\_tramo", # Agrupación por `id\_tramo`

"month": {"$month": "$valores.fecha"}, # Se extra el mes de la fecha

"day": {"$dayOfMonth": "$valores.fecha"} # Se extrae el día de la fecha

},

"media": {"$avg": "$valores.lectura"} # se calcula la media de las lecturas

}

}

# Paso 3: Orden por la media de forma descendente

step3 = {

"$sort": {"\_id.day": -1}

}

# Ejecución la agregación con los pasos

result = db.trafico.aggregate([step0, step1, step2, step3])

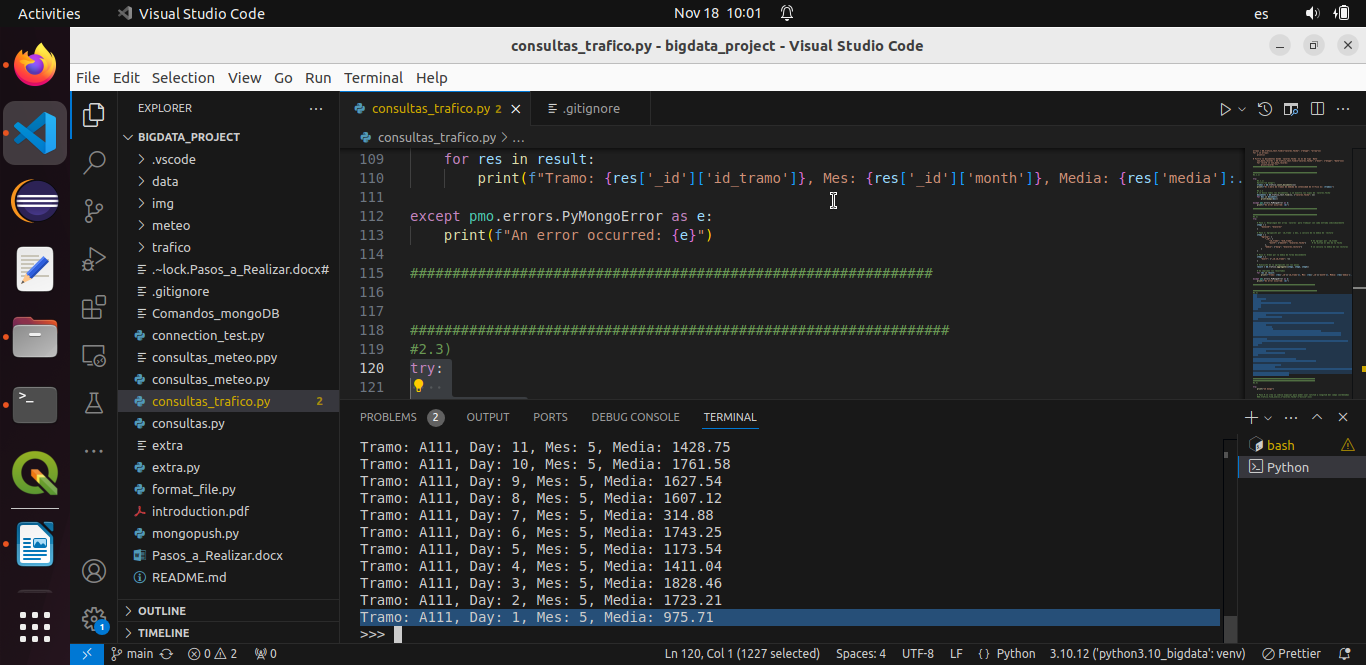
# Se imprime los resultados

for res in result:

print(f"Tramo: {res['\_id']['id\_tramo']}, Day: {res['\_id']['day']}, Mes: {res['\_id']['month']}, Media: {res['media']:.2f}")

except pmo.errors.PyMongoError as e:

print(f"An error occurred: {e}")



#### en terminal

db.trafico.aggregate([

{$match: {"id\_tramo": "A111"}},

{$unwind: "$valores"},

{$group: {

\_id: {

month: {$month:"$valores.fecha"},

day: {$dayOfMonth: "$valores.fecha"}

},

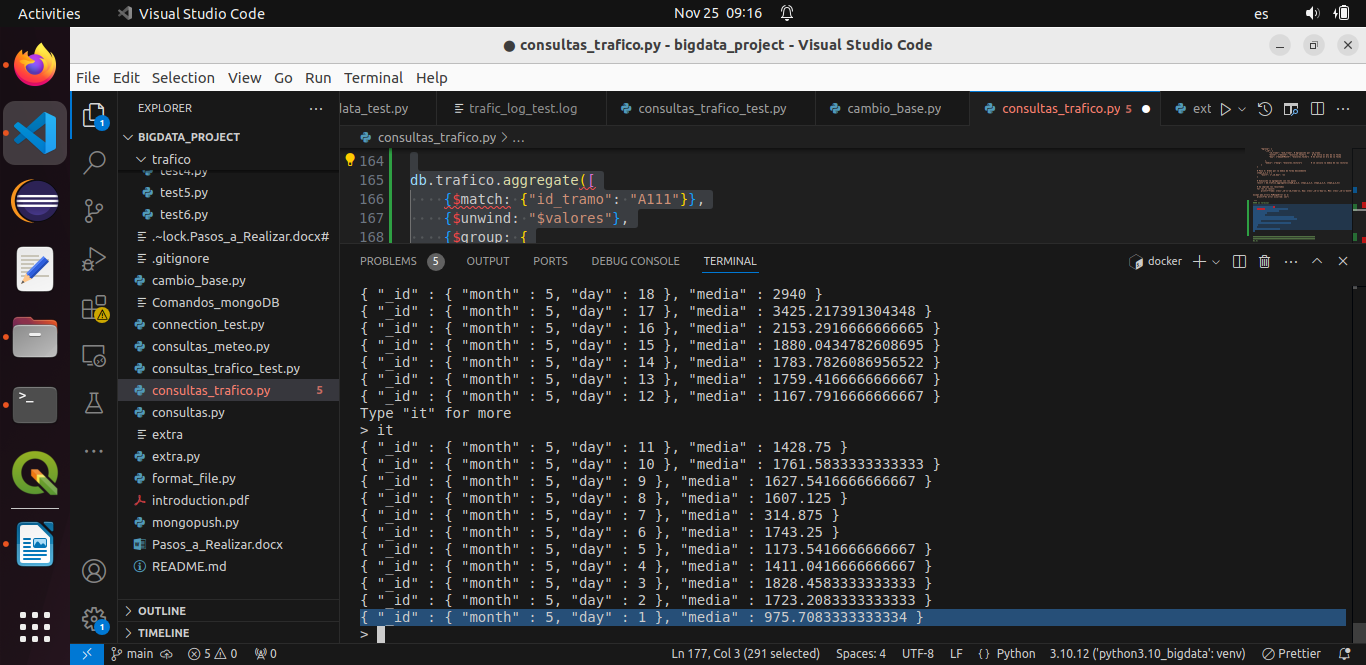
media: {$avg:"$valores.lectura"}

}

},

{$sort: {"\_id.day":-1}}

])



**2.4 (1.5 puntos).** Obtener un .csv donde se guarde el valor medio de tráfico para cada día del mes teniendo en cuenta los datos de tráfico de cada espira electromagnética cuyas coordenadas se encuentren dentro de un radio de 600 metros a partir de las coordenadas de la estación meteorológica Centre. Para ello se deben generar un índice “2dsphere” y utilizar la función $geoNear del método $aggregate para realizar la consulta geoespacial. A continuación, se deberá realizar la consulta guardando el resultado en una nueva colección, esta nueva colección se deberá exportar al correspondiente fichero csv y el resultado se deberá dibujar en una gráfica (usando LibreOffice, por ejemplo, dentro de la máquina virtual). Como ejemplo, para comprobar la solución, para el primer día del mes el resultado es 162.1.

### en terminal

db.trafico.createIndex({"coordenadas":"2dsphere"}) # creacion del indice

db.trafico.aggregate([

{$geoNear: {near: {"coordinates": [39.470718,-0.376384]},

distanceField: "DistanciaCalculada",

maxDistance: 600}},

{$unwind: "$valores"},

{$group: {

\_id: {

month: {$month:"$valores.fecha"},

day: {$dayOfMonth: "$valores.fecha"}

},

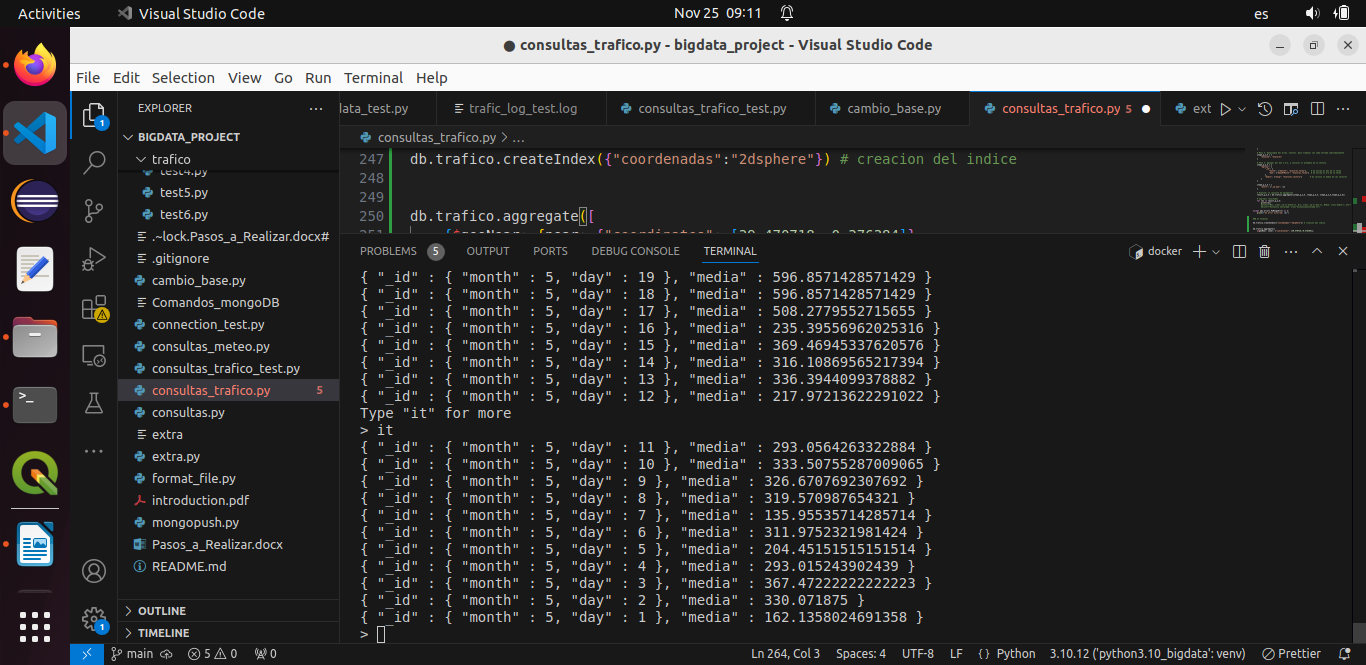
media: {$avg:"$valores.lectura"}

}

},

{$sort: {"\_id.day":-1}}

])



**2.5 (0.5 puntos)** Ordenar las estaciones meteorológicas por latitud de mayor a menor.

#en terminal

db.meteo.find(

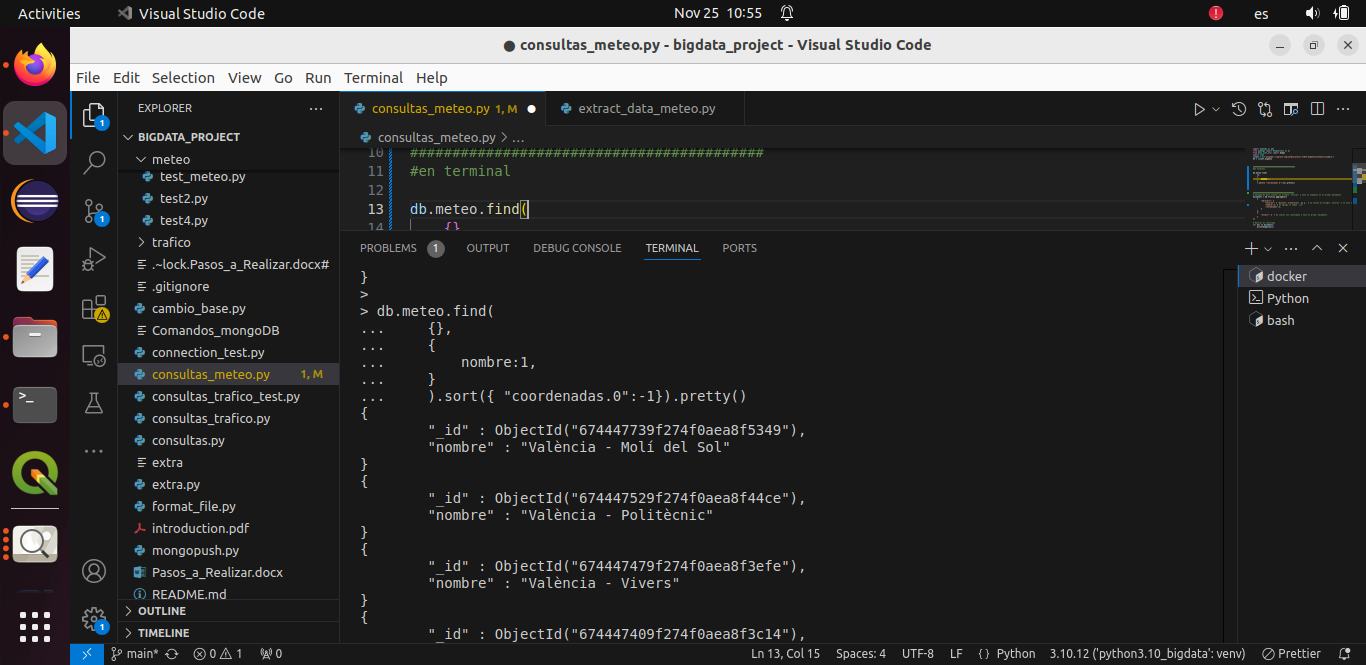
{},

{

nombre:1,

}

).sort({ "coordenadas.0":-1}).pretty()



**2.6 (1 punto)** Buscar el valor de NO2 para la Avenida de Francia para la fecha del 15 de mayo a las 12:00:00, en la salida deben figurar únicamente el identificador de la estación, las coordenadas y los valores de fecha y NO2.

##########################################

#en terminal

db.meteo.find({

nombre: { $regex: "França", $options: "i" },

"valores": {

$elemMatch: {

"fecha": ISODate("2024-05-15T12:00:00Z")

}

}

}, {

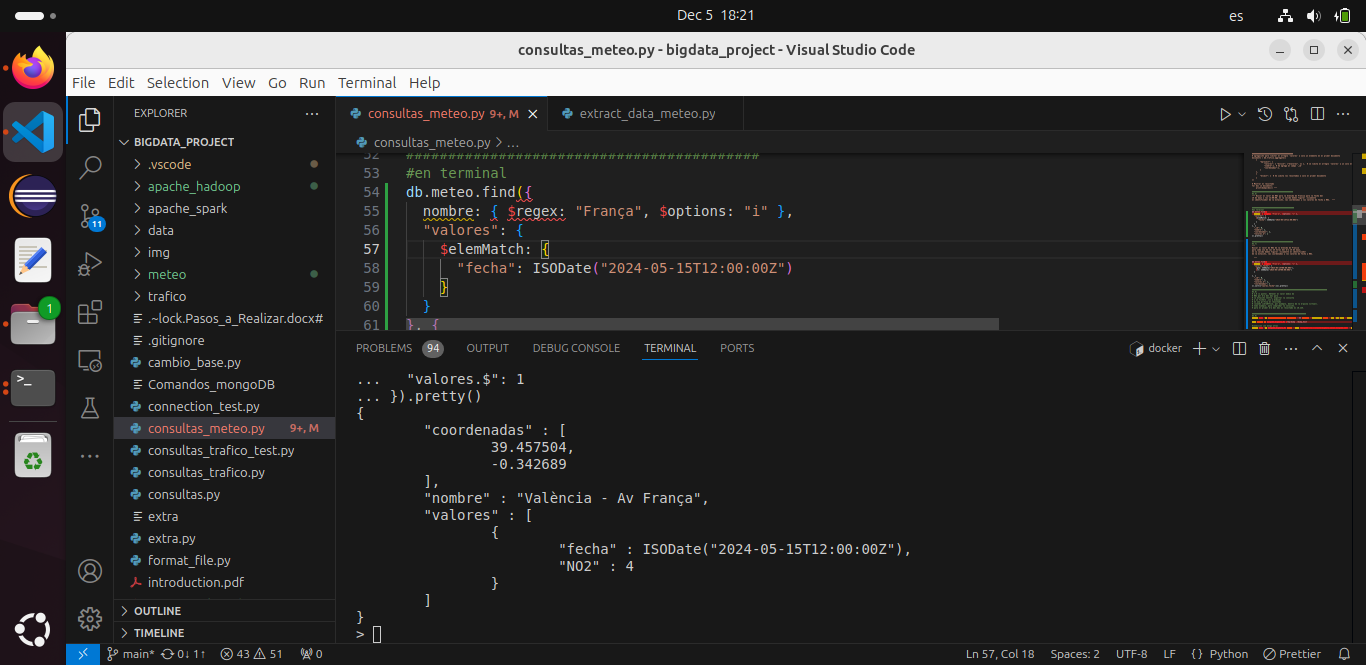
"\_id": 0,

"nombre": 1,

"coordenadas": 1,

"valores.$": 1

}).pretty()



**2.7 (0.5 puntos)** Buscar el valor de NO2 de la Avenida de Francia para cada una de las 24 horas del día 15 de mayo, en la salida deben figurar únicamente el identificador de la estación, las coordenadas y los valores de fecha y NO2.

db.meteo.aggregate([

{

$match: {

nombre: { $regex: "França", $options: "i" }

}

},

{

$unwind: "$valores"

},

{

$match: {

"valores.fecha": {

$gte: ISODate("2024-05-15T00:00:00Z"),

$lt: ISODate("2024-05-16T00:00:00Z")

}

}

},

{

$project: {

\_id: 0,

nombre: 1,

valores: 1,

coordenadas: 1

}

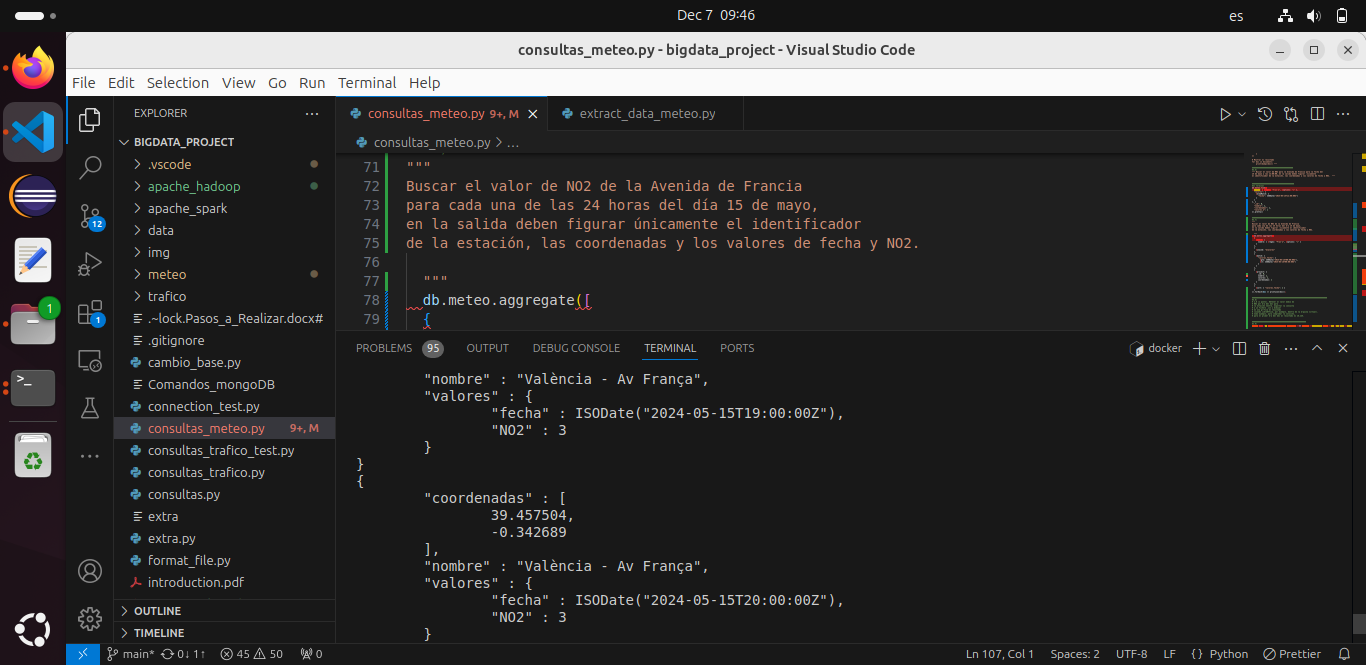
},

{

$sort: { "valores.fecha": 1 }

}

]).forEach(doc => printjson(doc));



**2.8 (1 punto).** Obtener el valor medio de NO2 por día del mes para la estación Centre, exportar la consulta a un fichero .csv y dibujar en una gráfica el resultado (usando LibreOffice, por ejemplo, dentro de la máquina virtual). Como ejemplo, para comprobar la solución, para el primer día del mes el resultado es 8.25.

Creación de colección con la consulta a exportar en.csv

db.meteo.aggregate([

{

$unwind: "$valores"

},

{

$match: { nombre: { $regex: "Centre", $options: "i" },

$expr: { $eq: [{$month: "$valores.fecha" }, 5]

}}}

,

{

$group: {

\_id: {

dia: { $dayOfMonth: "$valores.fecha" },

nombre: "$nombre"

},

meanNO2: { $avg: "$valores.NO2" }

}

},

{

$sort: { "\_id.dia": -1 }

},

{

$project: {

\_id: 0,

dia: "$\_id.dia",

nombre: "$\_id.nombre",

meanNO2: {$round:["$meanNO2", 2]}

}

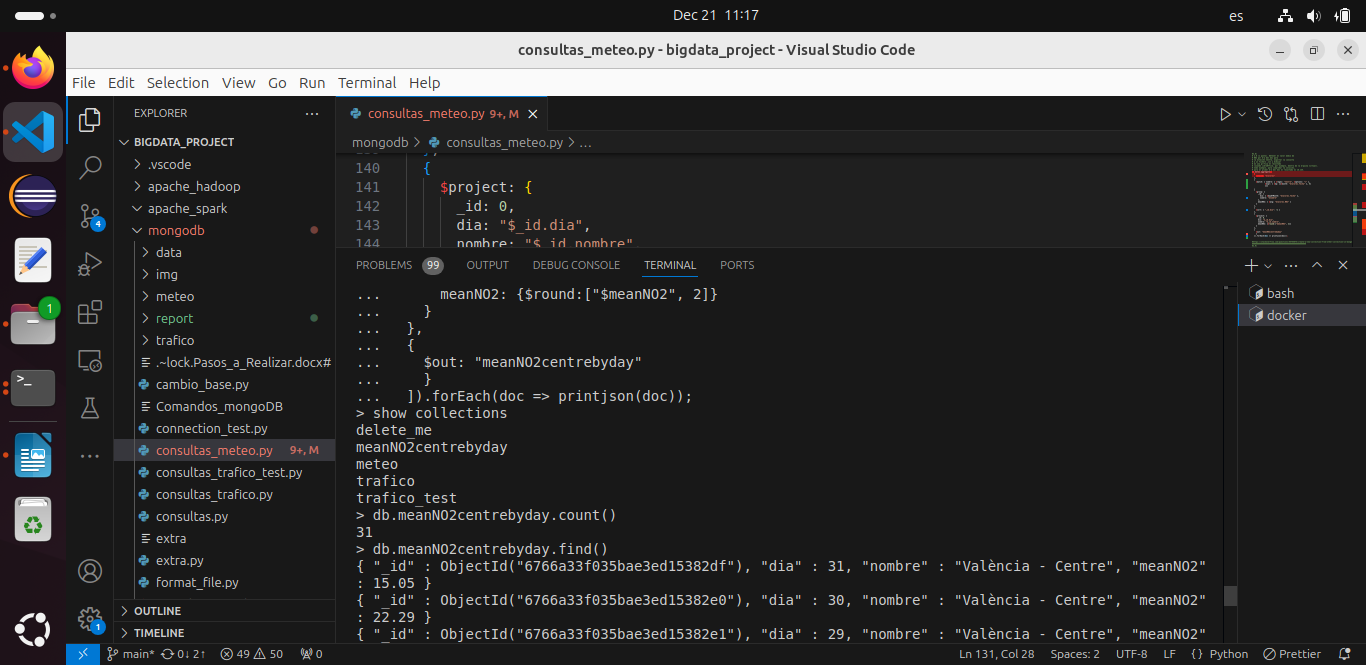
},

{

$out: "meanNO2centrebyday"

}

]).forEach(doc => printjson(doc));



Montaje del documento .csv

#export in .csv

docker exec -it mongodb\_bigdata\_01 mongoexport --db=bigdata --collection=meanNO2centrebyday --type=csv --fields=nombre,dia,meanNO2 --out=/tmp/meanNO2centrebyday.csv --username vagrant --password vagrant --authenticationDatabase admin

sudo docker cp mongodb\_bigdata\_01:/tmp/ ./mongodb/data/consults

docker exec -it mongodb\_bigdata\_01 mongoexport --db=bigdata --collection=meanNO2centrebyday --type=csv --fields=nombre,dia,meanNO2 --out=/tmp/meanNO2centrebyday.csv --username vagrant --password vagrant --authenticationDatabase admin

import matplotlib.pyplot as plt

import numpy as np

# Inicializar listas para las categorías y los valores

categories = []

values = []

# Leer el archivo CSV utilizando open()

with open('./mongodb/data/consults/tmp/meanNO2centrebyday.csv', 'r', newline='') as file:

# Omitir el encabezado

next(file)

for line in file:

nombre, dia, meanNO2 = line.strip().split(',')

categories.append(int(dia)) # Convertir el día a entero

values.append(float(meanNO2))

# Ordenar por el valor de día (de 1 a 31)

sorted\_data = sorted(zip(categories, values))

categories\_sorted, values\_sorted = zip(\*sorted\_data)

# Crear una lista de días del 1 al 31

all\_days = list(range(1, 32))

# Asegurarse de que todos los días del 1 al 31 están representados

full\_values = [0] \* 31 # Inicializar con ceros

for day, value in zip(categories\_sorted, values\_sorted):

full\_values[day - 1] = value # Asignar el valor correspondiente al día

# NO2 min and max

min\_index = full\_values.index(min(full\_values))

max\_index = full\_values.index(max(full\_values))

colors = ['grey'] \* len(full\_values)

colors[min\_index] = 'green'

colors[max\_index] = 'red'

# Crear el gráfico de barras

plt.bar(all\_days, full\_values, color=colors)

# Ajustar la visibilidad de las etiquetas del eje X (todos los días)

plt.xlabel('Day', fontsize='15')

plt.ylabel('NO2 level', fontsize='15')

plt.title('NO2 values in Valencia in May 2024', fontsize='18')

# Ajustar los ticks

plt.xticks(all\_days, fontsize='10', rotation=0)

# Ajustar los datos para la tendencia (línea)

x\_values = np.array(all\_days) # Días

y\_values = np.array(full\_values) # Niveles de NO2

# Ajuste polinómico de primer grado (línea recta)

slope, intercept = np.polyfit(x\_values, y\_values, 1)

trendline = slope \* x\_values + intercept

# Graficar la línea de tendencia

plt.plot(x\_values, trendline, color='blue', linestyle='--', label='Trendline')

# Mostrar la leyenda

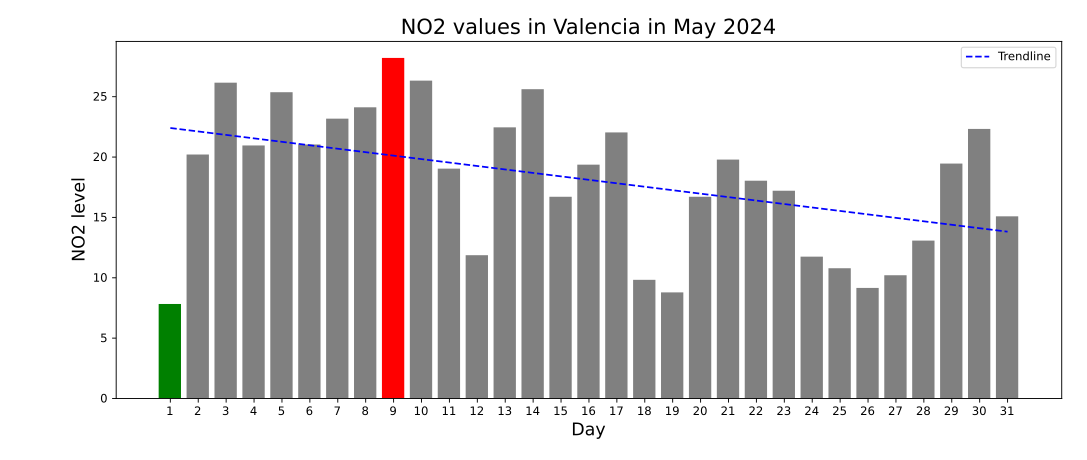
plt.legend()

# Ajustar el layout para evitar solapamientos

plt.tight\_layout()

# Mostrar el gráfico

plt.show()



**2.9 (1 punto):** Hacer copia de seguridad (ejecutable *Mongodump*) de la colección de estaciones meteorológicas e importarla a Mongo Atlas utilizando un clúster gratuito (M0). Buscar, en este clúster, para la estación Avda. Francia y el mes de mayo la fecha exacta de mayor y menor valor de NO2, la salida debe contener únicamente el nombre de la estación y los valores de fecha y NO2, el resultado se puede obtener a partir de dos consultas, una para el valor máximo y otra para el valor mínimo.

docker exec -it mongodb\_bigdata\_01 mongodump --db bigdata --collection meteo --out /tmp/dump --username vagrant --password vagrant --authenticationDatabase admin

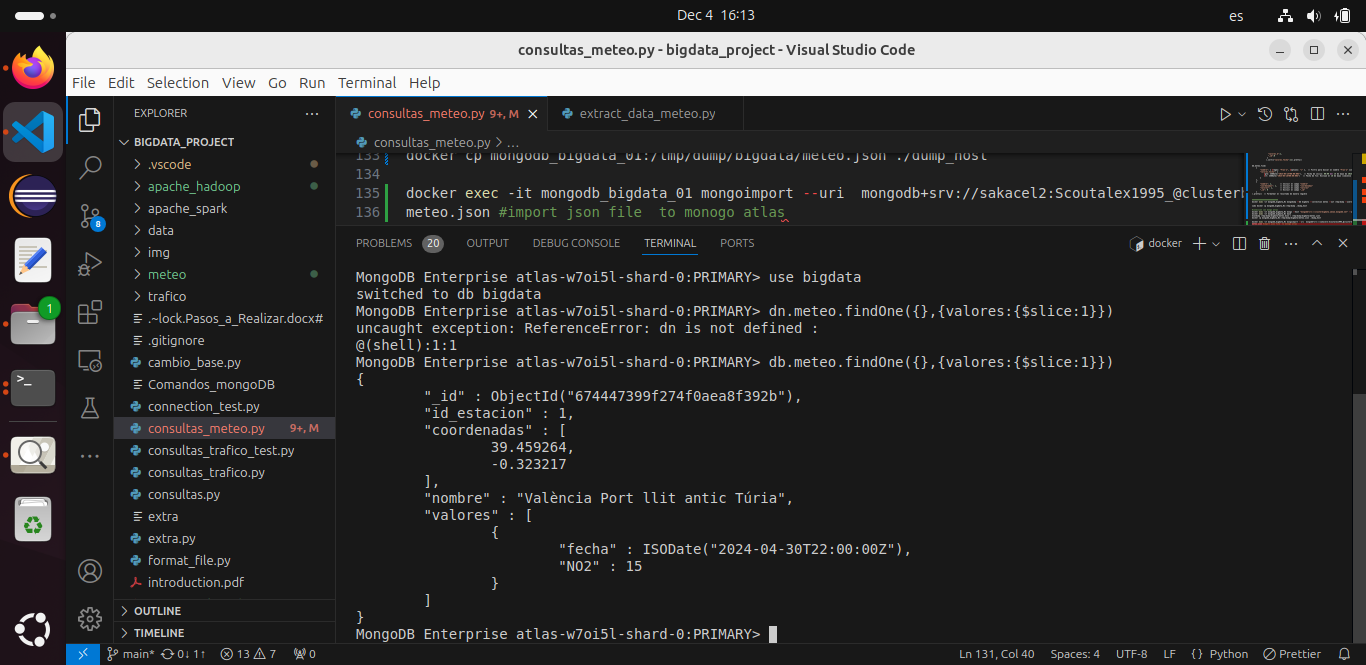
docker exec -it mongodb\_bigdata\_01 mongo --host "mongodb+srv://clusterbigdata.umnae.mongodb.net" --username sakacel2 --password Scoutalex1995\_ --authenticationDatabase admin

docker exec -it mongodb\_bigdata\_01 bash

bsondump /tmp/dump/bigdata/meteo.bson > /tmp/dump/bigdata/meteo.json

docker cp mongodb\_bigdata\_01:/tmp/dump/bigdata/meteo.json ./dump\_host

docker exec -it mongodb\_bigdata\_01 mongoimport --uri mongodb+srv://sakacel2:#######@clusterbigdata.umnae.mongodb.net/bigdata --collection meteo --type JSON --file tmp/dump/bigdata/



db.meteo.aggregate([

{

$match: {

nombre: { $regex: "França", $options: "i" } // Filtra documentos por nombre

}

},

{

$unwind: "$valores" // Descompón el array de "valores" para cada tupla de NO2 y fecha

},

{

$sort: { "valores.NO2": 1 } // Orden de los documentos por NO2 (de menor a mayor)

},

{

$group: {

\_id: null,

nombre: { $first: "$nombre" }, // Toma el primer valor de "nombre"

min: { $first: { NO2: "$valores.NO2", fecha: "$valores.fecha" } }, // Primer valor (mínimo)

max: { $last: { NO2: "$valores.NO2", fecha: "$valores.fecha" } } // Último valor (máximo)

}

},

{

$project: {

\_id: 0,

nombre: 1,

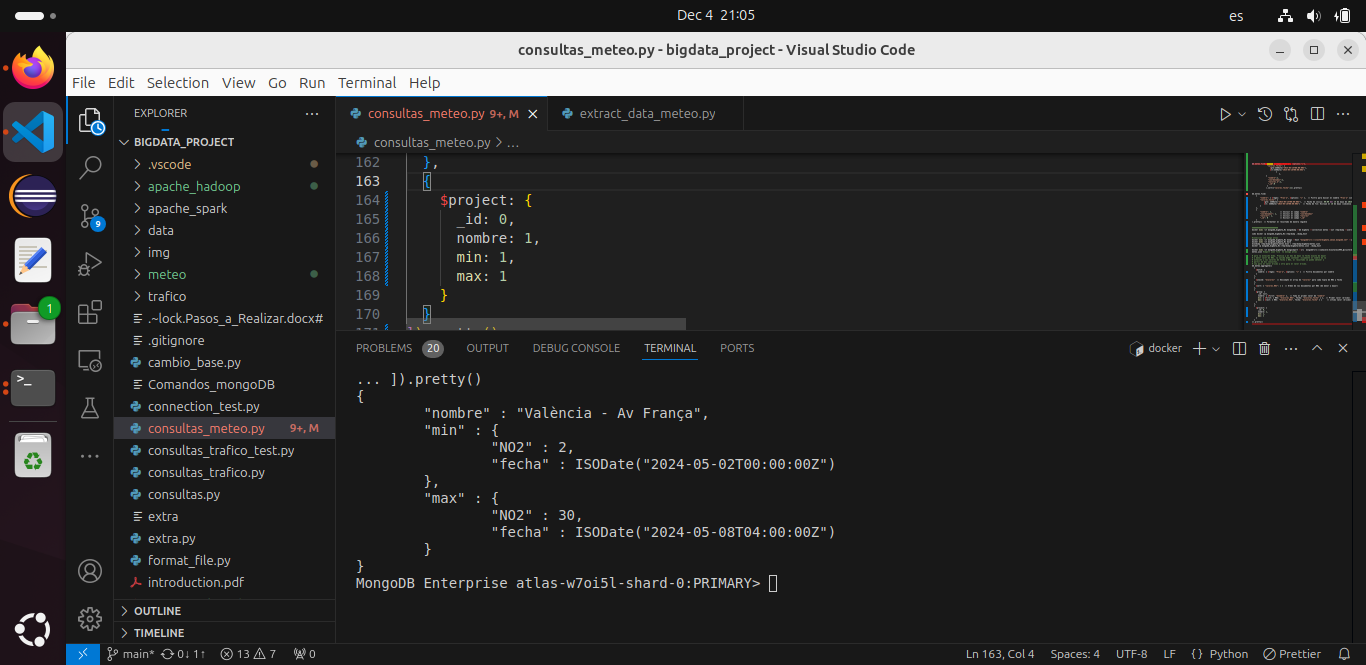
min: 1,

max: 1

}

}

]).pretty()



**Conclusiones (Evaluación de la competencia de responsabilidad y toma de decisiones)**

Se deberán presentar las conclusiones relativas a los datos con los que se ha trabajado (precisión, granularidad -resolución espacial y temporal-, confusión, valor, validez para los objetivos perseguidos, etc.), la plataforma o framework usado para su almacenamiento y/o análisis tanto de forma local como en la nube (facilidad de uso, generación de resultados, etc.), de los resultados obtenidos (correlaciones observadas, consecuencias, predicciones, agrupaciones, etc.) y sobre la bibliografía consultada por el alumno.

Este documento aborda el proceso de tratamiento de datos abiertos provenientes de diversas fuentes de información, el diseño de una base de datos en MongoDB y la realización de análisis exploratorios sobre dichos datos.

Durante el desarrollo del trabajo se enfrentaron varios desafíos relacionados con:

* La lectura de información en formatos no estructurados o semiestructurados.
* La falta de datos en algunas entradas de los documentos.
* La presencia de valores en columnas que no correspondían a datos reales (valores muy alejados de los esperados).
* La ausencia de ciertas columnas con datos relevantes en algunos documentos.
* El manejo de diferentes extensiones de archivo, como .csv y .txt.

Para extraer la información necesaria y crear documentos en MongoDB, se utilizó Python 3.12 junto con la biblioteca pymongo. Los datos se almacenaron en una base de datos MongoDB dentro de un contenedor Docker, utilizando la imagen oficial de la versión 4. La elección de esta versión se debió a las características del equipo en el que se desarrolló el trabajo, siendo compatible con sus componentes.

Una vez creados los documentos, las consultas sobre la base de datos se realizaron sin complicaciones. En cuanto al uso de MongoDB Atlas (la solución en la nube), se empleó el paquete gratuito para importar una copia de los datos meteorológicos previamente estructurados en el servidor Docker local.

El desarrollo del proyecto requirió apoyo constante de documentación, tanto visual (videotutoriales en YouTube, y cursos en línea) como escrita (apuntes de las clases de Big Data en 2024 y 2022, documentación oficial de MongoDB, Mongo Atlas y Docker). Esto fue esencial para implementar el código necesario para la extracción, estructuración de datos y consultas en la base de datos, lo cual implicó un extenso proceso de investigación.

Uso de ChatGPT

Se utilizó la herramienta de inteligencia artificial ChatGPT (https://chatgpt.com) como apoyo en tres aspectos clave:

* Identificación de errores en el código previamente creado.
* Propuestas de mejoras en el código.
* Explicaciones de procesos y uso de funciones y bibliotecas que no se encontraron en foros, documentación en línea, guías, videos o libros.

Reflexión final

Por último, como aprendizaje de este trabajo, se ha comprendido el proceso necesario para trabajar con Big Data en un mundo altamente globalizado, caracterizado por una variedad de formatos, formas de presentar la información y temporalidades en su disponibilidad. En muchos casos, se dispone de millones de datos, lo que hace imprescindible homogenizar toda la información en un único formato antes de importarla a una base de datos. En este caso se utilizó MongoDB, pero este principio aplica a cualquier base de datos SQL o NoSQL, así como a entornos como Apache Hadoop y Apache Spark.

Este trabajo ha evidenciado que trabajar con Big Data requiere una gran dedicación y un entendimiento profundo de la estructura de cada formato de datos. Estas habilidades convierten al profesional del Big Data en un recurso altamente valorado en 2024 y en el futuro próximo.

**Bibliografía (Evaluación de la competencia de responsabilidad y toma de decisiones)**

Se deberá presentar la bibliografía correspondiente que tendrá que contener todas las búsquedas realizadas por el alumno, tanto de carácter teórico en libros y artículos como de carácter técnico (búsqueda en internet de soluciones para la correcta instalación de las herramientas, para la generación de código -p.e. en blogs, plataforma Stack Overflow, información libre de diferentes cursos, etc.), todas estas referencias deberán estar adecuadamente referenciadas en el texto del proyecto de forma que no haya bibliografía sin su correspondiente referencia en el texto o en el código elaborado. Por último, en el apartado de conclusiones debe haber una parte sobre la bibliografía usada donde el alumno deberá considerar la importancia, fiabilidad y sencillez de uso o implementación de las diferentes fuentes usadas.

**Fuentes web**

Docker. (n.d.). Mongo: Official image for MongoDB [Imagen de Docker]. Docker Hub. Recuperado el 21 de diciembre de 2024, de <https://hub.docker.com/_/mongo>

Stack Overflow. (2015, 15 de junio). How to start a MongoDB shell in Docker container. Recuperado el 21 de diciembre de 2024, de <https://stackoverflow.com/questions/32944729/how-to-start-a-mongodb-shell-in-docker-container>

Valencia Open Data. (n.d.). Dataset de intensitat de trànsit - Trams Intensidad de Tráfico [Datos abiertos]. OpenDataSoft. Recuperado el 21 de diciembre de 2024, de <https://valencia.opendatasoft.com/explore/dataset/intensitat-transit-trams-intensidad-trafico-tramos/>

Generalitat Valenciana. (n.d.). Datos obtenidos a partir de la RVVCCA. Conselleria de Agricultura, Desarrollo Rural, Emergencia Climática y Transición Ecológica. Recuperado el 21 de diciembre de 2024, de <https://mediambient.gva.es/es/web/calidad-ambiental/datos-obtenidos-a-partir-de-la-rvvcca>

IETF. (2016, agosto). RFC 7946: GeoJSON format. Internet Engineering Task Force. Recuperado el 21 de diciembre de 2024, de <https://datatracker.ietf.org/doc/html/rfc7946>

Stack Overflow. (2019, 19 de enero). Create a new collection from another collection in MongoDB using $out. Recuperado el 21 de diciembre de 2024, de <https://stackoverflow.com/questions/54754674/create-a-new-collection-from-other-collection-in-mongodb>

MongoDB. (2024, 21 de diciembre). Mongoexport examples. MongoDB Documentation. Recuperado el 21 de diciembre de 2024, de <https://www.mongodb.com/docs/database-tools/mongoexport/mongoexport-examples/#std-label-mongoexport-fields-example>

**Recursos utilizados**

OpenAI. (2024). ChatGPT. Recuperado el 21 de diciembre de 2024, de [https://chatgpt.com](https://chatgpt.com/)

**Documentación de la clase**

Base de datos NoSQL MongoDB. (2022). MongoDB en la nube: MongoDB Atlas [Presentación]. Curso Big Data y Minería de Datos Geoespaciales.