Universidad Zaragoza
1542

Trabajo Fin de Máster

Simulación y predicción del efecto de los flujos de tráfico en la calidad del aire

Simulation and prediction of the effect of traffic flows on air quality

Autor

David Sáez García

Directora y codirectora

Raquel Trillo Lado

Lorena Marrodán Bretón

Escuela de Ingeniería y Arquitectura
2021

AGRADECIMIENTOS

En mi trabajo de fin de grado agradecí a Raquel Trillo-Lado la oportunidad de haber trabajado con ella y con el resto de compañeros en el proyecto TRAFAIR. Desde aquí quiero volver a agradecerle dicha oportunidad y el poder haber seguido trabajando en TRAFAIR durante más de un año. También quiero sumar a Lorena Marrodán a este agradecimiento, quién ha sido codirectora de este trabajo y que siempre ha estado disponible para resolver cualquier duda o cuestión.

Tampoco quiero olvidarme de enviar mis agradecimientos a mi familia, a mis amigos y a mi novia, quienes han estado en todo momento ahí para apoyarme en los momentos más difíciles.

Finalmente, me gustaría enviar una pequeño agradecimiento a todos los profesores y compañeros que he tenido a lo largo de mi etapa universitaria. Con este trabajo se cierra esa etapa de mi vida, la que considero que es seguramente la más importante y bonita.

Resumen

Simulación y predicción del efecto de los flujos de tráfico en la calidad del aire

El objetivo de este trabajo es el desarrollo de una herramienta que permita predecir la dispersión de contaminantes en la atmósfera en función de los flujos de tráfico y de otras fuentes de contaminación. Concretamente, se analiza la dispersión de los óxidos de nitrógeno (NO_x). Además, la herramienta permite simular situaciones hipotéticas como, por ejemplo, analizar cómo afecta a la contaminación atmosférica el incremento del uso del vehículo eléctrico en el entorno urbano.

Para ello, se han tomado como base un modelo de tráfico y un modelo de dispersiones que ya se encontraban en funcionamiento. En el caso del modelo de tráfico, a este se le han realizado varias mejoras como la capacidad predictiva. Además, se ha realizado un análisis empleando los datos obtenidos de las simulaciones de tráfico en la ciudad de Zaragoza, donde se ha visto que el tráfico es más intenso a las horas de entrada y salida del trabajo y que disminuye en los meses de julio y agosto.

En cuanto al modelo de dispersiones, inicialmente, este funcionaba empleando como fuente de emisiones únicamente el tráfico rodado. Por lo tanto, se la han añadido nuevas fuentes de emisión de contaminación atmosférica que se pueden encontrar en un entorno urbano, como son las generadas por las calefacciones domésticas y las generadas por las industrias. Además, se ha hecho un análisis de la capacidad predictiva del modelo, en el que se ha visto que las predicciones tienen correlación con la realidad aunque sería conveniente analizar cómo podría mejorarse el modelo.

Por otro lado, se ha llevado a cabo un proceso de calibración de una red de sensores de bajo coste que obtiene medidas de la concentración de diferentes contaminantes en la atmósfera. Esta red de sensores ha sido desplegada con anterioridad a este trabajo como una actividad que se enmarca dentro del proyecto europeo TRAFAIR. Para ello, se ha utilizado un algoritmo de aprendizaje automático y se ha visto que la capacidad predictiva es mayor o menor en función del contaminante considerado.

Por otro lado, se ha estudiado un posible futuro escenario de tráfico rodado en la ciudad. Para ello, se ha simulado cómo se reduciría la contaminación de NO_x si el 10 % de los vehículos actuales de Zaragoza fuesen eléctricos, concluyendo que los valores de NO_x se reducirían en un 9 %.

Finalmente, se ha mejorado sustancialmente la aplicación web TraFlow la cual ya

se encontraba en funcionamiento. Se le ha añadido la funcionalidad de visualizar los datos obtenidos de las simulaciones realizadas con el modelo de dispersiones.

En conclusión, se han diseñado y mejorado una serie de herramientas que permiten, por un lado, simular los flujos de tráfico y por otro, predecir la dispersión de contaminantes en la atmósfera considerando diferentes fuentes de contaminación atmosférica en la ciudad.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción y objetivos

En este capítulo se expone el contexto en el que se engloba el proyecto. Además, se incluye una pequeña introducción de todo el trabajo que se ha realizado. Por otro lado, se exponen brevemente los objetivos que se pretenden alcanzar con el presente proyecto.

1.1. Introducción

Se estima que el 11 % de las muertes anuales en España se deben a la contaminación atmosférica ocasionada por el uso de combustibles fósiles [1], aumentando esta cifra hasta aproximadamente el 20 % si se analiza la población global. Entre las principales fuentes de contaminación atmosférica se encuentra el tráfico vial, lo que ha provocado que se empiecen a tomar medidas en las grandes ciudades, tanto a nivel nacional como internacional. Sin embargo, hasta el momento, las administraciones públicas han carecido de herramientas precisas para estimar el nivel de contaminación del atmosférica en función del flujo de tráfico y poder realizar predicciones de la calidad del aire.

Por otro lado, se da la circunstancia de que en febrero de 2017 la Comisión Europea advirtió a España de las violaciones continuas de la normativa referente a la contaminación del aire. Reiteradamente se incumple el límite de contaminación del aire para el dióxido de nitrógeno (NO_2), cuya principal fuente de emisiones es el tráfico rodado.

Además, cabe destacar la situación de alerta climática en la que el mundo se encuentra envuelto en la actualidad. En el último informe del IPCC (*Intergovernmental Panel on Climate Change*), publicado en septiembre de 2021, se señala a la emisión de gases de efecto invernadero producidos por la actividad humana como la principal causa del cambio climático, así como de que cada vez se produzcan más fenómenos climáticos extremos.

En este contexto, el proyecto europeo TRAFAIR[1] surge con el objetivo de proporcionar herramientas precisas que permitan analizar la situación actual y, en consecuencia, poder tomar medidas que contribuyan a mejorar la calidad del aire a nivel urbano.

El presente trabajo se desarrolla en el marco del proyecto TRAFAIR en la ciudad de Zaragoza y tiene como objetivo principal el desarrollo de herramientas que permitan simular y predecir la calidad del aire en la ciudad de Zaragoza, en función de diferentes flujos de tráfico. Este trabajo es continuación del trabajo de fin de grado titulado *Desarrollo e implementación de herramientas de monitorización y simulación de tráfico basadas en datos abiertos* [2] realizado por el mismo autor del presente trabajo de fin de máster.

1.2. Objetivos

El objetivo principal de este trabajo es el desarrollo de una herramienta que permita predecir la dispersión de contaminantes en la atmósfera en función de los flujos de tráfico y de otras fuentes de contaminación.

Además, la herramienta ha de permitir a los usuarios visualizar dichas predicciones de forma sencilla y simular situaciones hipotéticas, como, por ejemplo, analizar el efecto sobre la contaminación atmosférica de un aumento en el uso del vehículo eléctrico.

Para la consecución de dicho objetivo, el proyecto se dividió en varias etapas que se detallan a continuación:

- **Estudio de metodologías y herramientas a utilizar y análisis de requisitos**. En esta fase se estudiaron diferentes fuentes de datos con información relevante para el trabajo, incluyendo artículos científicos que tratan sobre la problemática de la contaminación atmosférica.

- **Mejora de un modelo de tráfico** ya existente y que se desarrolló en [2], encargado de realizar la predicción, simulación y cálculo de la contaminación atmosférica generada por el tráfico rodado.

- **Análisis de los datos obtenidos por las simulaciones** a lo largo del año 2020. También se evaluará el rendimiento del modelo.

- **Calibración de las medidas de contaminación** realizadas por una red de sensores de bajo coste.

[1] `http://trafair.eu/`

- **Mejora de un modelo de dispersiones** que ya se encuentra en funcionamiento y que es capaz de calcular la dispersión de contaminantes en función únicamente del tráfico rodado.
- **Inclusión de otras fuentes de contaminación en un modelo de dispersiones** de contaminantes. Además de tener en cuenta la contaminación generada por el tráfico rodado, también se incluirán las siguientes fuentes de emisiones:
  - Las emisiones generadas en las viviendas de los ciudadanos. El principal foco de contaminación que se tendrá en cuenta en este apartado son las calefacciones domésticas.
  - Las emisiones generadas por las industrias que se encuentran dentro del dominio de la ciudad considerado.
  - Las emisiones generadas como consecuencia de ciertas actividades desarrollas en una de las depuradoras de la ciudad.
  - Las emisiones generadas como consecuencia de la actividad de un crematorio.
- **Comparación de los resultados obtenidos con el modelo de dispersiones** con medidas de contaminación obtenidas en tiempo real. Es decir, se hará una evaluación del modelo.
- **Generación y evaluación de escenarios de tráfico.** Se simulará cómo se reduciría, hipotéticamente, la contaminación atmosférica cuando, por ejemplo, el porcentaje de vehículos eléctricos que circula por la ciudad sea mayor que el actual.
- **Introducción de mejoras en TraFlow**. TraFlow es una aplicación web que se realizó en [2] y que permite la visualización de la predicción de tráfico. Se le añadirá la funcionalidad de visualizar mapas de contaminación generados con el modelo de dispersiones.

1.3. Arquitectura del sistema

En la figura 1.1 se detalla la arquitectura a alto nivel del sistema desarrollado. De todos ellos, los componentes más relevantes son:

- **Modelo de tráfico**. Se encarga de predecir y simular flujos de tráfico. Además, calcula las emisiones de $NO_x(NO + NO_x)$ producidas por dicho flujo de tráfico.

- **Modelo de dispersiones**. A partir de los datos generados por el modelo de tráfico, la predicción meteorológica y los datos de emisiones de otros fuentes, como las calefacciones domésticas, trata de predecir cómo los contaminantes se van a distribuir en la atmósfera de la ciudad.

- **Traflow**. Es una aplicación web que permite al usuario visualizar los resultados de las simulaciones de tráfico, así como los mapas de dispersión de contaminantes de una manera sencilla.

El resto de componentes del sistema son:

- Los **proveedores de datos** necesarios para la realización del proyecto. Aquí se incluyen, entre otros, los datos históricos de flujos de tráfico aportados por el ayuntamiento de Zaragoza.

- La base de datos **PostgreSQL** que se utiliza para almacenar todos los datos necesarios para el proyecto.

- Una **red de sensores LoRa** encargados de recoger datos de contaminación en tiempo real en la ciudad de Zaragoza.

- Una **API REST** para facilitar la entrada de datos en el modelo de dispersión.

- **Geoserver**, que se encarga de almacenar y de publicar los datos de dispersiones y de simulaciones de tráfico.

A lo largo del documento se explicará más en detalle cada uno de estos componentes.

Además, en dicha figura 1.1 se ilustran todas las conexiones entre los distintos componentes del sistema, reflejando las distintas entradas y salidas que necesitan y producen todos los componentes.

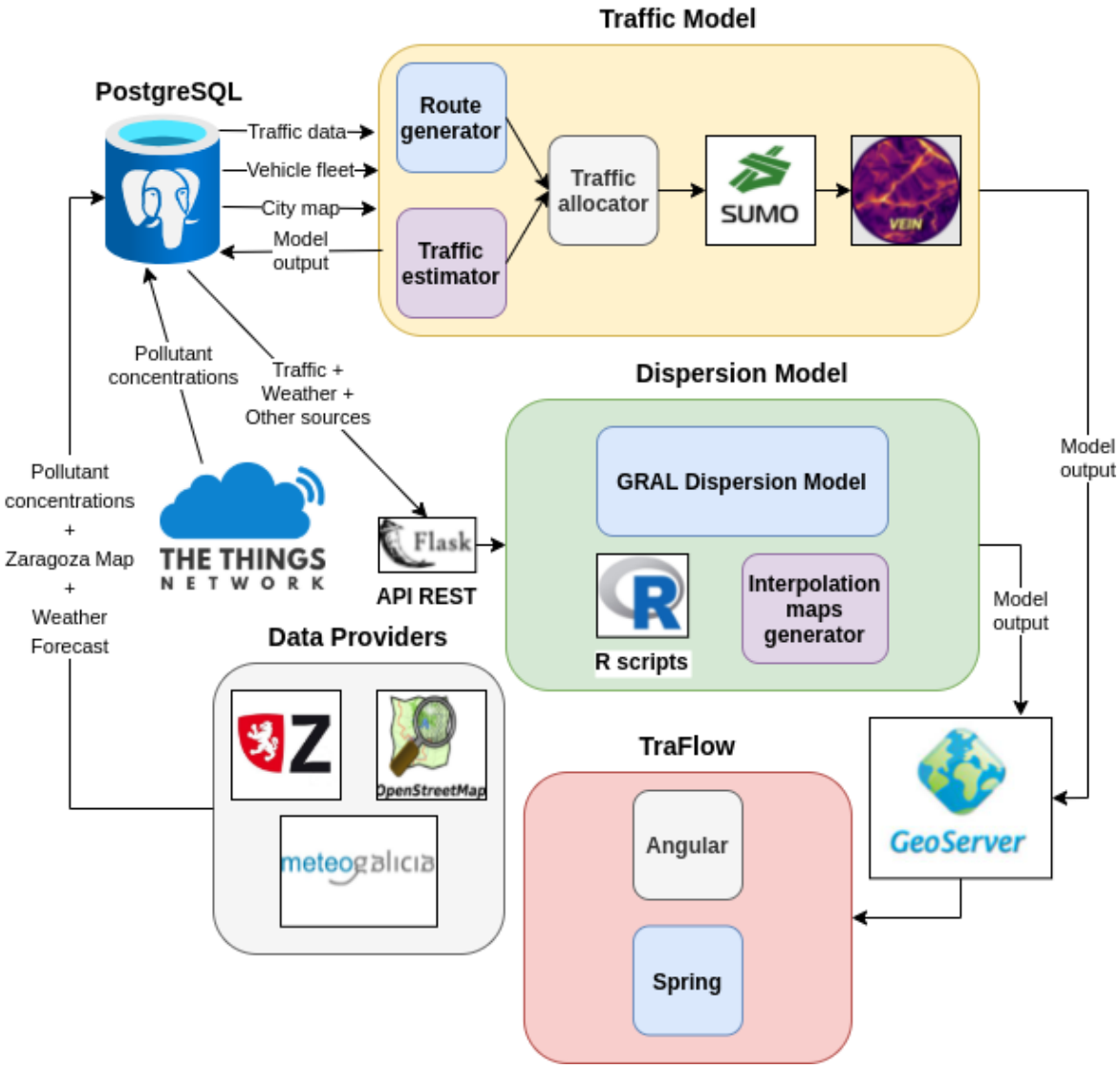

Figura 1.1: Arquitectura del sistema desarrollado.

1.4. Herramientas y tecnologías utilizadas

A continuación, se detallan las diferentes herramientas que se han empleado a lo largo del desarrollo de este trabajo, incidiendo en aquellas que han resultado más relevantes:

- OpenStreetMap (OSM, [3]): Proyecto colaborativo para crear mapas editables y libres. Concretamente, se utiliza el mapa de la ciudad de Zaragoza como base para realizar las simulaciones y predicciones de flujos de tráfico.
- Simulation of Urban Mobility (SUMO, [4]): Paquete de simulación de tráfico que permite simular cómo un conjunto de coches se mueve dentro de una red de carreteras.
- Vehicular Emissions INventories (VEIN, [5]): Paquete de R [6] que permite estimar las emisiones producidas por los distintos tipos de vehículos.
- PostgreSQL: Sistema Gestor de Bases de Datos Relacional. En el marco del proyecto TRAFAIR se ha diseñado un modelo de datos que, en el caso de Zaragoza, se ha implementado con este sistema.
- PostGIS: Complemento de PostgreSQL que facilita la gestión de datos espaciales.
- Spring Boot: Tecnología basada en Java empleada en el desarrollo de aplicaciones web.
- Bootstrap: Framework CSS y JavaScript utilizado para el diseño de interfaces.
- Angular: Framework para crear y mantener aplicaciones web de una sola página.
- Geoserver: Un servidor de datos espaciales en el que se publican los datos sobre las simulaciones de tráfico y predicciones de contaminación.
- La biblioteca *Scikit-learn* [7] para la generación de modelos de machine learning.
- El framework *Flask* para la creación de una API REST.

Por otro lado, se han utilizado 4 lenguajes de programación distintos: Python, R, Java y TypeScript. Además, se han utilizado los entornos PyCharm, IntelliJ, WebStorm y RStudio.

Capítulo 2

Modelo de tráfico

En este capítulo se describe el modelo de tráfico que se ha utilizado y mejorado en el trabajo. Se detallarán los pasos que se han seguido para la mejora del modelo ya existente, las decisiones que se han tomado y, finalmente, se realizará un análisis estadístico de los datos obtenidos de las simulaciones.

2.1. ¿De dónde se parte?

En primer lugar, se va a explicar brevemente el modelo de tráfico que se desarrolló en [2], y que constituye uno de los pilares del presente trabajo: qué datos se utilizan como entrada para alimentar a dicho modelo y qué datos de salida proporciona el modelo. Una descripción más detallada del mismo puede encontrarse en el trabajo disponible para su consulta en el repositorio oficial de la Universidad de Zaragoza, Zaguán[1].

En la figura 2.1 se ilustra el modelo desarrollado. A continuación, se describen los diferentes datos de entrada del modelo:

- **Datos históricos de flujos de tráfico**. En la ciudad de Zaragoza existen 46 estaciones permanentes. Una estación permanente es un dispositivo que se encuentra activo las 24 horas del día en un determinado punto de una vía, y que se encarga de medir el número de coches que pasan por dicha vía en un determinado periodo de tiempo configurable. En el caso de Zaragoza, dicho periodo es de una hora. La ubicación de estas estaciones es fija y no cambia a lo largo del tiempo. Es decir, por cada día, hora y estación se tiene el número de coches que pasaron por el punto donde la estación se encuentra instalada. Estos datos fueron proporcionados por el ayuntamiento de Zaragoza y para el periodo de tiempo desde el 01-01-2018 hasta el 31-12-2019.

- El **mapa de la ciudad de Zaragoza**. Dicho mapa contiene toda la geometría

[1] `https://zaguan.unizar.es/`

de las calles de la ciudad: distancia, anchura, intersecciones, rotondas... Además, también incluye otros datos, como los semáforos que hay en la ciudad, las restricciones de giros, el número de carriles de una calle o la velocidad máxima de una calle. Dicho mapa se obtuvo de OpenStreetMap [3].

– La **flota de vehículos de Zaragoza**. Incluye todos los tipos de vehículos que hay registrados en la ciudad de Zaragoza diferenciados por 4 características: el tamaño del motor, el tipo de carburante que utilizan y la clase. El término clase hace referencia a una clasificación europea que determina el tipo de vehículo que es, por ejemplo, los coches eléctricos pertenecerán a la clase *Eco*. Además, se tiene el total de vehículos que hay registrados de cada tipo. Estos datos fueron proporcionados por el ayuntamiento de Zaragoza.

Para facilitar el acceso a estos datos, se almacenaron en una base de datos PostgreSQL.

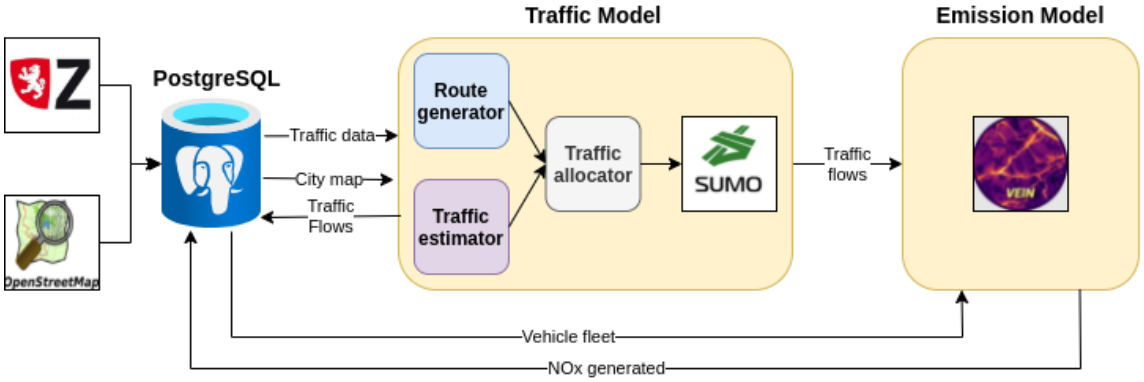

Figura 2.1: Arquitectura del modelo de tráfico y emisiones del que se parte.

Por otro lado, se distinguen dos grandes componentes en la arquitectura: el **modelo de tráfico** y el **modelo de emisiones**. Actualmente el modelo de emisiones está integrado dentro del modelo de tráfico, pero en un inicio se trataba de dos objetos que se encontraban lógicamente separados y que funcionaban por separado, de ahí esta distinción.

En cuanto al modelo de tráfico, este toma como entrada el día a simular, los datos de tráfico históricos y el mapa de la ciudad, y genera como salida el número de vehículos (y su velocidad media) que han circulado en cada una de las calles de la ciudad en cada una de las horas del día que se ha simulado. En la tabla 2.1 se ejemplifica cómo es una salida de este componente.

Internamente puede observarse que el modelo de tráfico está compuesto por 4 componentes:

– Un estimador de tráfico. Como ya se ha comentado anteriormente, se disponen de datos históricos de 46 estaciones distintas. A partir de dichos datos, se construye

Day	Edge	Hour	Num Cars	Mean Speed
01-11-2021	156398	0	53	5.3
01-11-2021	156398	1	29	6.1
...	...	...	...	..
01-11-2021	947812	22	101	12.5
01-11-2021	947812	23	120	12.5
...	...	...	...	...

Tabla 2.1: Ejemplo de salida del modelo de tráfico antiguo.

un modelo de regresión lineal múltiple que se utiliza para predecir el tráfico en cada uno de estos 46 puntos para el día que se quiera simular por cada hora del día. De esta manera se consigue simular días para los que el ayuntamiento no ha proporcionado datos.

– Un generador de rutas. Por cada calle en la que se encuentra instalada una estación, se generan rutas de tráfico que pasan por dichas calles. Una ruta no es más que una secuencia de calles por las que puede circular un vehículo con una serie de restricciones:

  - Ha de ser correcta. Es decir, dadas dos aristas consecutivas es posible llegar desde la anterior a la siguiente, no se circula en dirección contraria, no se toman direcciones prohibidas, etc.
  - Una calle no puede ser transitada dos veces en la misma ruta. De esta forma se evitan rutas circulares que pasen varias veces por la misma calle, para posteriormente poder repartir el tráfico de forma más uniforme por la ciudad.
  - Ha de tener como máximo una longitud de 30 calles. Esta restricción se toma para que el cálculo de las rutas pueda hacerse en un tiempo razonable.

– Un asignador de rutas, encargado de generar el tráfico para la simulación. Este componente toma como entrada las rutas generadas y la salida del estimador de tráfico para cada una de las estaciones y hora, y procede del siguiente modo: si se ha predicho que en la estación X deberían pasar N coches en una hora determinada, entonces se generan N vehículos y se le asigna una ruta a cada uno. Esta asignación se realiza de forma aleatoria con pesos, es decir, una ruta que contenga muchas vías grandes como avenidas de 3 carriles tiene más probabilidades de ser escogida que otra por la que únicamente se incluyan calles de un carril. Además, se tiene en cuenta la duración estimada de la ruta (en función de su longitud, velocidad permitida en las calles, semáforos...) para determinar

en qué momento el vehículo ha de entrar en la simulación. Es decir, la entrada de un vehículo en la simulación se distribuye uniformemente a lo largo de la hora que está siendo simulada, para evitar que, por ejemplo, todo el tráfico se genere en los primeros 15 minutos de la hora simulada y que al final de la simulación no hay tráfico circulando. Este componente se diseñó mediante varios scripts de Python.

– El simulador de tráfico SUMO [4]. Este componente lleva a cabo las simulaciones a partir del tráfico generado por el asignador de rutas y el mapa de la ciudad de Zaragoza. Es decir, simula cómo los vehículos se mueven por la ciudad, siguiendo las rutas que les han sido asignadas y teniendo en cuenta muchos parámetros, como la congestión del tráfico o los semáforos. En [2] pueden encontrarse más detalles de cómo funciona este simulador. Finalmente, la salida del simulador es procesada y es lo que se ofrece como salida final del modelo de tráfico. Este componente se diseñó mediante varios scripts de Python.

Por otro lado se encuentra el modelo de emisiones. Este toma como entrada la salida del modelo de tráfico y genera la cantidad de NO_x por distancia recorrida ($\frac{kg}{km}$) producida por los flujos de tráfico en cada una de las calles de la ciudad. Es decir, calcula los NO_x emitidos por los vehículos en base a los datos generados por el modelo de tráfico, tal y como se ejemplifica en la tabla 2.2. Para lograr dicho objetivo se ha utilizado VEIN [5], que es un paquete de R que permite estimar las emisiones producidas por un vehículo en función del tipo de vehículo, la velocidad a la que circula y la temperatura ambiente. A partir de este paquete se han construido los scripts que permiten el cálculo de las emisiones.

Day	**Edge**	**Hour**	**NOx**
01-11-2021	156398	5.3	0.5
01-11-2021	156398	6.1	0.43
...	...	...	...
01-11-2021	947812	12.5	1.21
01-11-2021	947812	12.5	1.46
...	...	...	...

Tabla 2.2: Ejemplo de salida del modelo de emisiones antiguo.

Finalmente, tanto la salida generada por el modelo de tráfico, como la generada por el modelo de emisiones, son almacenadas en la base de datos para su posterior explotación.

2.2. Mejoras respecto al modelo inicial

En esta sección se van a exponer las mejoras realizadas al modelo de tráfico presentado anteriormente. Además, se ilustrará la arquitectura final del modelo de tráfico.

2.2.1. Precálculo de las emisiones de NO_x

A alto nivel, podemos pensar que el paquete VEIN [5] es una función con la firma que se detalla en el algoritmo 2.1. Es decir, para calcular las emisiones a partir de los flujos de tráfico de una simulación, el proceso que se llevaba a cabo es el siguiente:

1. Iterar sobre todas las horas del día simulado.
2. Iterar sobre todos los tipos de vehículos que hay en la ciudad.
3. Iterar sobre todas las calles de la ciudad.
4. Como temperatura se utilizaba la temperatura media histórica en el mes al que pertenece el día simulado. Por ejemplo, si se están calculando las emisiones para el 05-11-2021, se cogerá como temperatura la media de noviembre en Zaragoza.
5. Para cada trío *(hora, vehículo, calle)*, recuperar los datos obtenidos en la simulación de tráfico: el número de coches y su velocidad media asociada.
6. Llamar a VEIN para calcular las emisiones de la tupla *(hora, vehículo, calle, numCoches, velocidad)*.
7. Promediar la emisión calculada en función del porcentaje que representa el vehículo respecto a la flota entera.

Algoritmo 2.1: Pseudocódigo del cálculo de emisiones.

```
def VEIN(num_coches, velocidad_media, temperatura,
         tipo_vehichulo):
    # Pre:  num_coches es el numero de coches para el que se
    #       quiere calcular la emision de NOx
    #
    #       velocidad_media es la velocidad media a la que
    #       han circulado esos coches
    #
    #       temperatura es la temperatura ambiente
    #
    #       tipo_vehiculo es el tipo de vehiculo. Por
    #       ejemplo, un camion diesel
    #
```

```
    # Post: Devuelve la cantidad de NOx en kg/km que producen
    #       "num_coches" coches circuluando a una velocidad
    #       "velocidad_media" cuando la temperatura ambiente
    #       es "temperatura"
    return do_estimations()
```

El proceso explicado anteriormente es muy costoso ya que para calcular las emisiones de un día en concreto hay que hacer 18.396.000 llamadas (24 horas x 100 tipos de vehículos x 7665 calles en la ciudad) a la función VEIN. Inicialmente, este cálculo conllevaba más de 24 horas de cómputo, por lo tanto era necesario buscar otra alternativa.

Para ello, en primer lugar se analizó en profundidad el código fuente de la biblioteca VEIN. En dicho análisis se detectó que el proceso era optimizable: muchas partes del código eran paralelizables, mientras que otras directamente utilizaban implementaciones de funciones no óptimas. Sin embargo, al final nada de esto se llevó a cabo puesto que se optó por otra solución.

La solución que finalmente se empleó fue el **precálculo de las emisiones discretizando la velocidad**. Es decir, en lugar de lanzar el procedimiento tras cada simulación del proceso de tráfico, las emisiones se tienen precalculadas para un número finito de velocidades. Para ello es necesario:

- Determinar el rango discreto de velocidades. El rango escogido fue [0, 1, ..., 149] $\frac{km}{h}$, sin incluir números decimales. En caso de que la velocidad media registrada en una calle sea mayor a 149 se considerará que es 150 $\frac{km}{h}$.

- Para cada tipo de vehículo, calcular la emisión de NOx que produce un único vehículo de dicho tipo a cada una de las 150 velocidades discretizadas.

- Para cada una de las 150 velocidades discretizadas, calcular la emisión producida por un vehículo promedio de Zaragoza. Por ejemplo, si únicamente hubiera dos tipos de vehículos, A y B, donde A representa el 80 % del total de la flota y B el 20 %, y se ha calculado que a $50\frac{km}{h}$ A genera un NOx de $5\frac{kg}{km}$ y B de $3\frac{kg}{km}$, entonces el NOx generado por un vehículo medio de Zaragoza a $50\frac{km}{h}$ es de $4{,}6\frac{kg}{km}(5 \cdot 0{,}8 \cdot 3 \cdot 0{,}2)$.

Como resultado final de este proceso, se tiene la cantidad de NOx en $\frac{Kg}{Km}$ que genera un coche medio de Zaragoza. Este valor estará calculado por cada tripleta tipo de coche, velocidad, temperatura. Como ya se ha comentado, la velocidad varía entre [0, ..., 149] y la temperatura depende del mes. Puesto que el cálculo de las emisiones a partir de la salida del modelo de tráfico es inmediata con la solución adoptada, se determinó que

lo más correcto era integrar el modelo de emisiones dentro del modelo de tráfico. Es decir, a partir de ahora la salida del modelo de tráfico también muestra la cantidad de NOx que se ha generado, tal y como se ejemplifica en la tabla 2.3.

Day	Edge	Hour	Num Cars	Mean Speed	NOx
01-11-2021	156398	0	53	5.3	0.5
01-11-2021	156398	1	29	6.1	0.43
...	...	...	...	...	...
01-11-2021	947812	22	101	12.5	1.21
01-11-2021	947812	23	120	12.5	1.46
...	...	...	...	...	...

Tabla 2.3: Ejemplo de salida del modelo de tráfico con las emisiones integradas.

2.2.2. Inclusión y discretización de la temperatura

Como ya se ha comentado el la sección 2.2.1, inicialmente el modelo tomaba una temperatura fija en función del mes del día que se pretendía simular. Esta decisión se tomó porque:

- Es más sencillo. En la propia documentación de VEIN [5] se sugiere que se tome esta solución.
- No se disponía de datos meteorológicos.

Por lo tanto, para poder tener en cuenta la temperatura son necesarias dos cosas:

- Una fuente de datos que proporcione esta información. En este caso, MeteoGalicia[2] se ha prestado para ceder la información meteorológica.
- Que la simulación se realice en un día para el que hay información meteorológica.

Para recuperar la información meteorológica se ha programado una tarea periódica que se ejecuta diariamente a las 7 de la mañana, encargada de recuperar la previsión meteorológica para el día en curso y los dos siguientes. El flujo que se lleva a cabo para recuperar y almacenar dicha información se ilustra en la figura 2.2.

Por otro lado, es necesario modificar los scripts encargados de llevar a cabo la simulación, para que cojan esta temperatura y si no está disponible, que cojan la temperatura media que se utilizaba previamente, tal y como se ilustra en el algoritmo 2.2. Finalmente se hace una discretización de la temperatura, haciendo un proceso análogo al de la discretización de la velocidad, y almacenando estos precálculos para su posterior explotación.

[2] `https://www.meteogalicia.gal/`

Algoritmo 2.2: Pseudocódigo para recuperar la temperatura.

```
def do_simulation(day,...):
    # Pre:  day es el dia a simular
    #
    # Post: hace una simulacion del dia
    tmp = retrieve_temp_from_db()
    if not tmp:
        tmp = retrieve_mean_temp_from_db()
    return perform_sim(day, temp, ...)
```

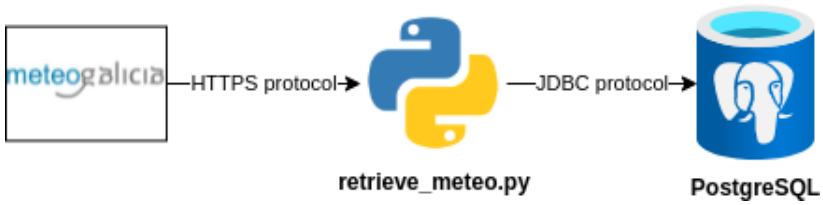

Figura 2.2: Flujo para recuperar diariamente la previsión meteorológica.

2.2.3. Mejoras sobre el generador de rutas

En primer lugar, **se incrementó la longitud máxima de las rutas** construidas por el generador de rutas. Inicialmente esta longitud era de 20 calles, mientras que ahora dicha longitud es de 40. Esto se ha podido conseguir porque se obtuvo acceso a una máquina con más recursos que la que se estaba empleando al inicio del proyecto. De esta forma, se consigue que el tráfico se reparta más uniformemente a lo largo de la ciudad y que no quede acumulado alrededor de los puntos donde se sitúan las estaciones permanentes en la realidad. Este cambio se realizó simplemente cambiando el valor de una variable de entorno en los scripts realizados.

Además, también se retocó el generador de rutas para que permita **indicarle una serie de calles** por las que el usuario **no** quiere que se **generen rutas de tráfico**. De este modo se logra que el modelo de tráfico tenga la opción de generar simulaciones en situaciones hipotéticas, como, por ejemplo, que una o varias calles de la ciudad se encuentren cortadas y por lo tanto el tráfico no pueda circular.

2.2.4. Mejoras sobre el predictor de tráfico

En [2] se explica detalladamente el predictor de tráfico que se ha construido. En resumen, se trata de una regresión lineal múltiple que tiene las siguientes variables de entrada:

– El identificador de la estación permanente para la que se quiere realizar la predicción. Se trata de una variable categórica con 46 posibles valores distintos,

uno por cada estación.

- El mes correspondiente al día para el que se quiere realizar la predicción. Se trata de una variable categórica cuyos valores van desde enero a diciembre.

- La hora para la que se quiere analizar la predicción. Esto es otra variable categórica cuyos valores van del 0 al 23.

- El tipo de día. Esto es otra variable categórica cuyos posibles valores son 'E' (entre semana), 'S' (sábado) o 'F' (festivo).

A partir de estas variables de entrada se construye un modelo para predecir el flujo de tráfico, que es la variable de salida y cuyos datos también están incluidos en el dataset. En cuanto al rendimiento del modelo se obtuvo un valor de R^2-ajustado de 0.7736. Cabe destacar que dicho valor es el obtenido directamente del propio entrenamiento, es decir, no se realizó una validación cruzada. Por tanto, se entrenarán varios modelos distintos y se escogerá el mejor de todos.

Datos de entrada para el entrenamiento

Como ya se ha comentado anteriormente, el ayuntamiento de Zaragoza cedió datos de tráfico correspondientes a los años 2018 y 2019 recogidos por 46 estaciones permanentes. En el trabajo desarrollado en [2] únicamente se tuvieron en cuenta los datos relativos a 2018, puesto que el resto de datos aún no estaban disponibles. Es decir, el *dataset* de entrenamiento ahora incluye datos de dos años en vez de uno. Cabe destacar que cada unidad del dataset se corresponde con el número de coches que pasarán por una estación y su velocidad media a una hora de un día determinado. Además de las variables de entrada que se utilizaron para el modelo de regresión lineal antiguo, se ha incluido:

- La velocidad media. Es la velocidad media de los vehículos en km/h que pasaron por la estación en la hora a la que se corresponde el dato.

- El día de la semana. Es una variable categórica cuyos valores van del 1 al 7.

- El trimestre del año. Es una variable categórica cuyos valores van del 1 al 4.

- El flujo de tráfico. El número de coches que pasaron por la estación en la hora a la que se corresponde el dato.

En la tabla 2.4 se ilustra cómo son estos datos de entrenamiento. La información de si un día es festivo o no se ha obtenido del calendario laboral de la ciudad de Zaragoza.

Estación	Mes	Hora	Tipo día	Weekday	Trimestre	Flujo
EP42.1	Enero	15	E	5	1	97
EP20.1	Marzo	9	S	6	1	131
...	...	...	...	...	...	...
EP18.2	Junio	9	D	7	2	77
...	...	...	...	...	...	...

Tabla 2.4: Ejemplo de datos del dataset de entrenamiento.

Modelos de machine learning, hiperparámetros de entrenamiento y variables de entrada valiosas

Una vez se procesaron los datos de entrenamiento, el siguiente paso es determinar qué modelos de machine learning se van a probar para escoger el mejor de todos ellos. En este trabajo se ha optado por 3 modelos distintos, para los cuales será necesario ajustar una serie de hiperparámetros:

- La **regresión Lasso** [8]. Se analizarán varias configuraciones distintas, probando varios valores para el hiperparámetro λ, y escogiendo el mejor modelo de todos.
  - Los valores que se probarán para el hiperparámetro λ serán 0, 20, 40, 60, 80, 100.
- El ***Random Forest*** [9]. Se probarán varias configuraciones distintas, combinando varios valores para el dos hiperparámetros: el número de árboles y el número mínimo de muestras que se requieren para dividir un nodo. Finalmente, se escogerá el mejor modelo de todos.
  - Los valores que se probarán para el hiperparámetro del número de árboles seŕan 100, 300, 500, 700, 900.
  - Los valores que se probarán para el hiperparámetro del número mínimo de muestras para dividir un nodo son 8, 12, 16, 20.
- El **perceptrón multicapa** [10]. Se probarán varias configuraciones distintas, probando varios valores para dos hiperparámetros: el número de capas y el número de neuronas por capa. Finalmente, se escogerá el mejor modelo de todos.
  - Los valores que se probarán para el hiperparámetro de número de capas son 2, 5, 8, 11.
  - Los valores que se probarán para el hiperparámetro de número de neuronas por capa son 4, 8, 12.

Por otro lado, es importante destacar que en el trabajo realizado en [2] se generó un único modelo para todas las estaciones permanentes. En este trabajo se ha decidido que **se generará un modelo para cada una de las 46 estaciones**, es decir, un total de 46 modelos.

Además, es necesario determinar cuáles de las variables son realmente valiosas y cuáles aportan ruido. Es decir, se trata de determinar si por ejemplo el mes influye en el tráfico o no. Para ello, se han definido las 8 configuraciones que se muestran en la tabla 2.5.

Mes	Hora	Tipo día	Velocidad	Weekday	Trimestre
		✓			
	✓	✓			
	✓	✓			✓
✓	✓	✓			
	✓			✓	
✓	✓			✓	
	✓			✓	✓
✓	✓	✓		✓	

Tabla 2.5: Configuraciones a probar para las variables de entrada.

En definitiva, **se probarán un total de 304 modelos por cada una de las 46 estaciones**: *8 configuraciones distintas x (6 modelos lasso + 5 x 4 modelos RF + 4 x 3 modelos NN)*. Esto hace un total de 13984 modelos a evaluar.

Evaluación de los modelos

Para determinar qué modelo es el mejor, en primer lugar es necesario definir qué métricas se van a emplear para evaluar un modelo. En este caso se ha optado por utilizar dos métricas típicas para evaluar modelos de *machine learning*: el R^2-ajustado y la raíz del error cuadrático medio, $RMSE$[3].

Además, se ha llevado a cabo una validación cruzada. Este procedimiento consiste en dividir el conjunto de entrenamiento en k subconjuntos y generar el modelo k veces: la primera vez se generará entrenando con todos los subconjuntos excluyendo al primero, la segunda con todos los subconjuntos excluyendo al segundo, y así sucesivamente. Al subconjunto que se queda fuera se le llama subconjunto de test y se utiliza para calcular las métricas R^2 y RMSE una vez se ha generado el modelo. Como por cada validación cruzada se obtienen k métricas, las métricas finales de la validación cruzada se calculan haciendo la media de las métricas de todos los subconjuntos. La validación

[3] `https://www.bmc.com/blogs/mean-squared-error-r2-and-variance-in-regression-analysis/`

cruzada tiene la ventaja de que se calculan dichas métricas con datos que el algoritmo no conoce, por lo tanto ofrece una imagen más realista del rendimiento del algoritmo.

Por tanto, por cada modelo se realizará una validación cruzada, obteniéndose las métricas R^2 y *RMSE*. De esta forma se podrá decidir qué modelo es el que mejores resultados aporta.

Finalmente, tras haber entrenado todos los modelos para todas las estaciones se escogerá el mejor modelo global. Es decir, se tratará de determinar cuál es la mejor configuración. Para las 304 configuraciones que se prueban para cada modelo, la mejor configuración global será la que mejores métricas obtenga haciendo la media de los valores obtenidos entre las 46 estaciones para dicha configuración.

Resultados

En la tabla 2.6 se muestran los 5 mejores modelos obtenidos de acuerdo con las métricas *RMSE* y R^2. A la vista de los resultados, se observa que, el mejor modelo:

- Es un ***Random Forest***.
- Sus hiperparámetros son **300 árboles** y un **mínimo de 8 muestras para dividir un nodo**.
- Usar como variables de entrada la **hora**, el **día de la semana**, el **tipo de día** y el **mes**.

Por lo tanto este modelo es el escogido y el que se utilizará para realizar las predicciones de tráfico. En el anexo 7.3 se detallan los resultados completos del entrenamiento.

Hiperparámetros	Variables	RMSE	R2
RF (trees=300, min_split=8)	hour;weekday;type_day;month	141.4485	0.9172
RF (trees=500, min_split=8)	hour;weekday;type_day;month	141.5462	0.9171
RF (trees=100, min_split=8)	hour;weekday;type_day;month	141.5731	0.9170
RF (trees=100, min_split=12)	hour;weekday;type_day;month	141.8598	0.9167
RF (trees=500, min_split=12)	hour;weekday;type_day;month	142.1329	0.9164

Tabla 2.6: Mejores 5 modelos obtenidos en el entrenamiento del predictor de tráfico.

2.2.5. Arquitectura final

La arquitectura final del modelo de tráfico se ilustra en la figura 2.3. En resumen, las mejoras introducidas respecto al modelo de tráfico desarrollado en [2] han sido:

- Integración del modelo de emisiones dentro del modelo de tráfico.

– Precálculo de las emisiones de NO_x, discretizando el rango de posibles velocidades de entrada para el algoritmo.

– Inclusión y discretización de la temperatura en el modelo.

– Generación de rutas para el tráfico excluyendo las calles que se indiquen. Así pueden simularse situaciones como el corte de una calle.

– Mejora en el algoritmo de predicción de tráfico.

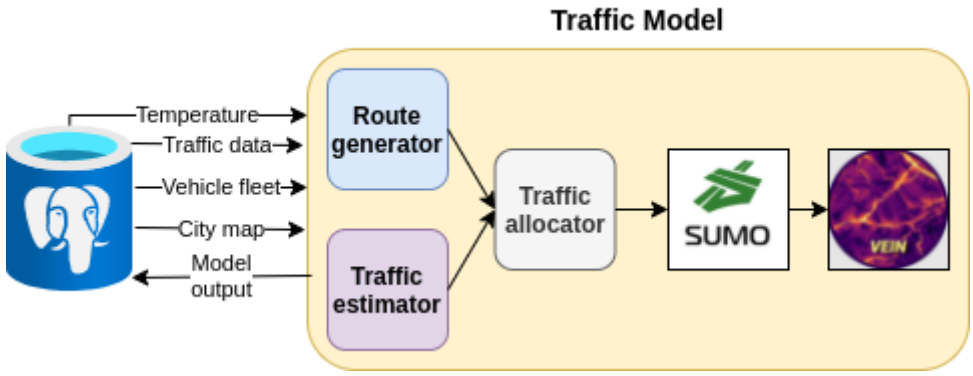

Figura 2.3: Arquitectura final del modelo de tráfico incluyendo las mejoras

2.3. Análisis y evaluación de los datos simulados

En esta sección se va detallar cómo se ha empleado una técnica de *clustering* de datos. Además, se va a llevar a cabo un análisis descriptivo de los datos de simulaciones correspondientes al año 2020.

2.3.1. Clustering de calles

El *clustering* o agrupamiento es una técnica de aprendizaje no supervisado dedicada a encontrar grupos o *clusters* dentro de un conjunto de datos. Es decir, busca agrupar las observaciones de forma que todas las incluidas en un mismo grupo sean lo más parecidas posibles entre sí. En este trabajo se tratará de buscar grupos dentro de los datos correspondientes a una simulación de un día, para así poder agrupar las calles en función de la contaminación generada por el tráfico rodado. Además de servir cómo análisis, esta información de agrupamiento se utilizará como entrada del modelo de dispersiones, lo que se explicará en puntos posteriores.

El algoritmo de agrupamiento que se ha decidido utilizar es el ***K-means*** [11]. En cuanto a los datos de entrada, el algoritmo recibe 7665 datos de 24 dimensiones:

– **7665 datos** porque el mapa de la ciudad que se extrajo de OpenStreetMap [3] tiene 7665 aristas, siendo una arista una calle o una parte de la misma. Es decir, el algoritmo recibirá un dato por cada calle o parte de calle de la ciudad.

– Cada dato tiene **24 dimensiones**, es decir, está compuesto por 24 valores distintos. Se debe a que cada dimensión se corresponde con una hora de las 24 horas del día que se haya simulado.

– El **valor que almacena una dimensión de un dato** se corresponde con la relación entre la contaminación que ha producido el tráfico rodado en una calle a una determinada hora, y la contaminación que ha producido el tráfico rodado en ese misma calle en la hora punta del día. La hora punta del día es la hora a la que la suma de la contaminación total de todas las calles es mayor.

– Se ha utilizado la **distancia euclídea** para calcular la variación dentro de un *cluster*. Se intentó emplear otras métricas como el *Dynamic Time Warping* [12] y el *Fast Dynamic Time Warping* [13], pero ninguno de las dos finalizaba en un tiempo de cómputo razonable. Por lo tanto, se descartó su uso.

Resultados

A continuación, se van a exponer los resultados obtenidos al aplicar el *clustering de calles* individualmente a todos los días simulados correspondientes al año 2020. En primer lugar, en la tabla 2.7 se muestra el número de días en los que hubo un determinado número de *clusters*. Como puede observarse, la gran mayoría de días se tienen 2 *clusters*, excepto 6 días en los que hay o 6 o 7 *clusters*. Por otro lado, en la tabla 2.8 se detalla el número medio de aristas y el total de emisiones (en $\frac{kg}{km}$) por cada cluster, en función del número total de *clusters*. Por ejemplo, la primera fila de la tabla indica que los días en los que hay únicamente dos *clusters*, el *cluster* cuyo identificador es 1 tiene una media de 3330 aristas y la emisión media total de NO_x es de $356.49\frac{kg}{km}$.

Si se analizan los datos correspondientes a los días en los que hay 2 *clusters*, que suponen la mayoría de días, se ve que hay dos clusters claramente diferenciados: el mayor en cuanto a número de aristas, que es también el más contaminante, y el otro *cluster*, que es el más pequeño en cuanto a número de aristas y el menos contaminante.

Número de *clusters*	**Número de días**
2	351
6	10
7	5

Tabla 2.7: Número de días de 2020 en los que hubo un determinado número de *clusters*.

Total clusters	Cluster	Num. edges	NOx [kg/km]
2	1	3330	356.49
2	2	2449	200.66
6	1	1072	239.72
6	2	2209	189.26
6	3	718	95.51
6	4	706	41.15
6	5	655	26.12
6	6	419	1.94
7	1	77	74.22
7	2	1418	27.09
7	3	1154	25.16
7	4	1211	17.83
7	5	926	2.00
7	6	254	0.21
7	7	734	0.01

Tabla 2.8: Valor medio de emisiones y calles dentro de un *cluster*, en función del número total de *clusters*.

En definitiva, el *clustering* ha servido para determinar que:

- Por lo general, las calles se agrupan en dos grupos en función de la contaminación que producen.
- Hay un grupo con mayores niveles de contaminación, que se corresponde también con el más numeroso en cuanto a número de calles.

2.3.2. Análisis descriptivo

En esta subsección se tratará de determinar cómo se comporta el tráfico a lo largo de un día, a lo largo de una semana y a lo largo de un año. En primer lugar, en la figura 2.4 se muestra el tráfico medio (número de coches) por hora para cada día de la semana. En dicho gráfico puede concluirse que:

- El tráfico no varía prácticamente entre los días de entre semana.
- El tráfico es mayor entre semana que el fin de semana.
- Entre las 0.00 y las 05.00 horas es cuando se observa el menor tráfico.
- Existen algunos picos de tráfico, concretamente a las 08.00, a las 13.00 y a las 18.00. Estos picos coinciden con las horas típicas de entrada y salida del trabajo.

Además, en la figura 2.5 se muestra el número total de coches promedio que circulan en la ciudad por cada día de la semana. Como ya se vió en la figura 2.4, el tráfico es

similar de lunes a viernes, y disminuye los fines de semana. El día con menos tráfico es el domingo.

Por otro lado, en la figura 2.6 se muestra un *boxplot* donde se realiza una comparación del flujo de tráfico medio por día entre meses. En dicha figura puede verse que el tráfico es muy similar desde enero hasta junio. A partir de entonces, el tráfico se reduce en los meses de julio y agosto, especialmente en este último. Estos resultados eran esperados, ya que estos meses son en los que normalmente los trabajadores toman sus vacaciones y abandonan la ciudad por semanas. El resto de meses, desde septiembre a diciembre, se comportan como los meses iniciales del mes. De hecho, puede concluirse que el tráfico es muy similar todos los meses del año, excluyendo julio y agosto. Finalmente, en la figura 2.7 se muestra el tráfico medio por hora en función de la semana del año. Como mostró la figura 2.6, el tráfico es muy similar a lo largo del año excepto en las semanas que se corresponden con julio y agosto.

En definitiva, se ha mostrado que **el flujo de tráfico es mayor de lunes a viernes**. Además, se ha mostrado que **el tráfico disminuye durante las noches**, y que hay determinados **picos de tráfico a lo largo del día** que coinciden con las horas típicas de entrada y salida del trabajo. Finalmente, se ha visto que **el tráfico es prácticamente el mismo comparando entre meses**, excepto en julio y agosto.

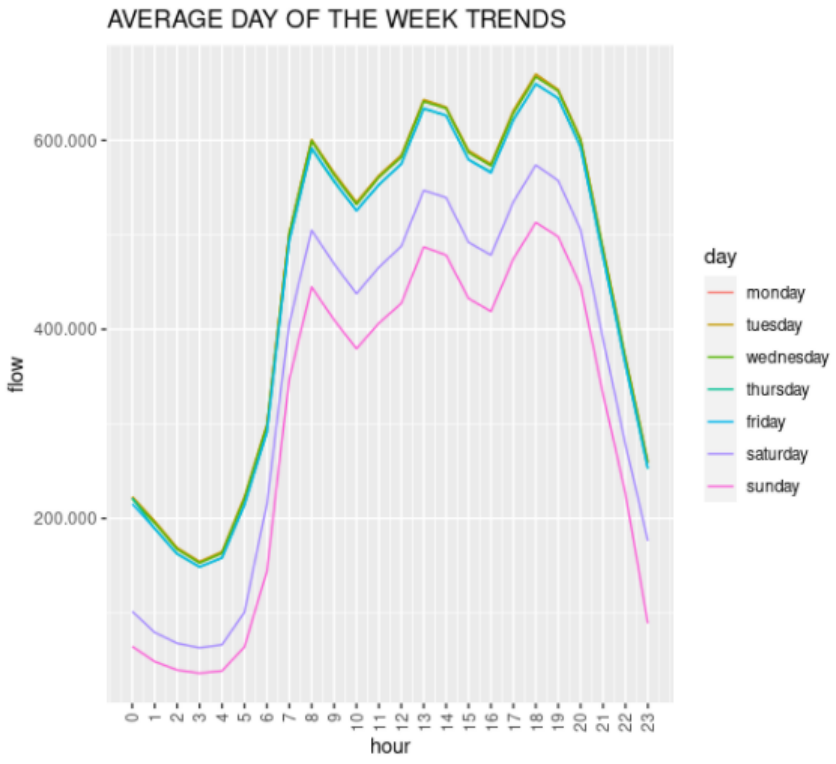

Figura 2.4: Tráfico medio dentro de un día.

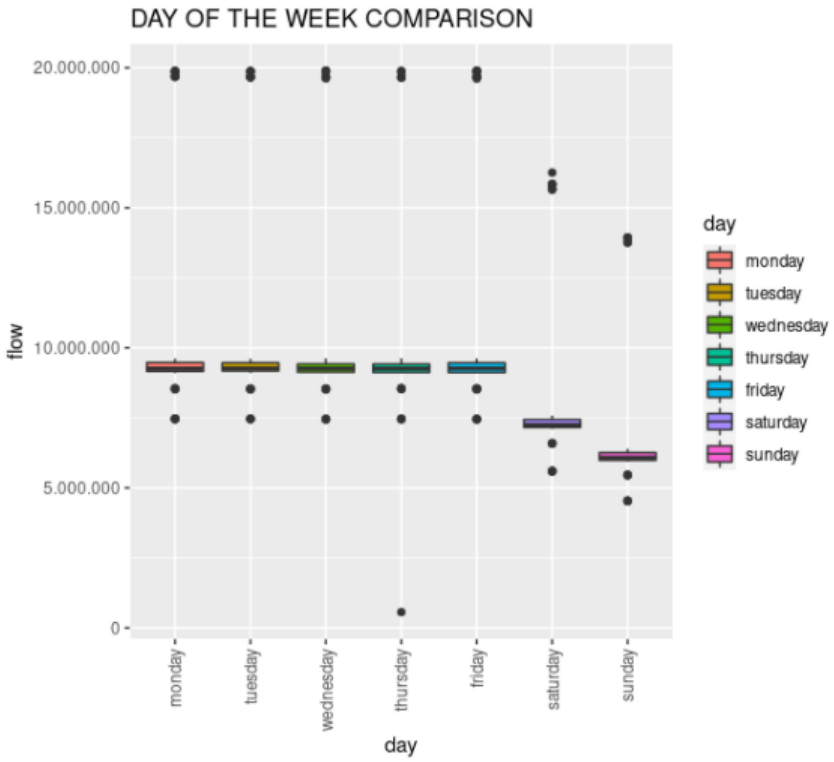

Figura 2.5: *Boxplot* que muestra el número total medio de coches por cada día de la semana.

Figura 2.6: Tráfico medio por día en función del mes del año.

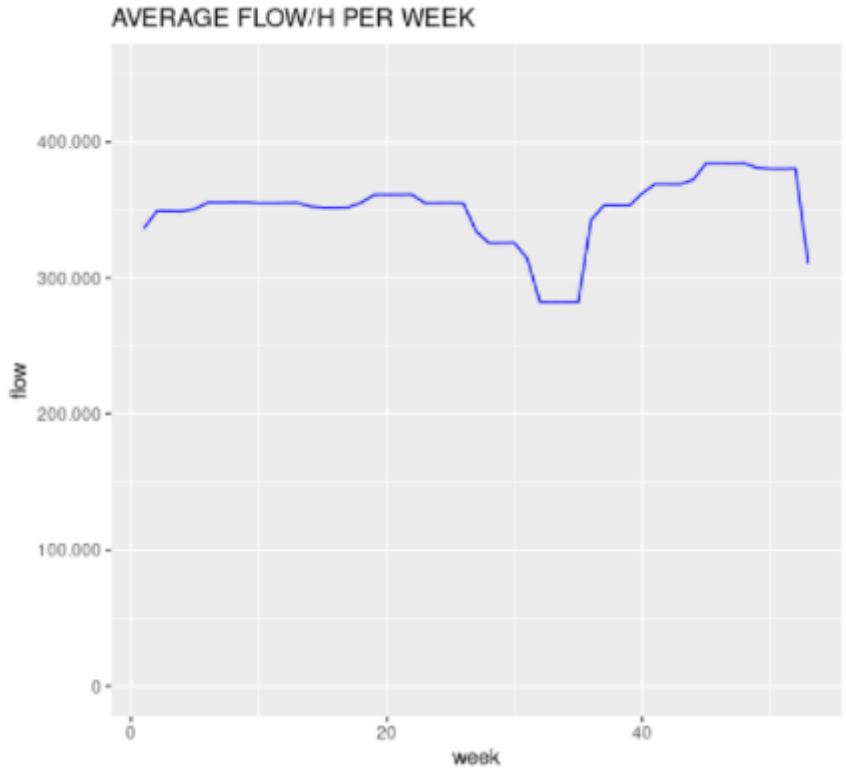

Figura 2.7: Tráfico medio por hora en función de la semana del año.

2.3.3. Evaluación

En este apartado se va a realizar una evaluación del modelo de tráfico, es decir, se tratará de determinar si el tráfico que se genera dentro de la simulación de tráfico (llevada a cabo por la herramienta SUMO [4]) se corresponde con el tráfico que se espera que se genere. Para realizar esta comparación se han simulado los días del 16 de diciembre de 2019 al 22 de diciembre de 2019, y estos datos de simulación se emplearán en la comparación.

Como ya se ha comentado anteriormente, el predictor de tráfico genera, para un determinado día, el número de coches que han atravesado 46 puntos clave de la ciudad por hora. Por tanto, se compararán estos 46 valores en cada hora con el número de vehículos que realmente han pasado por dichos puntos en la simulación. Esta evaluación es un proceso análogo al que se realizó en [2], y se ha vuelto a repetir en el presente trabajo para ver cómo se comporta el modelo con las mejoras que se han introducido.

Antes de realizar el análisis es necesario definir el término **error de simulación**. Este error es la diferencia absoluta entre el tráfico que se esperaba en la simulación (obtenido con el predictor de tráfico) y el tráfico que realmente se tuvo en la simulación. Por tanto, el **error relativo** es el porcentaje de este error. Por ejemplo, si el tráfico esperado en un punto a una hora era 10 y se ha obtenido un error de 15, el error relativo

será el 50 %. Se usará esta métrica para evaluar el modelo.

En primer lugar, se ha realizado un pequeño análisis del flujo de tráfico y del error en los días simulados para esta evaluación. El **flujo de tráfico medio** en una hora en uno de los 46 puntos clave de la ciudad es **643.45**, con un intervalo de confianza asintótico de [627.31, 659.60] (cuando el nivel de significación es 0.05). Por otro lado, el **error medio** en una hora en uno de los 46 puntos clave de la ciudad es **2.31**, con un intervalo de confianza asintótico de [1.82, 2.81]. Con estos dos valores puede verse claramente que el error en la simulación es muy bajo, puesto que en media el tráfico obtenido solo se desvía en 2 del tráfico esperado. Como puede verse en la figura 2.8, tanto el flujo de tráfico como el error siguen distribuciones asimétricas. Se ve que la mayoría de errores son bajos, y que la mayoría de predicciones de tráfico también.

Además, en las figuras 2.9 y 2.10 se muestran los errores relativos por día y hora, junto al tráfico total esperado en los puntos clave de la ciudad. Se muestra que **la tasa de error es muy baja** independientemente del día y la hora, y que nunca supera el 2 %, aunque si es cierto que se observa que **el error es algo inferior en las horas en las que el flujo de tráfico es menor**. Se observan algunos picos en el día 19-12-2019. Este tipo de picos se espera que ocurran en la simulación debido a la naturaleza aleatoria (en cuanto a la distribución del tráfico) del modelo. En definitiva, **el flujo de tráfico dado por el modelo de tráfico es prácticamente idéntico** en los 46 puntos clave de la ciudad.

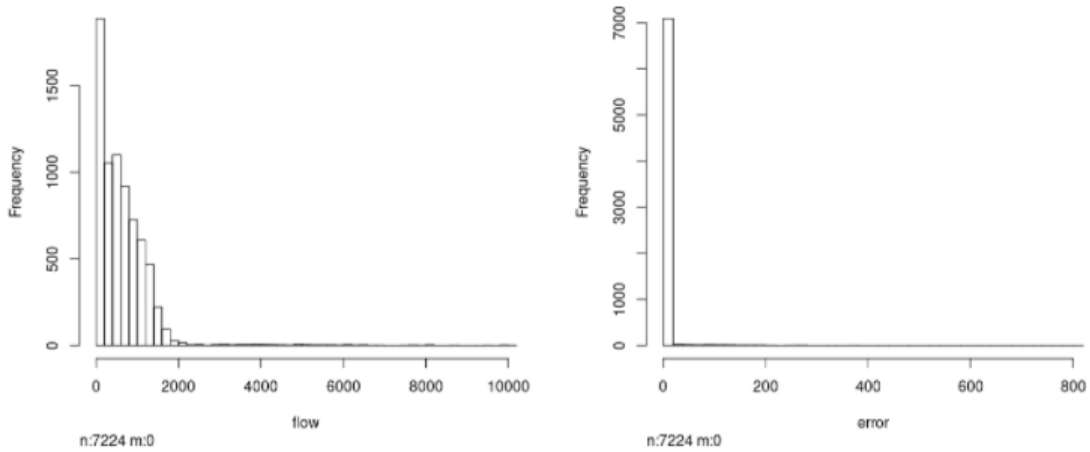

Figura 2.8: Distribución del flujo de tráfico y del error.

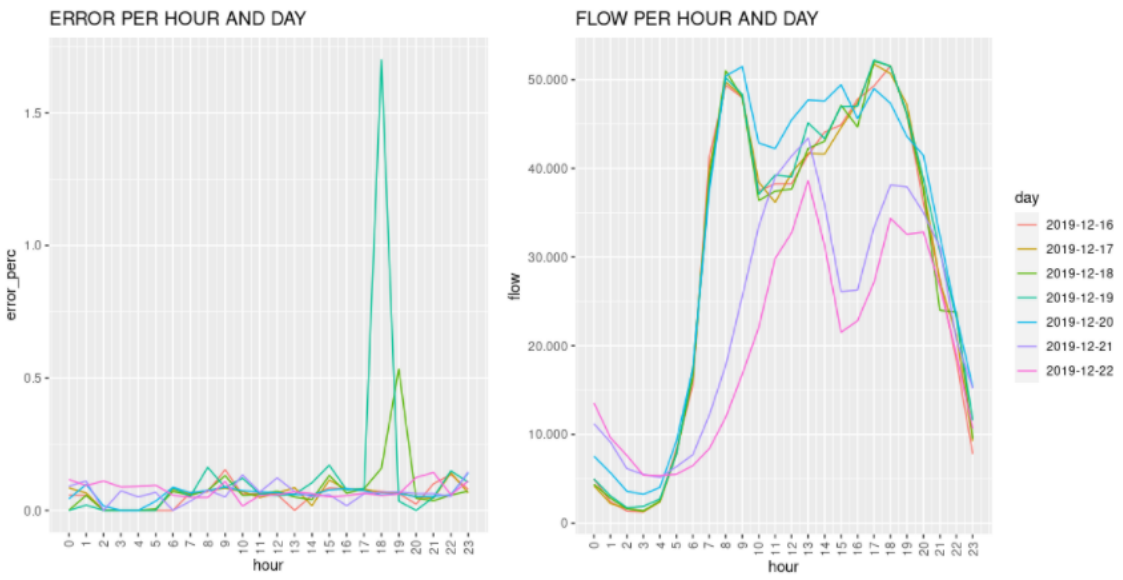

Figura 2.9: Error relativo y flujo de tráfico por hora en cada día.

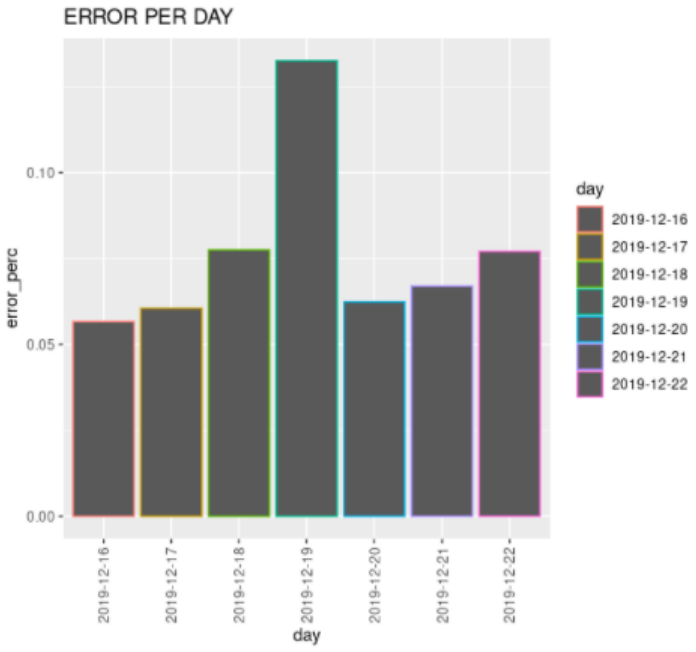

Figura 2.10: Error relativo por día.

2.4. Conclusiones

Se ha logrado alcanzar los objetivos que se plantearon relativos al modelo de tráfico. En primer lugar, se han introducido varias mejoras respecto al modelo de tráfico inicial, destacando el precálculo de las emisiones de NO_x, la posibilidad de generar simulaciones en las que se prohíbe el tránsito de vehículos en determinadas calles, y la mejora sobre la capacidad de predicción del modelo.

En segundo lugar, se ha hecho un análisis de los datos simulados. Para ello, se ha empleado un algoritmo de agrupamiento, que ha permitido determinar que, en general, las calles de la ciudad se agrupan en dos grupos distintos en función de los niveles de contaminación que generan, siendo, los niveles obtenidos más altos en un grupo que en otro. También se ha observado que el flujo de tráfico es mayor de lunes a viernes, y que durante los meses de julio y agosto este disminuye.

Finalmente, se ha realizado una evaluación del modelo, comparando el tráfico que se espera que tuviera la simulación y el tráfico que se ha generado en la simulación. Los resultados indican que el error de simulación es muy bajo y que, por lo tanto, el flujo de tráfico generado en el modelo se aproxima mucho al esperado, observándose que el error nunca supera al 2 %.

Capítulo 3

Calibración de una red de sensores

En este capítulo se va a detallar el proceso que se siguió para calibrar las medidas de diferentes contaminantes atmosféricos obtenidas a través de una red de sensores de bajo coste.

3.1. ¿De dónde se parte?

Dentro del proyecto TRAFAIR se desplegó y configuró una red de sensores de bajo coste. En total se desplegaron y configuraron 10 sensores, que sirven para tomar medidas periódicas de contaminación (cada 10 minutos), lo que permite conocer, en tiempo real, la calidad del aire. Concretamente, los contaminantes que se miden son NO, NO_2, CO y O_3. La arquitectura del sistema del que se parte se ilustra en la figura 3.1. Para más detalles, en el anexo B se detalla cómo se configuró dicha red de sensores.

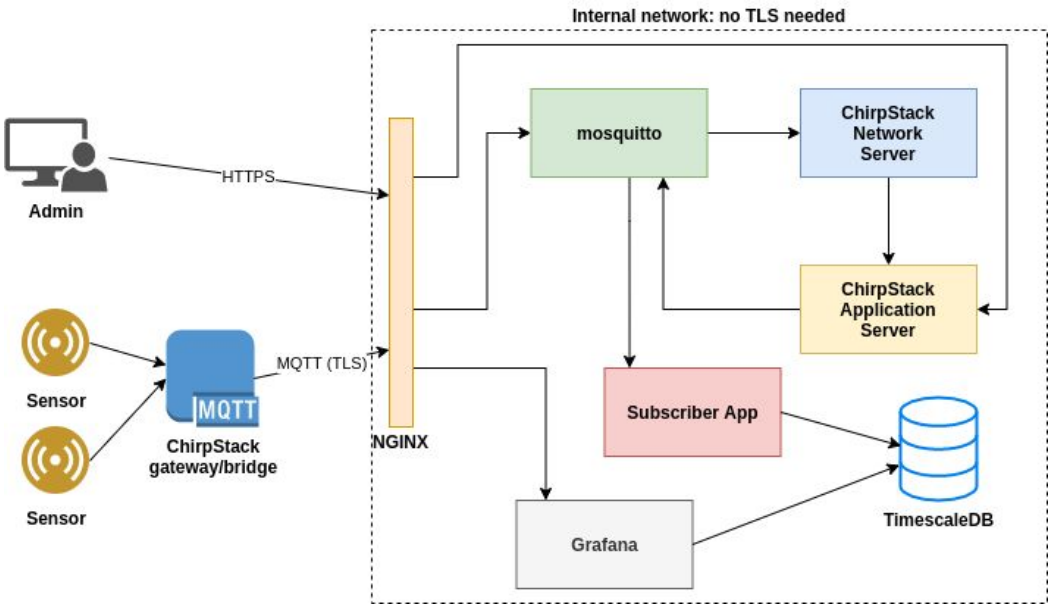

Figura 3.1: Infraestructura del sistema de sensores.

3.2. Calibración de los datos medidos por los sensores

En esta sección se va a detallar qué proceso se ha seguido para calibrar las medidas de contaminación realizadas por una red de sensores de bajo coste.

3.2.1. Datos tomados por los sensores

Los sensores toman medidas cada 10 minutos. Concretamente, cada medida consta de los siguientes datos:

- El **instante temporal** en el que fue tomada la medida. Por ejemplo, 2021-11-01 07:12:51.
- El **voltaje** que le queda a la batería del sensor. Por ejemplo, 5V.
- La **humedad**, entre 0 y 100, que hay en el ambiente. Por ejemplo, 58.2.
- La **temperatura ambiente**. Por ejemplo, 15.1.
- **Dos valores por cada contaminante** (NO, NO_2, CO y O_3) que se obtienen a partir de una serie de electrodos incluidos en las celdas electroquímicas de las que consta el sensor. Los nombres de estos valores son:
  - no_we y no_aux para el monóxido de nitrógeno, NO.
  - no2_we y no2_aux para el dióxido de nitrógeno, NO_2.
  - co_we y co_aux para el monóxido de carbono, CO.
  - o3_we y o3_aux para el ozono, O_3.

Además de estos valores, cada medida también incluye las concentraciones de cada contaminante, que son calculadas internamente por el sensor a partir de los valores que acaban de ser descritos y de acuerdo con una calibración de fábrica.

Sin embargo los valores de concentración que calcula cada sensor son de muy baja calidad, incluso dándose situaciones en las que el sensor calcula concentraciones negativas, lo que evidentemente es imposible. Por lo tanto, se ha descartado el uso de estos valores de concentración y se va a proceder a realizar una calibración más rigurosa de las medidas.

3.2.2. Calibración de las medidas

La calibración de una medida consiste en calcular los valores de concentración real para cada uno de los 4 contaminantes considerados, NO, NO_2, CO y O_3, a partir del resto de valores tomados por el sensor: la humedad, la temperatura ambiente y los 8 valores correspondientes a los electrodos.

Para esta calibración se empleará un *Random Forest* [9]. Este es un algoritmo de aprendizaje supervisado, lo que requiere que para el entrenamiento del modelo se dispongan de los valores de concentración reales para cada contaminante. Es decir, dado el conjunto de medidas realizadas por los sensores que se va a utilizar en el entrenamiento, por cada una de estas medidas es necesario disponer de la concentración real **en el mismo instante temporal** y **en el mismo lugar físico** de la ciudad en el que se tomó la medida.

El ayuntamiento de Zaragoza dispone de 8 estaciones distribuidas por toda la ciudad, encargadas de tomar medidas de contaminación precisas para el NO, NO_2, CO y O_3. Por lo tanto, cada sensor se coloca durante un periodo temporal superior a un mes en la misma ubicación física en la que se encuentra uno de estas estaciones. De este modo se logra tener durante dicho periodo todos los datos que toman los sensores junto a las medidas de concentración reales, lo cual puede emplearse para realizar la calibración y entrenar los modelos de *Random Forest*.

En resumen, **por cada uno de los 10 sensores de contaminación se tratará de generar 4 modelos predictivos**: uno para predecir el NO, uno para predecir el NO_2, uno para predecir el CO y el último para predecir el $O3$.

Cabe destacar que el uso del *Random Forest* fue una decisión que se tomó en consenso en el proyecto TRAFAIR y que no fue tomada en este trabajo. Esta decisión se basó en un estudio [14] en el que se hizo un análisis de qué algoritmo es el más adecuado. Además, en dicho estudio también se concluyó que los hiperparámetros para el algoritmo de número de árboles y número mínimo de medidas para dividir un nodo fueran 50 y 20 respectivamente.

Resultados para el CO

En la tabla 3.1 se muestran los resultados de entrenamiento para el CO. Para cada experimento, los valores para el *rmse* y el *r2* que se muestran son los valores promedio entre los obtenidos en los modelos generados para cada sensor, mediante validación cruzada. El mejor modelo es el que:

– Utiliza como variables de entrada la temperatura, la humedad, el *co_we* y el *co_aux*.

– Normaliza estas variables de entrada.

El valor de 0.13 para el *r2* nos dice que hay cierta correlación entre las predicciones y la realidad, aunque dista mucho de 1 que implicaría correlación perfecta.

experiment	pollutant	normalized	rmse	r2
rf with with_temp_hum	co	TRUE	0.08	0.13
rf with with_temp_hum	co	FALSE	0.08	0.12
rf with only_aux	co	TRUE	0.09	-0.27
rf with only_aux	co	FALSE	0.09	-0.27
rf with default	co	TRUE	0.09	-0.26
rf with default	co	FALSE	0.09	-0.26

Tabla 3.1: Resultados de entrenamiento para el CO

Resultados para el NO

En la tabla 3.2 se muestran los resultados de entrenamiento para el NO. Para cada experimento, los valores para el *rmse* y el *r2* que se muestran son los valores promedio entre los obtenidos en los modelos generados para cada sensor, mediante validación cruzada. El mejor modelo es el que:

– Utiliza como variables de entrada la temperatura, la humedad, el *no_we* y el *no_aux*.

– No normaliza estas variables de entrada.

El valor de -0.08 para el *r2* nos indica que la capacidad predictiva para el *NO* es muy pobre y que realmente el modelo no hace predicciones que puedan ser tomadas en cuenta.

experiment	pollutant	normalized	rmse	r2
rf with with_temp_hum	no	FALSE	12.62	-0.08
rf with with_temp_hum	no	TRUE	12.62	-0.09
rf with only_aux	no	TRUE	16.32	-1.4
rf with only_aux	no	FALSE	16.37	-1.43
rf with default	no	FALSE	13.52	-0.48
rf with default	no	TRUE	13.55	-0.49

Tabla 3.2: Resultados de entrenamiento para el NO

Resultados para el NO2

En la tabla 3.3 se muestran los resultados de entrenamiento para el NO_2. Para cada experimento, los valores para el *rmse* y el *r2* que se muestran son los valores promedio

entre los obtenidos en los modelos generados para cada sensor, mediante validación cruzada. El mejor modelo es el que:

- Utiliza como variables de entrada el *no2_we* y el *no2_aux*.
- Normaliza estas variables de entrada.

El valor de 0.08 para el *r2* nos dice que hay cierta correlación entre las predicciones y la realidad, aunque dista mucho de 1 que implicaría correlación perfecta.

experiment	**pollutant**	**normalized**	**rmse**	**r2**
rf with default	no2	FALSE	14.19	0.08
rf with default	no2	TRUE	14.18	0.08
rf with with_temp_hum	no2	TRUE	14.77	-0.02
rf with with_temp_hum	no2	FALSE	14.81	-0.03
rf with only_aux	no2	FALSE	15.61	-0.1
rf with only_aux	no2	TRUE	15.61	-0.11

Tabla 3.3: Resultados de entrenamiento para el NO2

Resultados para el O3

En la tabla 3.1 se muestran los resultados de entrenamiento para el O_3. Para cada experimento, los valores para el *rmse* y el *r2* que se muestran son los valores promedio entre los obtenidos en los modelos generados para cada sensor, mediante validación cruzada. El mejor modelo es el que:

- Utiliza como variables de entrada el *o3_we*, el *o3_aux*, *no2_we* y el *no2_aux*. También se incluyó el O_3 porque también altera los electrodos asociados al NO_2.
- Normaliza estas variables de entrada.

El valor de 0.54 para el *r2* nos dice que hay mucha correlación entre las predicciones y la realidad.

experiment	**pollutant**	**normalized**	**rmse**	**r2**
rf with default	o3	FALSE	13.85	0.54
rf with default	o3	TRUE	13.82	0.54
rf with with_temp_hum	o3	FALSE	14.15	0.53
rf with with_temp_hum	o3	TRUE	14.09	0.53
rf with only_aux	o3	TRUE	16.16	0.35
rf with only_aux	o3	FALSE	16.15	0.35

Tabla 3.4: Resultados de entrenamiento para el O3

3.3. Conclusiones

Se ha llevado a cabo un proceso de calibración de las medidas de contaminación atmosférica. Para el O_3 la calidad de la calibración lograda es muy buena. Esta calidad disminuye para el CO y el NO_2 y, finalmente, se ha visto que la calidad de la calibración para el NO es mala ya que las métricas obtenidas no han sido satisfactorias.

Capítulo 4

Modelo de dispersiones

En este capítulo se va a detallar el modelo de dispersiones que se ha empleado. El objetivo de este componente es predecir la dispersión de los contaminantes en la atmósfera, en función de las condiciones meteorológicas y de las emisiones generadas por distintas fuentes de contaminación.

Una de estas fuentes es el tráfico rodado, el cual se calculará haciendo uso del modelo de tráfico explicado en secciones anteriores.

Se parte con un modelo de dispersiones ya desplegado, por tanto se tratará de mejorar dicho modelo.

La idea es que este modelo esté en constante funcionamiento para disponer de la predicción de la dispersión de NO_x día a día.

4.1. El modelo lagrangiano de Graz

Para estimar la dispersión de los contaminantes en la atmósfera se ha utilizado un modelo de dispersión de partículas lagrangiano. Concretamente, se ha utilizado el modelo lagragiano de Graz[1] (GRAL). Este software combina un modelo lagraginano y un modelo de campo de flujo a microescala para predecir la dispersión de partículas en la atmósfera, y como resultado de la simulación genera los valores de concentraciones de los contaminantes a lo largo de la ciudad.

El principio básico de este modelo es el seguimiento de un conjunto ficticio de partículas moviéndose en determinadas trayectorias dentro de un campo de viento 3D. Además, se emplea un modelo de campo de flujo a microescala para predecir el flujo alrededor de obstáculos, como los edificios de la ciudad. Esto permite que en las simulaciones pueda tenerse en cuenta el modelo 3D de la propia ciudad, para ver cómo afecta a la dispersión de contaminantes junto a la predicción meteorológica del viento.

GRAL proporciona dos formas de computar una simulación:

[1] `https://gral.tugraz.at/`

– *Steady-state mode*. Se hace el seguimiento de las partículas hasta que estas abandonan el dominio del modelo independientemente del tiempo que necesiten para hacerlo. Por ejemplo, una partícula dejaría de seguirse cuando se aleja lo suficiente de la ciudad.

– *Transient mode*. El seguimiento de las partículas se hace hasta que se cumple un determinado tiempo definido por el usuario. Esto permite, por ejemplo, llevar a cabo una simulación de cómo será la dispersión de contaminantes de un determinado día, a una determinada hora. Este modo es el que se utilizará en el presente trabajo, puesto que la idea es simular intervalos de tiempo finitos, como un día.

En cuanto a los parámetros de entrada del modelo necesarios para realizar una simulación, estos pueden dividirse en dos grandes grupos:

– Los parámetros de configuración del modelo. Estos son valores y datos para configurar la simulación. Los más relevantes son:

  - El **modelo 3D de la ciudad**. GRAL necesita de un fichero que almacene la geometría de la ciudad: localización de edificios, anchura, altura...
  - El **dominio**. El dominio indica sobre qué parte del modelo 3D de la ciudad quiere realizarse el seguimiento de partículas. Por lo general el dominio será un cuadrado o un rectángulo bidimensional.
  - El **tamaño de celda**. El dominio se divide en celdas de un determinado tamaño en metros cuadrados. Cuanto más pequeña la celda más precisa será la simulación, pero el coste en tiempo de la simulación aumentará.
  - El **número de partículas por segundo** que entran en la simulación. Como ya se ha comentado, el modelo se basa en el seguimiento de partículas, por tanto cada segundo se generarán X partículas para ser seguidas por el modelo, para así determinar la dispersión de contaminantes. X es el número de partículas que se haya indicado.

– Los grupos contaminantes o *source groups*. Un *source group* se corresponde con una agregación de emisiones de contaminantes que siguen un mismo patrón y que pueden ser moduladas durante diferentes horas del día que está siendo simulado. En este trabajo se emplearán 4 *source groups* distintos:

  - Las **emisiones producidas por el tráfico rodado**, es decir, la salida del simulador de tráfico. Como se verá en secciones posteriores, se definirá más de un *source group* para el tráfico rodado.

- Las **emisiones generadas por las calefacciones domésticas**.
- Las **emisiones generadas por las industrias** que se encuentran dentro de la ciudad.
- Las **emisiones generadas por la depuradora** de la Almozara.
- Las **emisiones generadas por el cementerio** de Torrero.

Además, será necesario indicar la modulación para cada *source group*. La modulación consiste en indicar a qué horas del día que está siendo simulado es mayor o menos la cantidad de polución generada. Todo esto se explicará más en detalle en secciones posteriores.

Finalmente, se va a explicar brevemente cómo realiza las simulaciones GRAL, cuando se dispone de todos los parámetros de entrada necesarios. Para cada día o intervalo temporal que se quiera simular, GRAL divide dicho intervalo en horas. Es decir, GRAL realmente realiza simulaciones de una hora. Para realizar una simulación de una hora, en primer lugar GRAL tiene que calcular el campo 3D del viento en función de las condiciones meteorológicas y la geometría 3D de la ciudad. Una vez hecho esto GRAL realiza la simulación de la dispersión de contaminantes, teniendo en cuenta dicho campo, almacena los resultados en un fichero para su posterior explotación y simula la siguiente hora.

4.2. ¿De dónde se parte?

Como ya se ha comentado, se parte de un modelo de dispersiones ya desplegado, tal y como se ilustra en la figura 4.1. Este modelo se encarga de hacer las predicciones de dispersión de contaminación en función de la predicción meteorológica y de la predicción de tráfico realizada con el modelo de tráfico. Como puede verse el modelo consta básicamente del componente GRAL, que es un software que se encarga de realizar la simulación.

Por tanto se tratará de mejorar este modelo. Concretamente:

– Se dividirá el tráfico en varios *source group* en vez de en uno, que es como estaba configurada la simulación hasta el momento en el que se realizó este trabajo.

– Se integrarán otras fuentes de contaminación.

– Se añadirán una serie de *scripts* para procesar la salida del modelo, calcular métricas y subir los resultados de simulación a *Geoserver*, un servidor de datos espaciales.

Estos 3 puntos se explican en secciones posteriores.

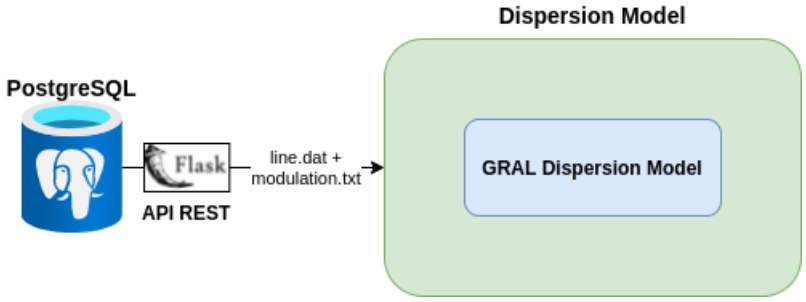

Figura 4.1: Arquitectura inicial del modelo de dispersiones.

4.2.1. Parámetros de configuración del modelo

Los parámetros de configuración que se estaban empleando en la ciudad de Zaragoza y que se seguirán empleando son:

- El modelo 3D de la ciudad. El modelo 3D de la ciudad ha sido extraído del catastro municipal y se ha exportado a un fichero de texto con el formato específico que necesita GRAL para poder utilizarlo.

- El dominio. En Zaragoza se ha considerado un dominio de $8km \cdot 8km = 64km^2$, que cubre la mayor parte del área urbana de Zaragoza. En primer lugar se empezó a utilizar un dominio más extenso, pero este se redujo para reducir el tiempo de cómputo. En la figura 4.2 se ilustra el dominio que se ha empleado en GRAL. En dicha imagen también se ilustra la localización de las estaciones meteorológicas pertenecientes al ayuntamiento de Zaragoza, cuyas medidas se utilizarán posteriormente para evaluar el modelo de dispersiones.

- El tamaño de celda. Se ha decidido emplear 69000 celdas dentro del dominio.

- El número de partículas por segundo que se va a emplear es de 5000, tal y como recomiendan los propios desarrolladores de GRAL, que consideran que con ese número de partículas se consigue un buen balance entre calidad de simulación y tiempo de cómputo.

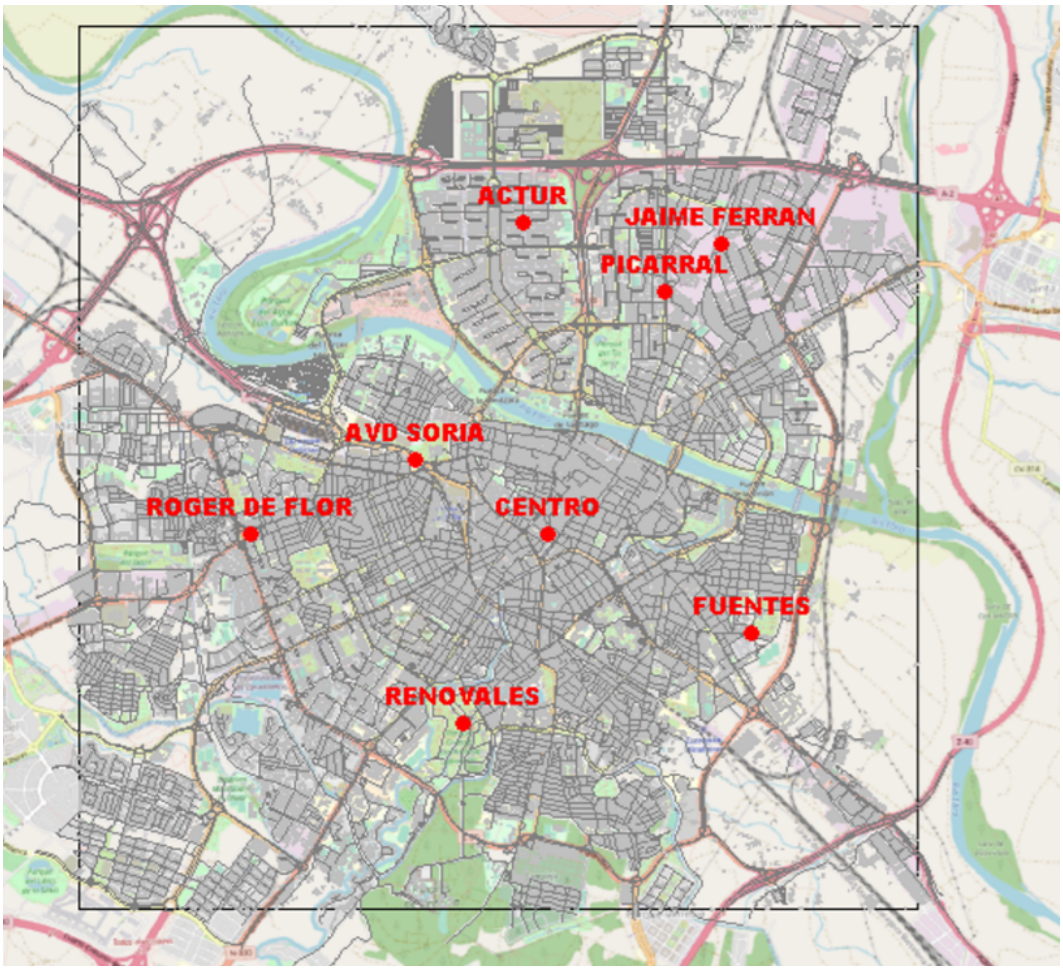

Figura 4.2: Dominio utilizado en GRAL (enmarcado dentro de un cuadrado) y localización de las estaciones de calidad del aire del ayuntamiento de Zaragoza.

4.3. Inclusión de contaminantes en el modelo

En esta sección se va a explicar qué procedimientos se han seguido para la inclusión de las diferentes fuentes de contaminación atmosférica en el modelo de dispersiones.

4.3.1. Introducción

Como ya se ha comentado anteriormente, un *source groups* se corresponde con una agregación de contaminantes que siguen el mismo patrón, y concretamente se han considerado las emisiones producidas por el tráfico rodado, las calefacciones domésticas, las industrias y la depuradora de la Almozara y el cementerio de Torrero, que se encuentran dentro del dominio fijado.

Toda esta información necesaria para llevar a cabo una simulación se encuentra almacenada en una base de datos PostgreSQL. Para facilitar la recuperación y explotación de estos datos, se decidió encapsular el acceso a dicha base de datos mediante un API REST. Además, se crearon una serie de *shell scripts* encargados de recuperar esta información vía web y de alimentar a GRAL con dichos datos. Este flujo de datos se refleja en la figura 4.3.

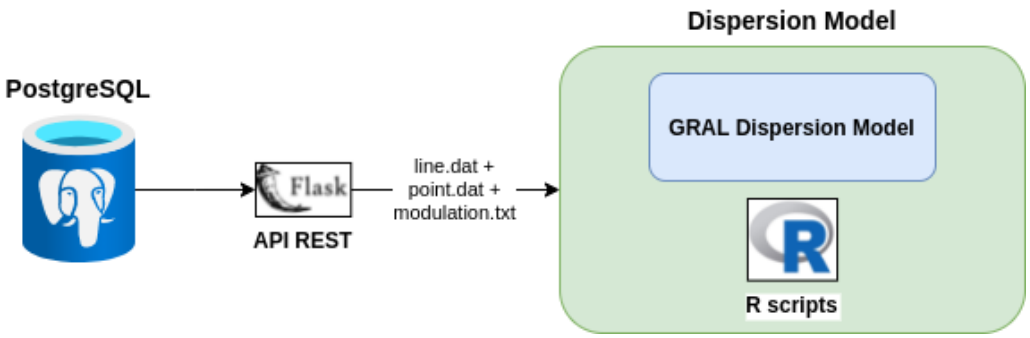

Figura 4.3: Modelo de dispersiones.

Como puede observarse en la figura 4.3, a partir de la información que existe en la base de datos se generan 3 ficheros de entrada para GRAL:

– Fichero **line.dat**. Este fichero contiene la emisión de NO_x por hora como consecuencia del tráfico rodado en cada calle de la ciudad, es decir, contiene el resultado de aplicar el modelo de tráfico, explicado en secciones anteriores, a un día determinado. Por lo tanto, dentro de la simulación de GRAL en la calle en cuestión se generará dicha cantidad de NO_x. Cabe destacar que aunque se simulen 24 horas solo se especifica un valor para la emisión de NO_x por hora, y este valor se modulará a lo largo del día simulado.

– Fichero **point.dat**. Este fichero contiene la emisión de NO_x por hora generada por el resto de fuentes de contaminación consideradas. En este fichero se definen varios puntos en la ciudad y la emisión por hora de NO_x para cada punto. Por tanto, dentro de la simulación en el punto en cuestión se generará dicha cantidad de NO_x. Como en el caso anterior, aunque se simulen 24 horas solo se especifica un valor para la emisión de NO_x por hora.

– Fichero **modulation.txt**. Cada calle o punto que se ha definido en los ficheros *line.dat* y *point.dat* tiene asociado un *source group* y una emisión de NO_x por hora. Pues bien, en este fichero se modula la cantidad de emisión generada por hora. Por ejemplo, si la emisión de NO_x para una calle que se ha definido en el fichero *line.dat* es 5 gramos, y se ha especificado que la modulación es 0.5 a las 6.00 y 2 a las 7.00, significa que en esa calle se generarán en la simulación 2.5 gramos a las 6.00 y 10 gramos de NO_x a las 7.00.

4.3.2. Primer *source group*: emisiones generadas por las viviendas

Las emisiones generadas por las viviendas se constituyen básicamente por las emisiones generadas por las calefacciones domésticas. Al proyecto TRAFAIR se le han entregado datos relativos al inventario de emisiones de NO_x realizado por el ayuntamiento de Zaragoza. Concretamente, el dato que se proporcionó es que en el año 2015, **318.55 toneladas** de NO_x fueron generadas por las viviendas. Por tanto, dicha información se distribuirá tanto espacial como temporalmente a lo largo de la ciudad y del año para calcular las emisiones generadas por las calefacciones día a día. Se considerará que las emisiones actuales generadas por las viviendas siguen siendo las mismas actualmente.

En primer lugar hay que **distribuir estas emisiones a lo largo del tiempo**, es decir, hay que calcular cómo estas 318.55 toneladas se distribuyen día a día a lo largo de un año. Para ello se siguen los siguientes pasos:

1. Se toman los datos de temperaturas máximas y mínimas por día del año 2015. Un ejemplo de estos datos se muestran en la tabla 4.1.

2. Para cada día se calcula el siguiente coeficiente: $coef = max(18 - \frac{T_{max}+T_{min}}{2}, 0)$.

   – Un coeficiente de 0 significa que ese día no se ha encendido la calefacción.

   – Nótese que cuanto menor sea la media entre la temperatura máxima y la mínima ($\frac{T_{max}+T_{min}}{2}$), mayor será el coeficiente. Esto se corresponde con los días más fríos.

3. Se calcula la media de todos los coeficientes. A esto se le llama *mean_coeff*.

4. Se añade un factor de regularización a todos los coeficientes: $coef = coef + 0{,}2 \cdot mean\_coef$. Así por ejemplo se tienen en cuanta situaciones como que un día de verano, en el que en principio no se enciende la calefacción, tenga emisiones asociadas. Por ejemplo, porque se ha encendido el calentador en dicha vivienda para tener agua caliente.

5. Se normalizan todos los coeficientes: $coef = \frac{coef}{mean(coef)}$.

6. Se calcula el NO_x asociado a cada coeficiente y día: $NO_x = coef \cdot \frac{318{,}55}{365}$.

7. Se almacena en la base de datos el NO_x asociado a cada coeficiente que se calculó inicialmente, en el paso 2. De este modo, cuando se vaya a simular un día, se calcula su coeficiente y se recuperan las emisiones asociadas a dicho coeficiente. En la tabla 4.2 se tiene un ejemplo de cómo son estos datos.

Día	T_{max}	T_{min}
01-01-2015	12.6	-0.9
...	...	...
31-12-2015	11.7	5.2

Tabla 4.1: Temperaturas máximas y mínimas en los días de 2015.

NO_x	Coef
5.1	10
7.45	13

Tabla 4.2: Ejemplo de datos para distribuir las emisiones de las calefacciones a lo largo del tiempo.

Una vez realizada la distribución a lo largo del tiempo puede calcularse para un día determinado la cantidad total de NO_x que las calefacciones generarán en la ciudad. El siguiente paso es **calcular cómo se distribuirá esta emisión por la ciudad**. Para ello se ha implementado un método sencillo:

- Cada edificio de la ciudad se considera que es un foco de emisión.

- Se calcula el volumen que ocupa cada edificio y la suma total de los volúmenes de todos los edificios.

- Para cada edificio se calcula el siguiente coeficiente: $coef = \frac{VOLUMEN_{edificio}}{VOLUMEN_{total}}$.

– La emisión asociada a cada edificio de la ciudad por hora se calcula multiplicando el coeficiente por la cantidad total de emisiones esperadas en el día en cuestión, y dividido por 24.

– La emisión por hora se añade al fichero *point.dat*. Nótese que se generará una entrada en dicho fichero por cada edificio de la ciudad.

Modulación

La modulación que se ha empleado para cada hora es la que se refleja en la tabla 4.3. Como no se disponía de ningún dato a partir del cual basar esta modulación, en el proyecto TRAFAIR se llegó al consenso de emplear esta modulación. Como puede verse, se ha tratado de que la modulación sea menor en horas de madrugada y mayor en horas en las que se supone que hay mayor consumo, como las 19.00 de la tarde. Nótese que la suma de todos los valores de modulación es 24, para que la emisión total del día sea la que se calculó en el proceso previamente descrito ($emision\_dia = 24 \cdot (0{,}41 + 0{,}41 + ... + 1 + 0{,}4) \cdot NOxEnLineDat$).

Hora	***Source group***	**Modulación**
0	Domestic heating	0.41
1	Domestic heating	0.41
2	Domestic heating	0.41
3	Domestic heating	0.41
4	Domestic heating	0.41
5	Domestic heating	0.5
6	Domestic heating	1.2
7	Domestic heating	1.5
8	Domestic heating	1.56
9	Domestic heating	1.54
10	Domestic heating	1.4
11	Domestic heating	1.15
12	Domestic heating	1.1
13	Domestic heating	1
14	Domestic heating	0.98
15	Domestic heating	0.95
16	Domestic heating	0.9
17	Domestic heating	1.2
18	Domestic heating	1.4
19	Domestic heating	1.5
20	Domestic heating	1.35
21	Domestic heating	1.32
22	Domestic heating	1
23	Domestic heating	0.4

Tabla 4.3: Modulación para las emisiones producidas por las calefacciones.

4.3.3. Segundo *source group*: emisiones generadas por las industrias

Para este tipo de emisiones solo se han considerado las industrias que se encuentran dentro del dominio que se ha definido con anterioridad. Para ello, se ha tomado como base el mapa de emisiones industriales del año 2006, el cual se ilustra en la figura 4.4 y que refleja las toneladas anuales de NO_x generadas por las industrias.

De todos los focos de la figura 4.4 solo se considerarán los 3 focos más emisores, es decir, los 3 que tienen tonos más oscuros. Debido a que no se dispone de valores concretos para cada foco, se toma el punto medio del intervalo que puede verse en la leyenda en función del color. Por tanto, los valores tomados en los 3 focos son **105.5, 295.75 y 681.5 toneladas al año**.

Una vez se tiene la cantidad total de emisiones, el siguiente paso es **distribuir dicha emisión a lo largo del tiempo** para cada uno de los 3 puntos. En este *source group* la distribución es muy sencilla. Se ha considerado que las industrias situados en los focos funcionan todos los días del año las 24 horas, por tanto la emisión de un día simplemente consiste en dividir la emisión total en dicho foco por 365. Finalmente la emisión por día se divide entre 24 y se añade al fichero *point.dat*. En este caso como solo se han considerado 3 focos, solo se añaden 3 entradas.

Modulación

La modulación en este caso es muy sencilla. Como se ha considerado que las industrias trabajan las 24 horas, la modulación de cada hora es simplemente 1. En la tabla 4.4 se ejemplifica esta información. Como en el caso anterior, la suma de los valores de modulación es 24.

Hora	***Source group***	**Modulación**
0	Industries	1
...	...	...
23	Industries	1

Tabla 4.4: Ejemplo de modulación para el *source group* de industrias.

4.3.4. Tercer y cuarto *source group*: emisiones generadas por la gestión de residuos

Para este tipo de emisiones se han considerado únicamente dos focos, considerándose un source group distinto para cada uno de ellos:

– **La depuradora de la Almozara**. En el año 2015 las emisiones totales de NO_x generadas en el tratamiento de aguas residuales fueron 0.069 toneladas. Se

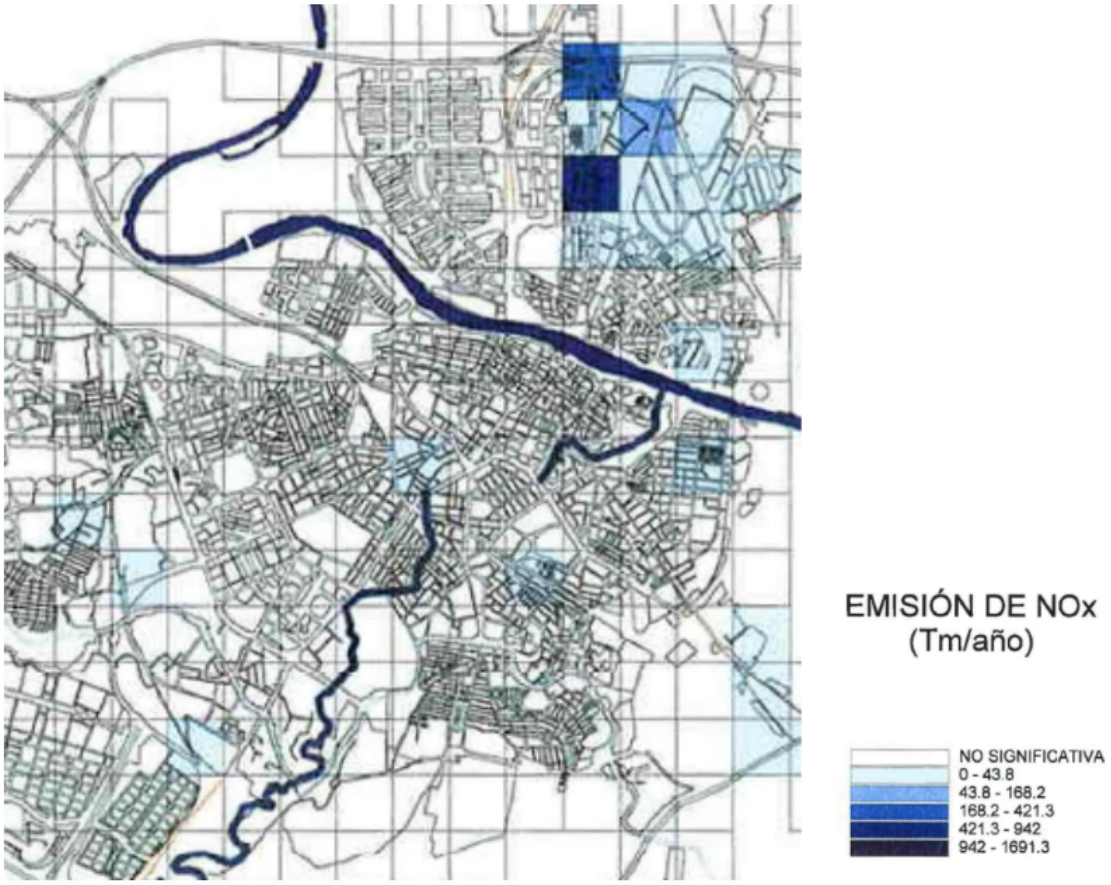

Figura 4.4: Mapa de emisiones industriales en 2006.

asumirá que las emisiones de NO_x siguen siendo las mismas y que se distribuirán homogéneamente a lo largo del tiempo, es decir, aproximadamente este foco emitirá unos 0.19 kg de NO_x diarios. Finalmente, este valor se divide por 24 para obtener la emisión por hora de NO_x.

- **El cementerio de Torrero**. En el año 2015 las emisiones totales de NO_x generadas en los procesos de cremación y por el uso de gas natural fueron 3.07 toneladas. Se asumirá que las emisiones de NO_x siguen siendo las mismas y que se distribuirán homogéneamente a lo largo del tiempo, es decir, aproximadamente este foco emitirá unos 8.41 kg de NO_x diarios. Finalmente, este valor se divide por 24 para obtener la emisión por hora de NO_x.

Modulación

En cuanto a la depuradora de la Almozara se ha asumido que esta funciona las 24 horas del día, por tanto la modulación será 1 para todas las horas. Por otro lado, para el cementerio de Torrero se ha considerado que este funciona 12 horas al día, de 08.00 a 20.00. Por tanto, la modulación será 2 de 08.00 a 20.00 y 0 el resto de horas. Como en los casos anteriores, la suma de los valores de modulación para ambos s*ource group* es 24.

4.3.5. *Source groups* relativos al tráfico rodado

Las emisiones generadas por el tráfico rodado se corresponden con las estimadas por el modelo de tráfico que se ha detallado en secciones anteriores. Además se consideran varios *source groups* para el tráfico rodado. Estos grupos son los que se obtienen empleando el algoritmo de *clustering* que se ha explicado en secciones anteriores.

Cabe destacar que cuando se empezó este proyecto las simulaciones ya tenían en cuenta el tráfico rodado. Sin embargo, únicamente suponían un *source group*. Por tanto dicho *clustering* también se realizó con el objetivo de poder hacer esta división en el modelo de emisiones.

Las emisiones relativas al tráfico se incluyen en un fichero llamado *line.dat*. En dicho fichero se tiene que especificar para cada calle cuánto NO_x emite por hora. Como para la simulación de un día se tienen 24 horas, se ha decidido que el valor que se utilizará para cada calle es la emisión en dicha calle en la hora punta del día, siendo la hora punta del día la hora en la que la suma total de la contaminación producida por todas las calles es mayor.

Respecto a la **modulación**, para hora y *source group* se calcula dividiendo la suma total de emisiones de NO_x de las calles pertenecientes al *source group* a esa hora entre

la suma total de emisiones de NO_x pertenecientes al *source group* en la hora punta.

4.4. Funcionamiento del modelo

En la figura 4.5 se ilustra el aspecto final del modelo de dispersiones. Los pasos que se siguen para llevar a cabo una simulación son los siguientes:

- Se especifica el día a simular. Por ejemplo, el 01-10-2021.
- Se especifican los parámetros de configuración: dominio, mapa de la ciudad, número de celdas... Por lo general serán siempre los mismos.
- El modelo de dispersiones se comunica con la API REST, y esta se encarga de recuperar:
  - Los datos relativos al día a simular y generar los ficheros de entrada *line.dat*, *point.dat* y *modulation.txt*.
  - La predicción meteorológica para el día en cuestión.
- Se realiza la simulación de las 24 horas.
- Se procesa la salida del modelo utilizando una serie de scripts en lenguaje R y se almacena en Geoserver.

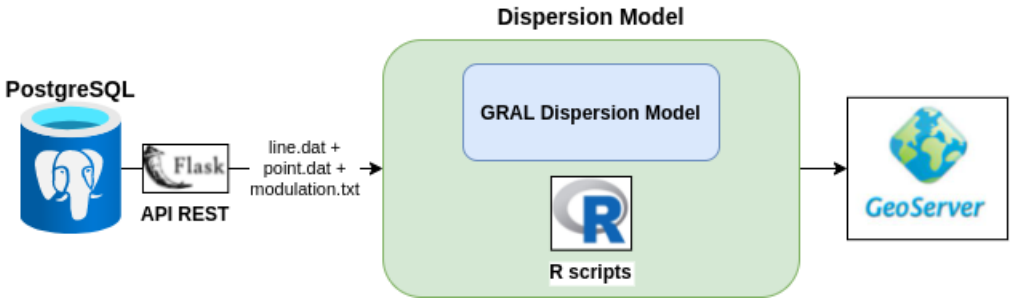

Figura 4.5: Aspecto final del modelo de dispersiones.

Por otro lado, cabe destacar que **las simulaciones son bastante costosas** tanto en tiempo como en espacio. En tiempo cuesta unas 12 horas de cómputo simular 24 horas de dispersiones, y en espacio este proceso ocupa unos 60 GB de memoria RAM. Por tanto, ha sido necesario utilizar HPC (*High Performance Cluster*) HERMES perteneciente al I3A (Instituto Tecnológico de Aragón) para realizar estas simulaciones.

Cada día se llevan a cabo dos simulaciones, una para el día siguiente y otra para dentro de dos días. Es decir, cada día se simula dos veces pero lo que se cambia es la predicción meteorológica, ya que se recupera día a día desde MeteoGalicia. De este

modo la segunda predicción de un día siempre será más precisa puesto que la calidad de la predicción meteorológica será mayor. Finalmente, en la figura 4.6 se muestra el aspecto que tiene la salida de GRAL procesada para poder ser visualizada en un mapa.

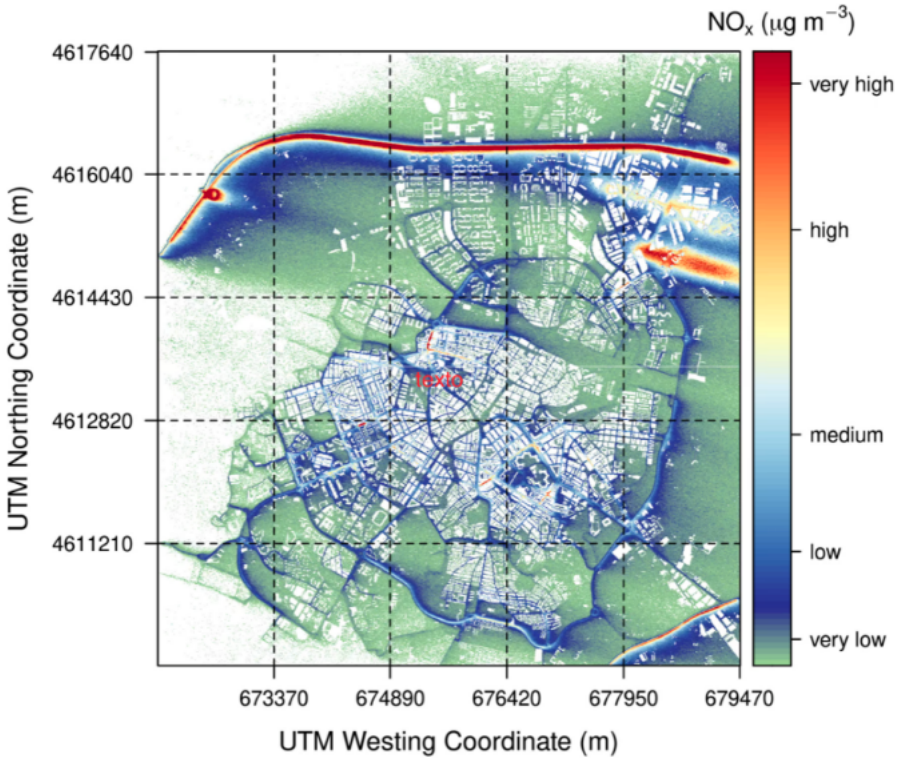

Figura 4.6: Ejemplo resultados obtenidos con GRAL.

4.5. Evaluación del modelo

Para llevar a cabo la evaluación del modelo se han realizado simulaciones en el período comprendido entre el 21 de septiembre de 2020 y el 1 de marzo de 2021. Estas simulaciones se compararán con las observaciones de las estaciones calidad del aire del ayuntamiento de Zaragoza (AQM) y con las observaciones de los sensores de bajo coste para las cuales previamente se ha llevado a cabo un proceso de calibración.

4.5.1. Comparación con las estaciones AQM

Las concentraciones de NO_x obtenidas en las simulaciones se comparán con las observaciones registradas en ocho estaciones AQM urbanas, cuya ubicación, código identificador y tipo de ubicación donde se han colocado se detallan en la tabla 4.5. Una estación AQM es una estación de contaminación perteneciente al ayuntamiento de Zaragoza, y para las cuales se disponen de todo el histórico de medidas de NO_x.

Name/location	Code (DB)	Site type
Actur	40	Urban background
Avda. de Soria	39	Urban traffic (intense)
Centro	38	Urban traffic (moderate)
El Picarral	26	Urban traffic(moderate)
Jaime Ferrán	32	Suburban traffic (moderate)
Las Fuentes	37	Urban traffic (intense)
Renovales	36	Urban background
Roger de Flor	29	Urban traffic (intense)

Tabla 4.5: Estaciones de contaminación y su localización.

En las figuras 4.7, 4.8, 4.9 y 4.10 las concentraciones horarias de NOx estimadas por GRAL se comparan mediante diagramas de dispersión con todas las estaciones AQM en Zaragoza. En estas figuras, la línea continua representa una concordancia perfecta con las observaciones y, dentro de las líneas discontinuas, los resultados y las observaciones del modelado concuerdan con un factor de dos y 0.5. Como puede observarse la concordancia de las medidas reales con las simuladas no es al 100 % real. Por lo general, se observa que el modelo tiende a hacer estimaciones por debajo de las reales.

En la figura 4.11 se ilustra el Diagrama de Taylor, que fue generado con el paquete openair de R [15]. Este gráfico permite la visualización de tres indicadores estadísticos diferentes:

- El coeficiente de correlación de Person, r. En la figura se observa que por lo general este valor se sitúa entre 0.2 y 0.4 para todas las estaciones. Esto quiere decir que efectivamente existe correlación entre la predicción y las medidas reales, si bien es cierto que este factor está lejano de valor 1 que implicaría correlación perfecta.

- El error cuadrático medio, RMSE. En la figura se observa que por lo general este valor se sitúa entre 10 y 15 para todas las estaciones. Esto casa con lo que se ha visto en los diagramas de dispersión.

- La desviación estándar el modelo. En la figura se observa que por lo general se encuentra por debajo de 10, excepto para las estaciones del Actur y de Jaime Ferrán que es cercana a 20.

Finalmente se realizó una estimación cuantitativa de la concordancia entre las concentraciones simuladas y observadas siguiendo las siguientes métricas estadísticas. Estos resultados se reflejan en las tablas 4.6 y 4.7:

– FAC2, *Factor of 2*. Porcentaje de las predicciones dentro de un factor de 2 de las observaciones. Por lo general es muy bajo porque el modelo tiende a predecir menores valores que los reales. En los *scatter plots* estos sería el porcentaje de puntos que se encuentra por encima de la recta central.

– MB, *Mean Bias*. Como en el caso anterior, es negativo porque el modelo tiende a hacer predicciones de contaminación más pequeñas que las reales.

– MGE, *Mean Gross Error*. Es el error medio de predicción. Por lo general la predicción falla enntre 16 y 20 unidades.

– NMB, *Normalized Mean Bias*. Es el sesgo normalizado. Indica que las predicciones son por lo general más pequeñas que las medidas reales.

– NMGE, *Normalized Mean Gross Error*. Es el error medio normalizado.

– RMSE, *Root Mean Squared Error*. Error cuadrático medio. Como el error medio, nos indica que se está cometiendo un cuadrático entre 20 y 30 elevado al cuadrado

– r, Coeficiente de correlación de Pearson. Por lo general se encuentra entre 0.2 y 0.4, indicando que hay correlación entre las predicciones y las medidas.

– IOA, *Index of Agreement*. Ratio entre el RMSE y el error potencial.

En definitiva, se ha visto que las predicciones hechas por el modelo están correlacionadas con las medidas reales. Sin embargo, el modelo tiende a estimar por debajo los valores reales de contaminación.

Station	**Actur**	**Avenida Soria**	**Centro**	**Picarral**
FAC2	0.26	0.20	0.13	0.17
MB	-3.83	-13.59	-18.64	-16.67
MGE	16.20	14.62	20.22	18.44
NMB	-0.21	-0.67	-0.74	-0.68
NMGE	0.88	0.73	0.81	0.75
RMSE	21.16	18.63	23.73	23.15
r	0.19	0.37	0.21	0.24
IOA	0.04	0.25	-0.03	0.20

Tabla 4.6: Métricas obtenidas evaluando GRAL (1).

Station	Jaime Ferrán	Las Fuentes	Renovales	Roger de Flor
FAC2	0.45	0.12	0.07	0.06
MB	-2.59	-20.67	-17.73	-27.79
MGE	15.29	21.92	18.20	28.14
NMB	-0.12	-0.74	-0.82	-0.85
NMGE	0.70	0.79	0.84	0.86
RMSE	20.78	27.12	21.62	33.24
r	0.20	0.27	0.19	0.30
IOA	0.12	0.17	0.05	0.05

Tabla 4.7: Métricas obtenidas evaluando GRAL (2).

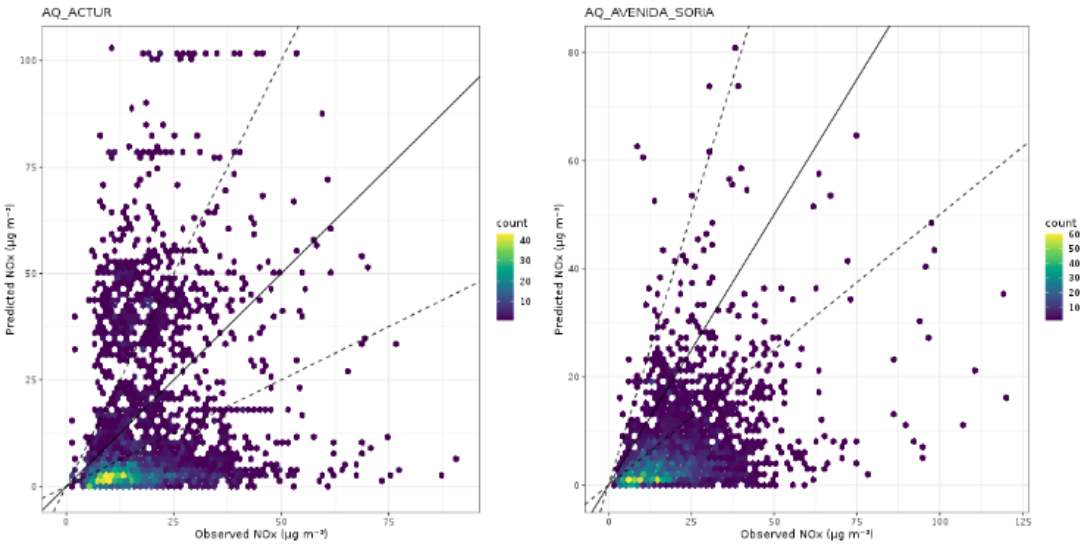

Figura 4.7: *Scatter plot* para las estaciones 40 y 39.

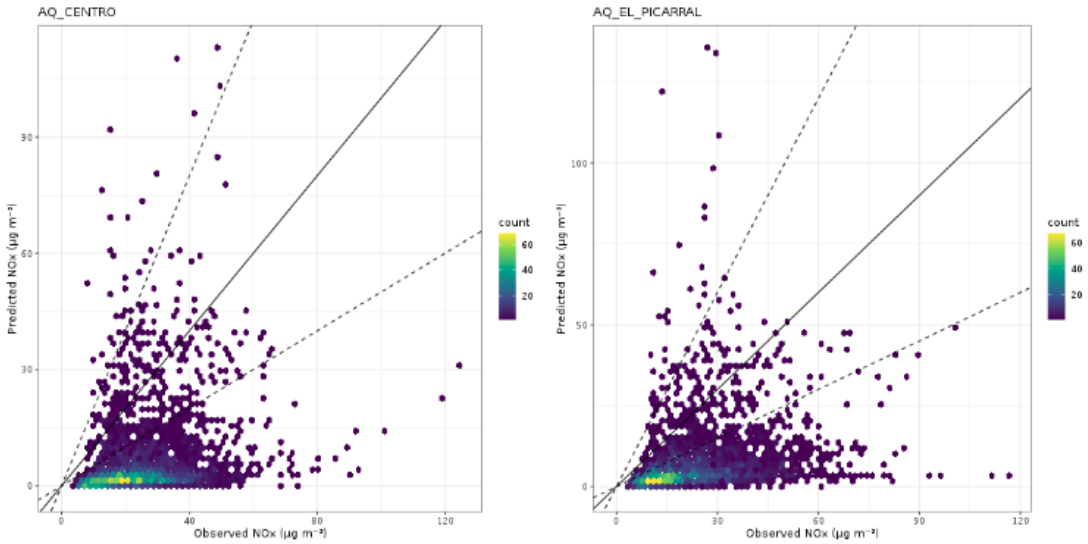

Figura 4.8: *Scatter plot* para las estaciones 38 y 26.

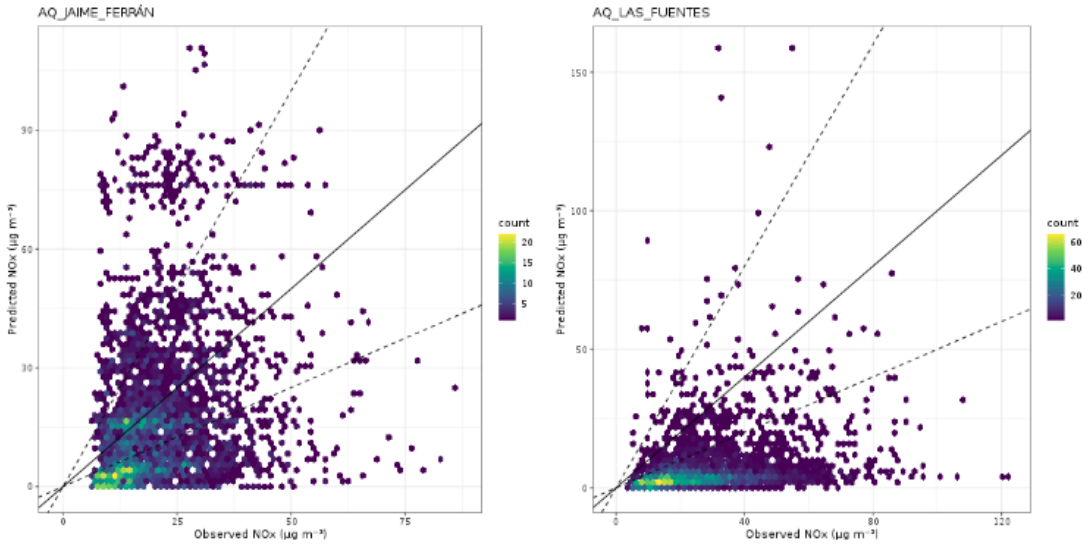

Figura 4.9: *Scatter plot* para las estaciones 32 y 37.

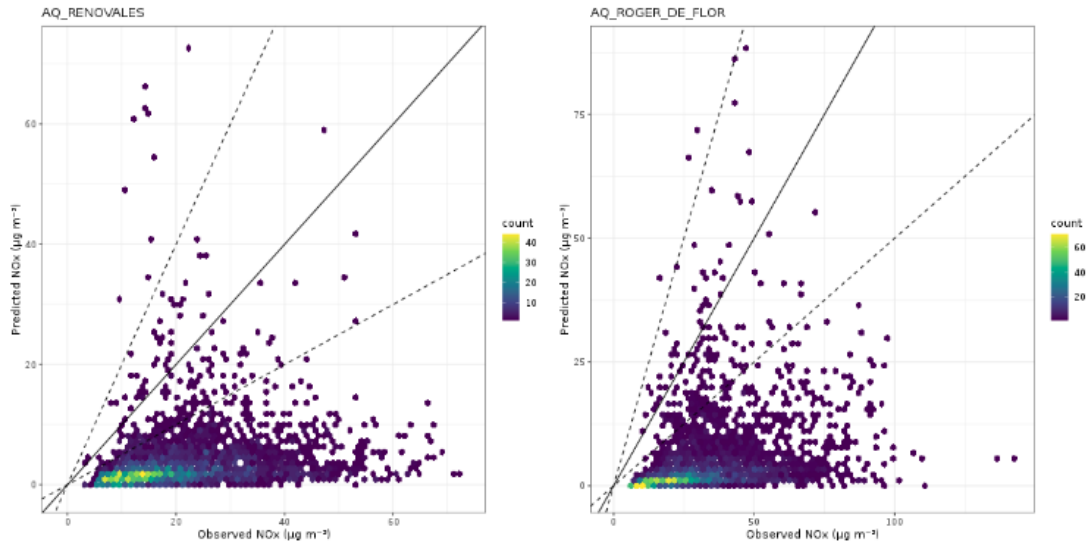

Figura 4.10: *Scatter plot* para las estaciones 36 y 39.

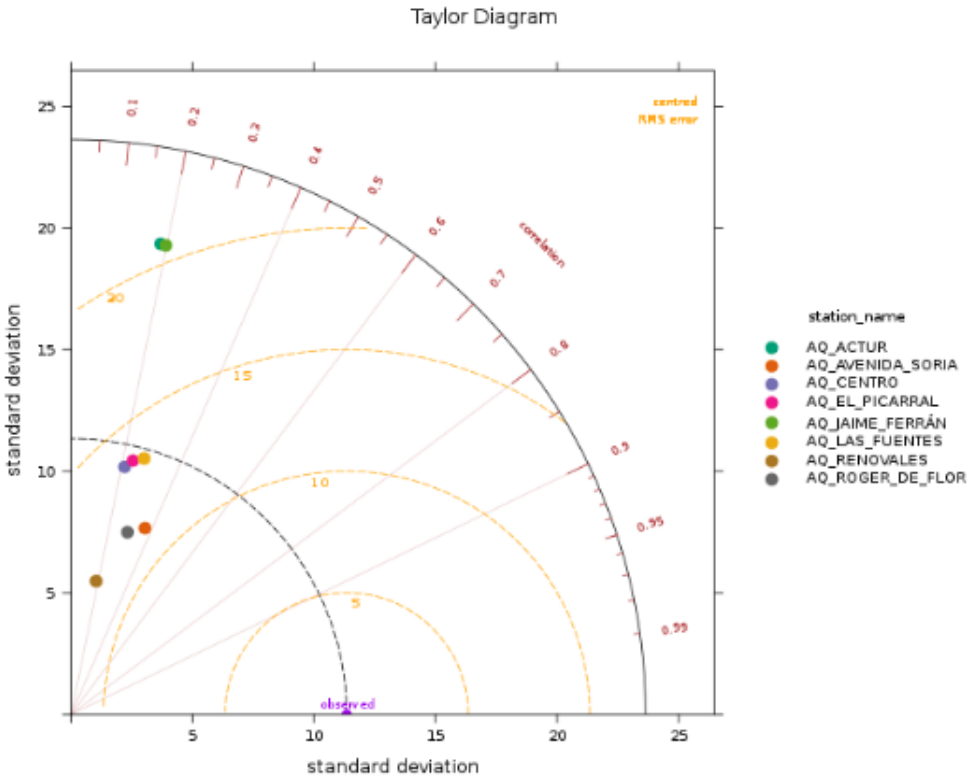

Figura 4.11: Diagrama de Taylor para las estaciones AQM.

4.5.2. Comparación con los sensores de bajo coste

Las concentraciones de NO_x obtenidas en las simulaciones se comparan con los valores medidos con los sensores de bajo coste que se han configurado en este trabajo, cuya ubicación y características se detallan en la tabla 4.8.

Número de sensor	Localización	Tipo de ubicación
4017	Edificio Lorenzo Normante	Urban traffic (intense)
4018	Centro AQM	Urban traffic (moderate)
4019	Edificio Etopía	Urban traffic (intense)
4020	Facultad de Estudios Sociales	Urban traffic (intense)
4021	Centro AQM	Urban traffic (moderate)
4022	Centro AQM	Urban traffic (moderate)
4023	Centro AQM	Urban traffic (moderate)
4024	Centro AQM	Urban traffic (moderate)
4025	Edificio Lorenzo Normante	Urban traffic (intense)
4026	Near highway Z-40	Urban traffic (intense)

Tabla 4.8: Ubicación de los sensores de bajo coste.

En las figuras 4.12, 4.13, 4.14, 4.15 y 4.16 las concentraciones horarias de NOx estimadas por GRAL se comparan mediante diagramas de dispersión con todos los sensores de bajo coste. Como puede observarse la concordancia de las medidas reales con las simuladas no es al 100 % real. Como para las estaciones AQM, se observa que el modelo tiende a hacer estimaciones por debajo de las reales.

En la figura 4.17 se ilustra el Diagrama de Taylor. Este gráfico permite la visualización de tres indicadores estadísticos diferentes:

- El coeficiente de correlación de Person, r. En la figura se observa que por lo general este valor se sitúa por debajo de 0.2 y es cercano a 0, lo que indica que no hay gran correlación.

- El error cuadrático medio, RMSE. En la figura se observa que por lo general este valor se sitúa por debajo de 50.

- La desviación estándar el modelo. En la figura se observa que por lo general se encuentra por debajo de 50.

Finalmente se realizó una estimación cuantitativa de la concordancia entre las concentraciones simuladas y observadas, empleando las mismas métricas que para las estaciones AQM, tal y como se observa en las tablas 4.9 y 4.10. Las conclusiones son las mismas que en la comparación hecho con las estaciones AQM.

En definitiva, se ha visto que las predicciones hechas por el modelo no están muy correlacionadas con las observaciones de los sensores, al contrario de lo que ocurría

con las observaciones de las estaciones. Esto indica claramente, tal y como se dijo en la sección correspondiente a la calibración, que la calibración de los sensores tiene margen de mejora.

Low-cost sensor	4017	4018	4019	4020	4021
FAC2	0.11	0.10	0.08	0.34	0.07
MB	-17.02	-14.14	-20.71	2.70	-16.87
MGE	18.25	14.45	20.81	23.20	17.00
NMB	-0.84	-0.82	-0.86	0.12	-0.84
NMGE	0.90	0.84	0.87	1.03	0.85
RMSE	19.90	16.57	22.52	37.69	18.56
r	-0.03	0.20	-0.04	0.13	0.25
IOA	-0.50	-0.17	-0.42	-0.58	-0.37

Tabla 4.9: Métricas obtenidas evaluando GRAL (3).

Low-cost sensor	4022	4023	4024	4025	4026
FAC2	0.11	0.10	0.08	0.34	0.07
MB	-17.02	-14.14	-20.71	2.70	-16.87
MGE	18.25	14.45	20.81	23.20	17.00
NMB	-0.84	-0.82	-0.86	0.12	-0.84
NMGE	0.90	0.84	0.87	1.03	0.85
RMSE	19.90	16.57	22.52	37.69	18.56
r	-0.03	0.20	-0.04	0.13	0.25
IOA	-0.50	-0.17	-0.42	-0.58	-0.37

Tabla 4.10: Métricas obtenidas evaluando GRAL (4).

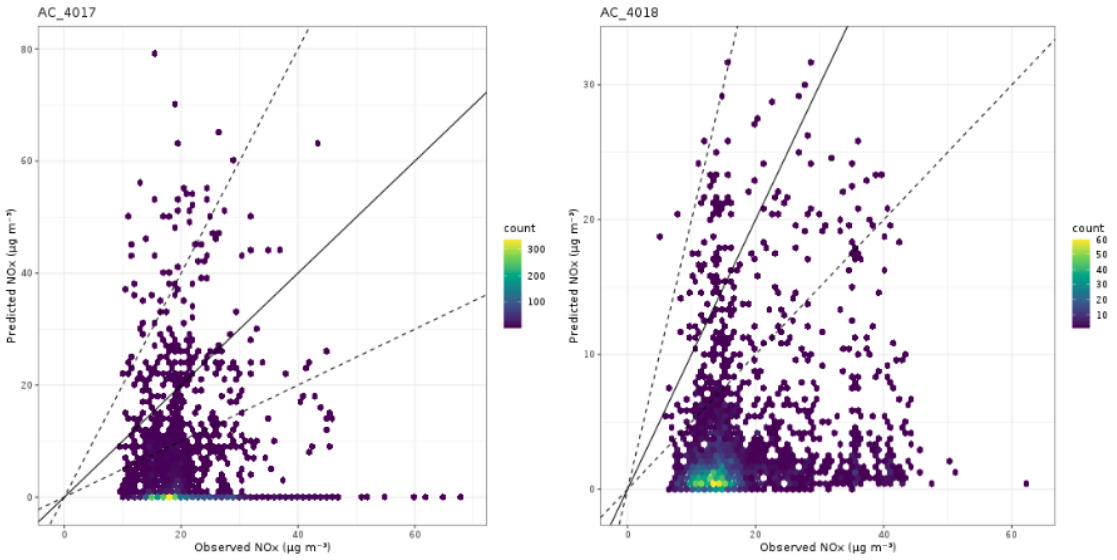

Figura 4.12: *Scatter plot* para los sensores 4017 y 4018.

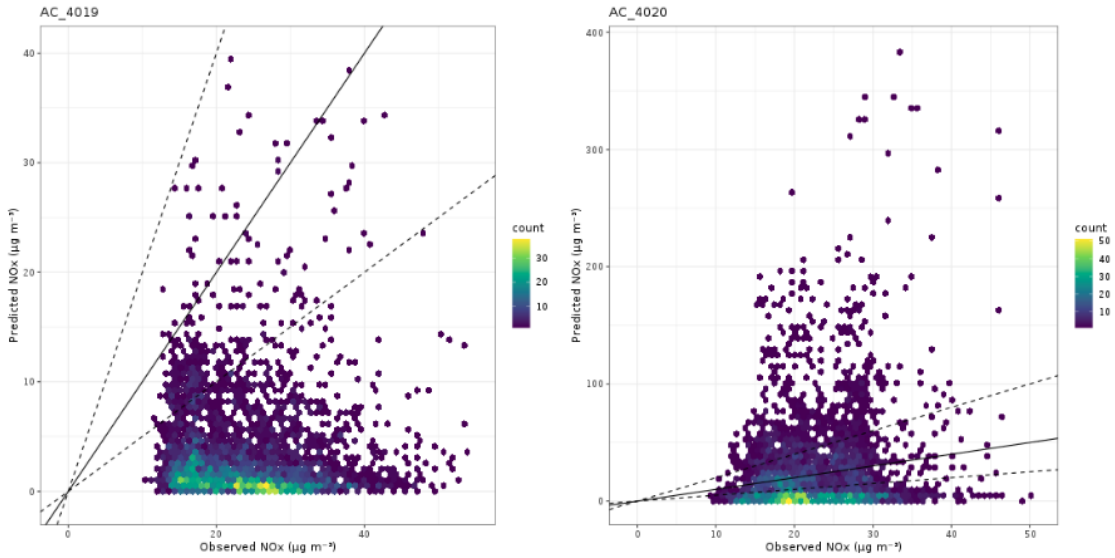

Figura 4.13: *Scatter plot* para los sensores 4019 y 4020.

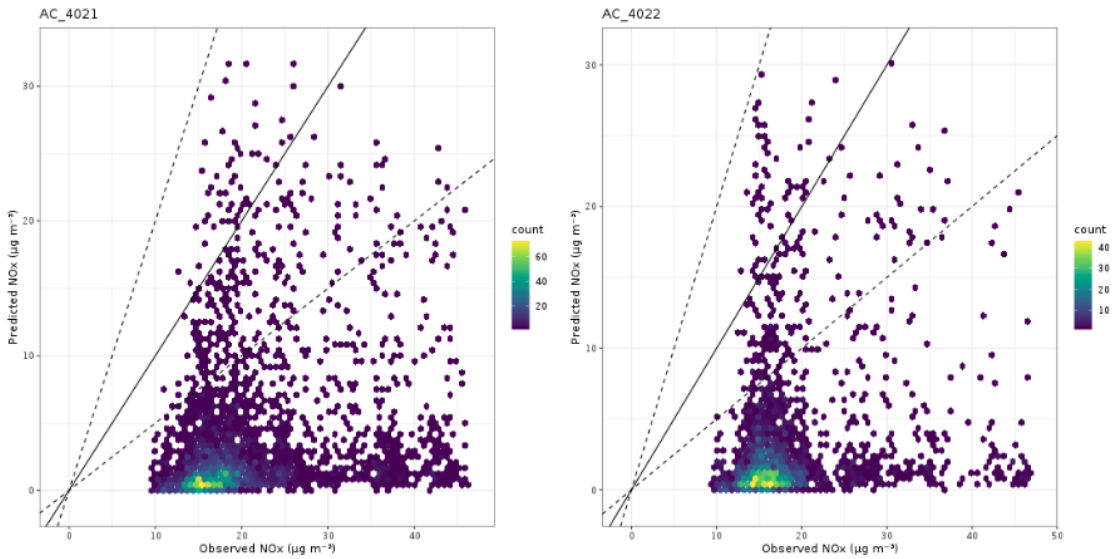

Figura 4.14: *Scatter plot* para los sensores 4021 y 4022.

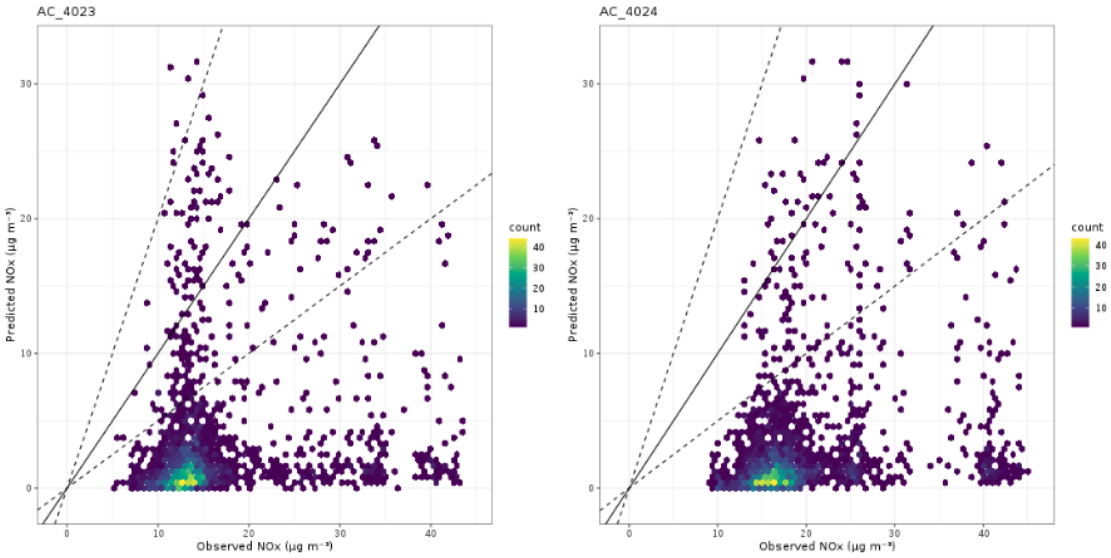

Figura 4.15: *Scatter plot* para los sensores 4023 y 4024.

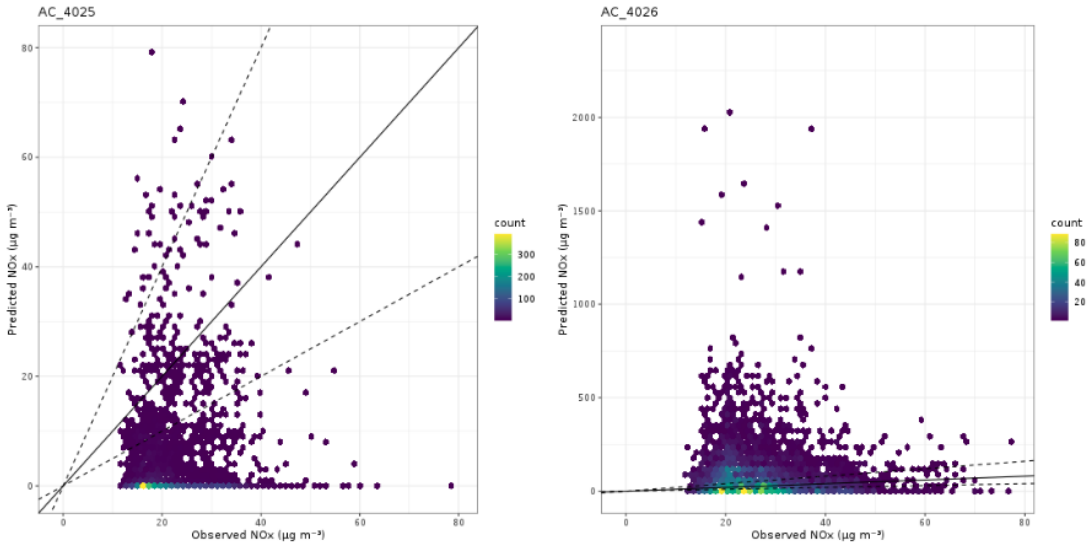

Figura 4.16: *Scatter plot* para los sensores 4025 y 4026.

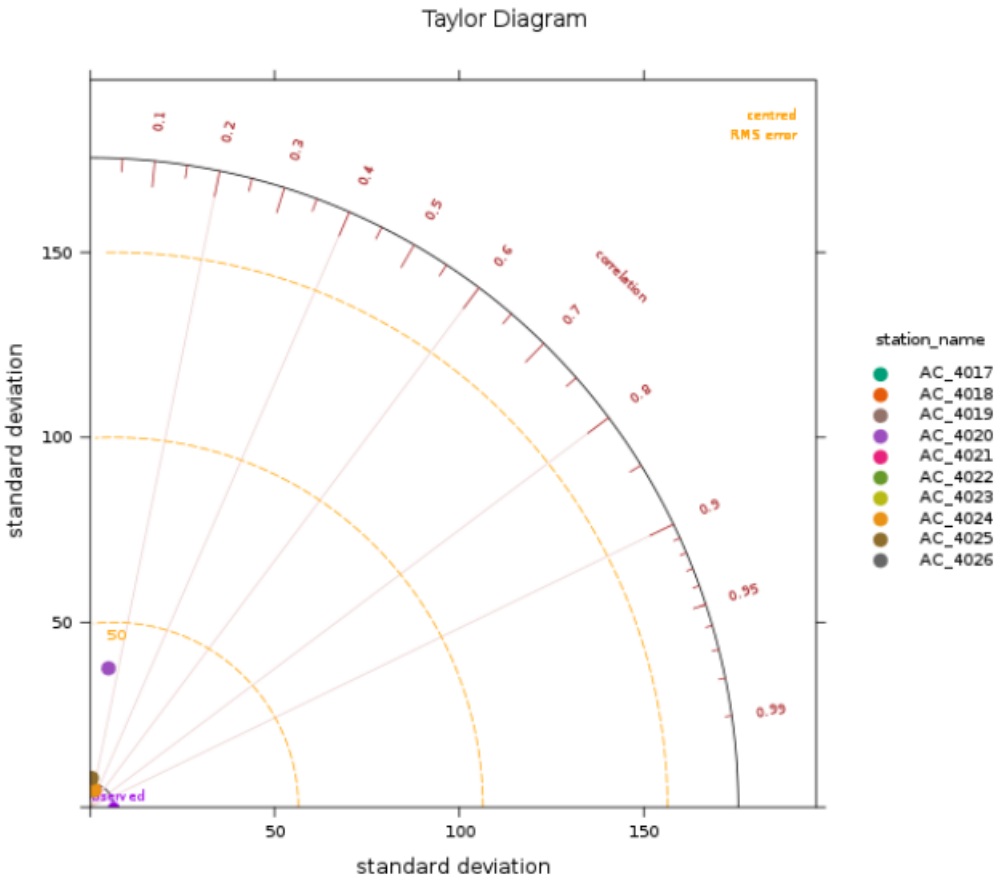

Figura 4.17: Diagrama de Taylor para los sensores.

4.6. Conclusiones

Se ha logrado alcanzar los objetivos que se plantearon relativos al modelo de dispersiones. En primer lugar, se ha mejorado un sistema que utiliza el **modelo lagrangiano de Graz** (*GRAL)* que permite simular la dispersión de contaminantes por la ciudad teniendo en cuenta la predicción meteorológica. Este sistema permite realizar simulaciones día a día para así disponer de la predicción de la dispersión de los contaminantes en la atmósfera.

Posteriormente, se ha detallado el proceso que se ha seguido para poder **integrar en el modelo varias fuentes de emisiones** de NO_x. Concretamente, se han integrado las emisiones generadas por el tráfico rodado, que son obtenidas mediante el modelo de tráfico, las emisiones generadas por las calefacciones de las viviendas, las emisiones generadas por las industrias que se encuentran dentro de la ciudad de Zaragoza, las emisiones generadas por el cementerio de Torrero y las emisiones generadas por la depuradora de la Almozara.

Finalmente se ha realizado una **evaluación del modelo**, comparando las predicciones del modelo con datos reales de contaminación, recogidos mediante una serie de estaciones que pertenecen al ayuntamiento de Zaragoza y mediante una red de sensores de bajo coste. En este análisis se ha visto que existe correlación entre las predicciones del modelo y las medidas reales, aunque es necesario ajustar más el modelo ya que tiende a hacer predicciones por debajo de las medidas reales.

Capítulo 5

Escenarios de tráfico

A lo largo del trabajo se han introducido dos herramientas, el modelo de tráfico y el modelo de dispersión. La primera por sí sola permite predecir el tráfico para un día determinado, y juntas permiten predecir la dispersión de contaminantes en la atmósfera de la ciudad de Zaragoza, considerando además del tráfico diferentes fuentes de emisión de contaminantes.

En este sección, se van a utilizar ambas herramientas para simular una situación hipotética. Concretamente, se simulará cómo varía la cantidad de contaminación en el aire si el porcentaje de vehículos eléctricos aumenta en la ciudad de Zaragoza.

5.1. Escenarios considerados

Para la identificación de nuevos escenarios de tráfico se ha realizado un análisis de futuras medidas relacionadas con la gestión del tráfico en la ciudad de Zaragoza para reducir la contaminación atmosférica. Para ello, se han identificado dos documentos principales impulsados por el ayuntamiento. El primero, corresponde a una revisión del actual Plan de Movilidad Urbana Sostenible (PMUS)[1] que se firmó en 2019 con una vigencia de 8 años, es decir, hasta 2026.

El segundo documento, Estrategia de Cambio Climático, Calidad del Aire y Salud de Zaragoza, Horizonte 2030 (ECAZ 3.0)[2] es una colección de varias medidas en tres direcciones principales: lucha contra el cambio climático, mejora de la calidad del aire y la salud, y eficiencia y reducción del uso de recursos. Considerando ambos documentos, uno de los principales objetivos es incrementar el número de vehículos eléctricos que circulan en la ciudad a un 10 % del total, es decir, alrededor de 35.000 vehículos eléctricos para el 2026, al tiempo que se reducen los vehículos diésel y gasolina más antiguos, que son considerablemente más contaminantes.

[1] `https://www.zaragoza.es/sede/portal/movilidad/plan-movilidad/`

[2] `https://www.zaragoza.es/sede/portal/medioambiente/cambio-climatico/ecaz30/`

Tomando como base los documentos PMUS y la ECAZ 3.0, se han planteado como hipótesis dos escenarios diferentes: uno considerando la composición actual de la flota de vehículos local, y el otro considerando el escenario propuesto para 2026. En la figura 5.1 se muestra el cambio en el porcentaje para cada tipo de vehículo que se espera para 2026.

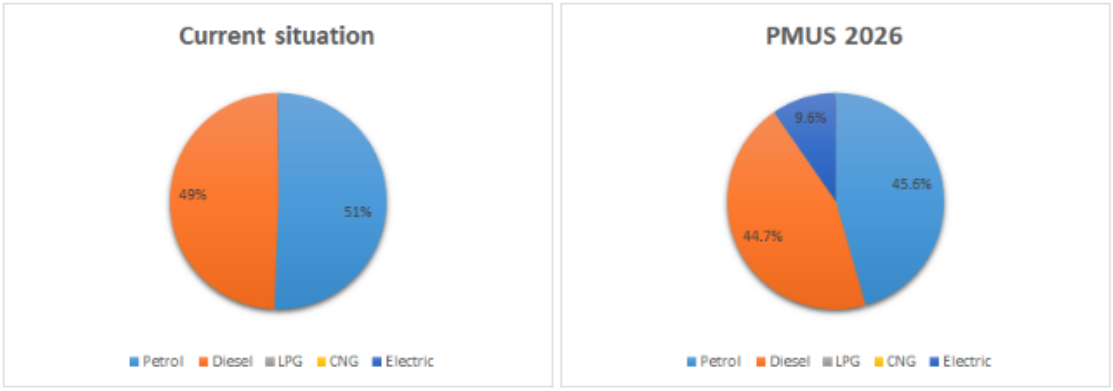

Figura 5.1: Comparación de los tipos de vehículos en ambos escenarios.

5.1.1. Situaciones meteorológicas consideradas

Se han definido diferentes situaciones meteorológicas para comprobar cómo varía la calidad del aire en función de la flota de vehículos considerada y la época del año. Por tanto, las simulaciones de GRAL se han realizado considerando ocho situaciones meteorológicas, es decir dos para cada temporada, una para un día de fin de semana y otra para un día laborable.

Además, se ha realizado un análisis climatológico para identificar diferentes condiciones climáticas representativas para 2019 y 2020, como años de referencia, y aquellas para las que se tiene almacenada información en la base de datos del proyecto. Esta selección se ha realizado con respecto a algunos parámetros meteorológicos: velocidad del viento, radiación solar global y temperatura, y también concentraciones de contaminantes.

Durante la temporada de invierno, los días seleccionados se corresponden con condiciones meteorológicas favorables a la acumulación de contaminación, como baja velocidad del viento, baja radiación solar global, niebla y bajas temperaturas. Los días de verano se caracterizan por condiciones de alta radiación solar, bajas velocidades del viento y altas temperaturas. Para las otras dos temporadas se han seleccionado días intermedios. Los días concretos que se han propuesto son los siguientes:

– Invierno: el 5 de febrero de 2020 (día laborable) y el 9 de febrero de 2020 (fin de semana).

– Verano: el 24 de julio de 2019 (día laborable) y el 29 de julio de 2019 (fin de semana).

– Primavera: el 16 de abril de 2020 (día laborable) y el 19 de abril de 2020 (fin de semana).

– Otoño: el 6 de noviembre de 2019 (día laborable) y el 10 de noviembre de 2020 (fin de semana).

5.1.2. Resultados y conclusiones

En la tabla 5.1 se detalla la reducción de emisiones (%) para el escenario PMUS 2026 con respecto a la composición actual de la flota de vehículos que se ha obtenido en las simulaciones. Como puede observarse, en las 8 situaciones atmósfericas consideradas la reducción es prácticamente la misma, en torno al 9.09 %.

En conclusión, se ha probado la utilidad de las herramientas desarrolladas. Como en este caso, estas pueden emplearse para simular situaciones hipotéticas como qué pasaría si cambia la flota de vehículos, u otras situaciones como qué pasaría si se prohíbe la circulación de tráfico rodado en determinadas zonas de la ciudad.

Season	**Day**	**Day Type**	**PMUS 2026**
Winter	February 5th , 2020	working day	9.092
	February 9th , 2020	weekend	9.091
Spring	April 16th, 2020	working day	9.090
	April 19th, 2020	weekend	9.088
Summer	July 24th, 2019	working day	9.090
	July 29th, 2019	weekend	9.091
Autumn	November 6th, 2019	working day	9.090
	November 10th, 2019	weekend	9.094

Tabla 5.1: Reducción de emisiones (%) para el escenario PMUS 2026 con respecto a la composición actual de la flota de vehículos.

Capítulo 6

TraFlow

En esta sección se va a exponer la aplicación TraFlow. Ésta consiste en una aplicación web que permite visualizar tanto la predicción de tráfico como la dispersión de contaminantes. Es decir, es una herramienta que se utiliza para visualizar los resultados de los modelos de tráfico y de dispersión

Esta aplicación ya fue introducida y desarrollada en [2]. Por lo tanto, en esta sección únicamente se van a detallar las mejoras que se han hecho a dicha aplicación.

6.1. Arquitectura de la aplicación

En [2] se detalló la arquitectura de la aplicación. Esta es la que se ilustra en la figura 6.1. En resumen, consta de 3 componentes principales:

- Una base de datos **PostgreSQL**. Esta es la misma base de datos que viene utilizándose a lo largo de este trabajo.
- Una **API REST**. Esta API implementa una serie de operaciones que permiten recuperar información almacenada en la base de datos relativa a los flujos de tráfico y su contaminación asociada.
- La **interfaz de usuario**, que es el componente con el que interacciona el usuario final y que se encarga de mostrar los mapas.

En el presente trabajo se modificó la arquitectura de la aplicación, tal y como se muestra en la figura 6.2. Como puede observarse, a nivel arquitectural solo se ha añadido un nuevo componente, **Geoserver**[1]. Este se trata de un servidor de datos espaciales, en el que se almacenan los mapas relativos a las simulaciones realizadas por GRAL. Su configuración se llevó a cabo dentro del marco del proyecto TRAFAIR, pero no se engloba en este trabajo.

[1] `http://geoserver.org/`

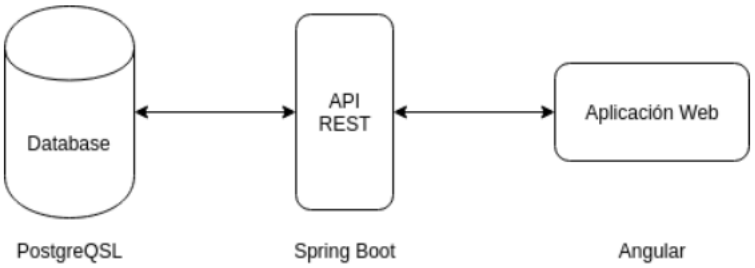

Figura 6.1: Arquitectura antigua de la aplicación TraFlow.

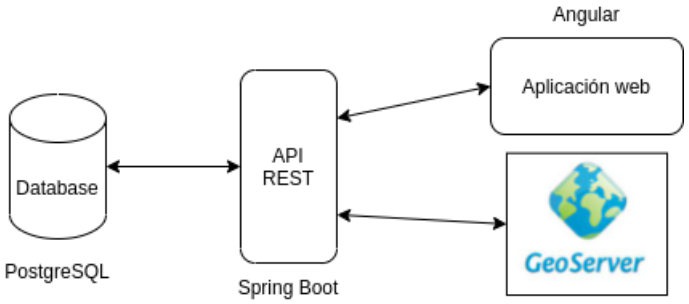

Figura 6.2: Arquitectura nueva de la aplicación TraFlow.

6.2. Nuevas funcionalidades y cambios en la interfaz de usuario

La principal funcionalidad que se le ha añadido a TraFlow es la posibilidad de consultar los mapas de contaminación generados con el modelo de dispersiones. Además, la aplicación también permite visualizar los mapas para las ciudades de Santiago de Compostela y Módena, además de Zaragoza.

La pantalla principal de la aplicación se muestra en la figura 6.3. Como puede observarse, el usuario puede:

- Escoger el día para el que quiere mostrar o bien los mapas de dispersiones de contaminación o bien los flujos de tráfico.
- Escoger la ciudad para la que quiere mostrar el mapa.
- Pulsar o bien el botón asociado a los mapas de dispersiones o bien el botón asociado a los flujos de tráfico.
    - Respecto al mapa de tráfico que se mostraba en [2], el mapa actual es mucho más vistoso, tal y como se ve en la figura 6.5.
    - El mapa de dispersiones se muestra en la figura 6.4. Si se pincha en un punto de un mapa, se mostrará la contaminación registrada en dicho punto. Además, se ha incluído una leyenda.

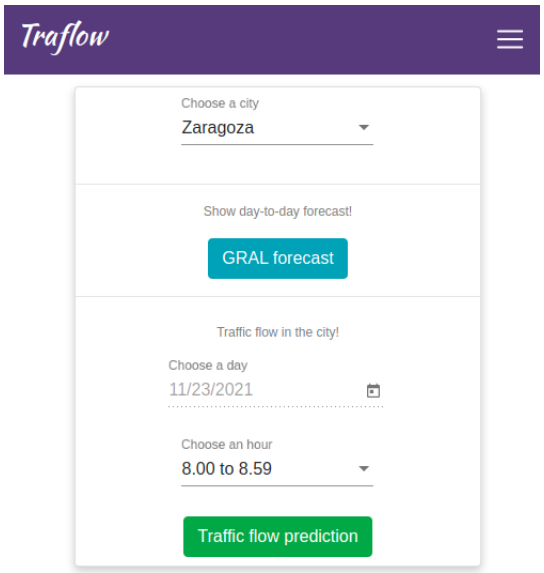

Figura 6.3: Pantalla principal de la aplicación TraFlow.

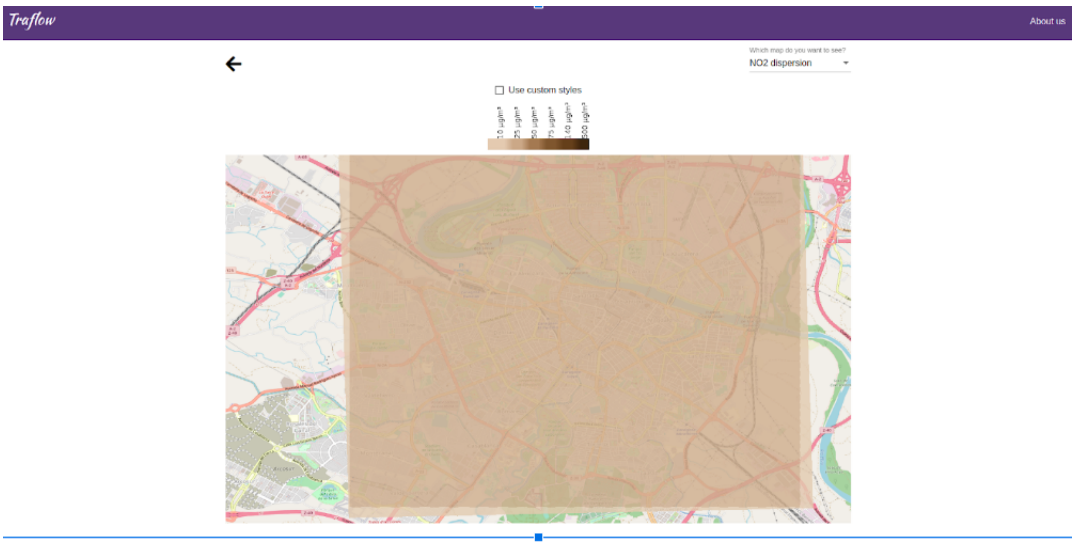

Figura 6.4: Mapa de dispersión mostrado en TraFlow.

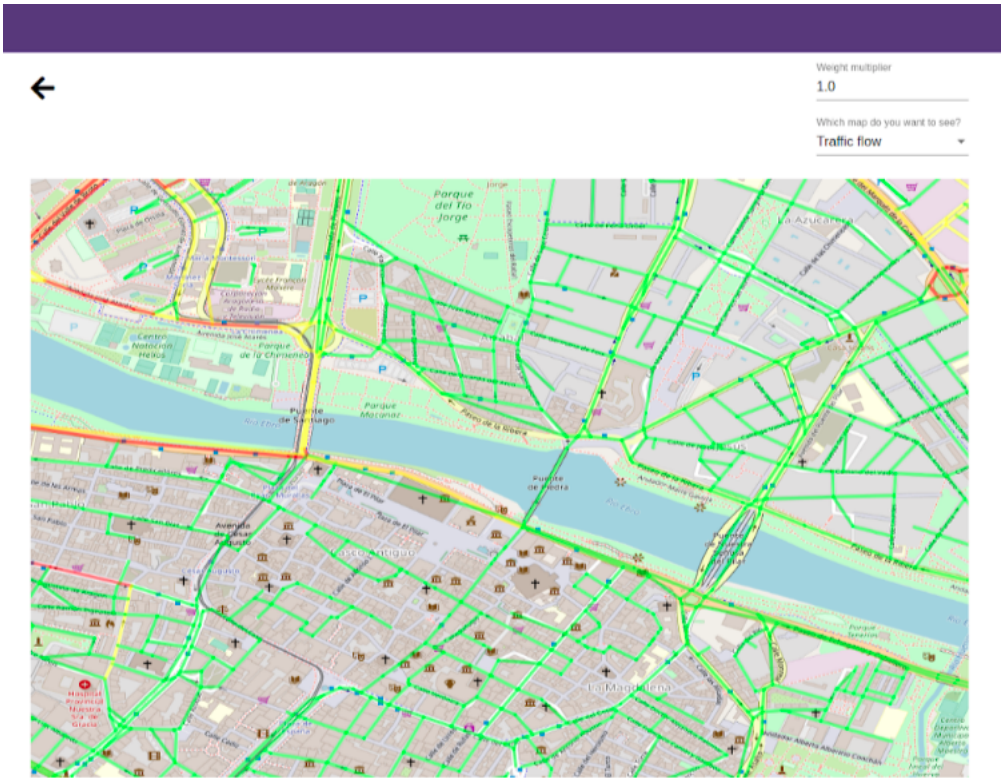

Figura 6.5: Mapa de tráfico mostrado en TraFlow.

Capítulo 7

Conclusiones

En este último capítulo se dará la opinión personal sobre el proyecto. Además, se expondrán los esfuerzos invertidos y las perspectivas futuras.

7.1. Evaluación personal

Trabajar dentro de un proyecto europeo como TRAFAIR, que es en el que se engloba este trabajo, ha sido una experiencia muy satisfactoria y de la que he aprendido mucho. En primer lugar, me gustaría comentar que la realización de este trabajo de fin de máster ha sido mucho más dura que cualquiera de los trabajos que realicé a lo largo del grado y máster, incluído el trabajo de fin de grado.

Además, para la realización de este proyecto ha sido necesario trabajar en un equipo multidisciplinar, que engloba desde expertos medioambientales hasta técnicos informáticos. Personalmente considero que ha sido una muy buena experiencia ya que me ha ayudado a entender y abordar los problemas desde puntos de vista distintos.

Como ya se ha comentado a lo largo del documento, este trabajo es continuación de mi trabajo de fin de grado. Por tanto, he podido emplear todo lo realizado en dicho trabajo y ver la utilidad real que tenía el sistema que se construyó. Además, se han cumplido varias de las propuestas de trabajo futuro que se propusieron, como la simulación de situaciones de tráfico hipotéticas.

En definitiva, valoro la realización de este trabajo como una experiencia muy positiva, en la que he adquirido conocimientos no estudiados ni en el grado ni en el máster y que me ha permitido trabajar dentro de un equipo multidisciplinar.

7.2. Metodología y esfuerzos invertidos

Para la realización del proyecto se ha seguido una metodología basada en el ciclo de Deming (Planificar-Hacer-Verificar-Actuar), para asegurar la calidad del producto

final.

Respecto a los esfuerzos invertidos, el proyecto comenzó el 1 de marzo de 2020, justo al finalizar el curso académico, y su duración ha sido de 1 año y 8 meses, con un parón de 5 meses por motivos laborales. En la figura 7.1 y en la tabla 7.1 se detalla el diagrama de Gantt y los esfuerzos invertidos.

Tarea	**Tiempo (h)**
Mejora del modelo de tráfico	130
Análisis de las simulaciones de tráfico	50
Calibración de medidas de contaminación	50
Mejora del modelo de dispersiones	50
Inclusión de otras fuentes	100
Evaluación del modelo de dispersiones	50
Generación de escenarios de tráfico	50
Mejoras en TraFlow	30
Documentación	50
Total	560

Tabla 7.1: Horas invertidas en el proyecto.

7.3. Trabajo futuro

Existen varias líneas de trabajo que se pueden abordar. En primer lugar, se propone mejorar la calibración de los sensores de bajo coste. Como se ha visto a lo largo de este trabajo, la calibración para determinados contaminantes no es del todo precisa.

Por otro lado, se propone mejorar la capacidad predictiva del modelo de dispersiones. Se ha comprobado que el modelo que se ha configurado empleando GRAL tiende a hacer predicciones por debajo de las reales. Por tanto, se propone realizar un análisis de las posibles causas de este problema, como la calidad de los datos de entrada o el propio modelo en sí, y de determinar cómo podría solucionarse.

Finalmente se propone la evaluación de más escenarios. Se ha hecho la evaluación de un escenario en el que se ha considerado que el porcentaje de vehículos eléctricos que circulan por la ciudad es mayor al actual. Por tanto, se propone evaluar otros escenarios como, por ejemplo, que se prohibiese el tráfico vial en determinadas zonas de la ciudad.

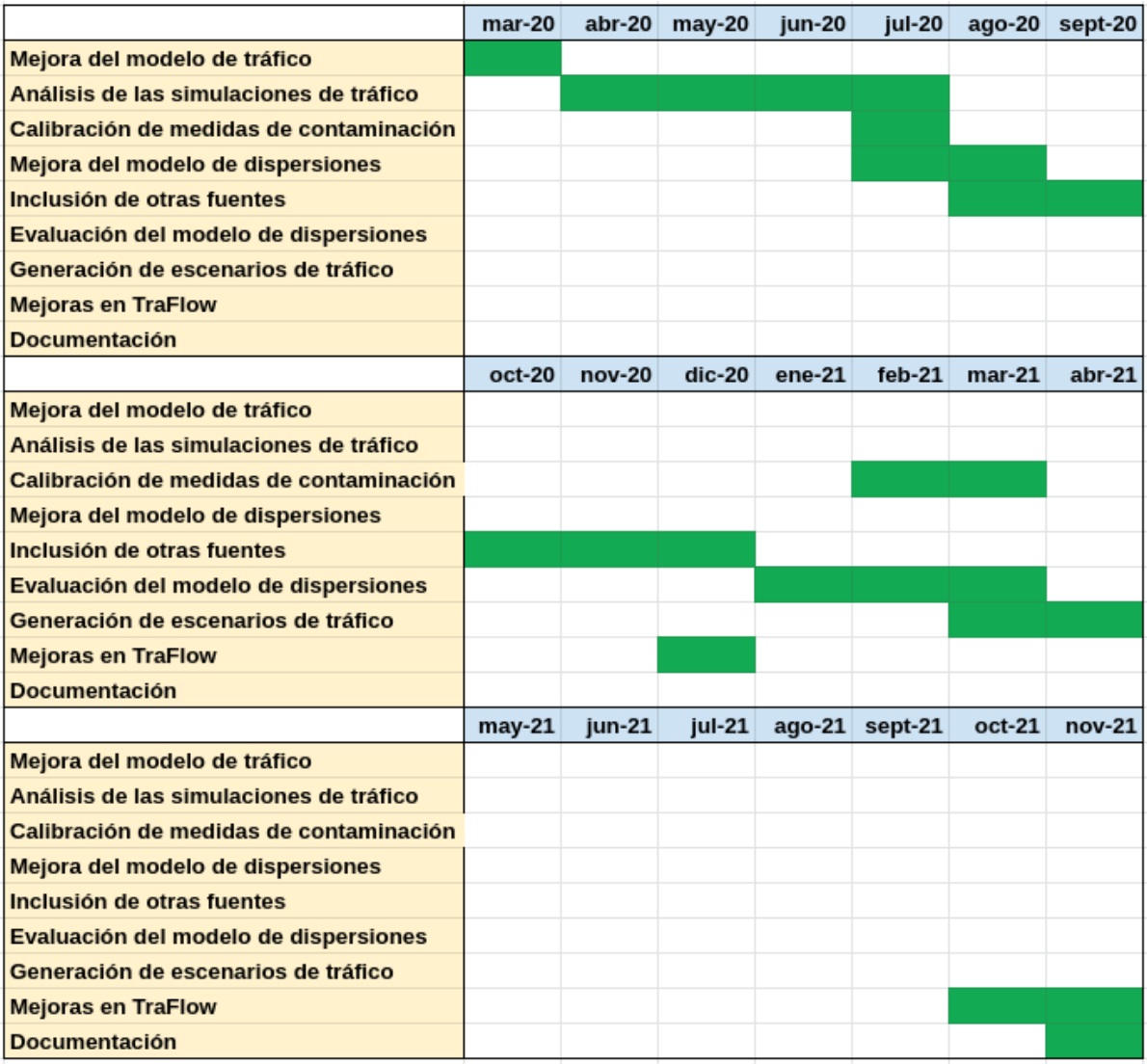

Figura 7.1: Diagrama de Gantt.

Bibliografía

[1] Karn Vohra, Alina Vodonos, Joel Schwartz, Eloise A. Marais, Melissa P. Sulprizio, and Loretta J. Mickley. Global mortality from outdoor fine particle pollution generated by fossil fuel combustion: Results from geos-chem. *Environmental Research*, 195:110754, 2021.

[2] D. Sáez. Desarrollo e implementación de herramientas de monitorización y simulación de tráfico basadas en datos abiertos. *Universidad de Zaragoza, EINA*, 2019.

[3] Mordechai (Muki) Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, October 2008.

[4] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, November 2018.

[5] S. Ibarra-Espinosa, R. Ynoue, S. O'Sullivan, E. Pebesma, M. D. F. Andrade, and M. Osses. Vein v0.2.2: an r package for bottom–up vehicular emissions inventories. *Geoscientific Model Development*, 11(6):2209–2229, 2018.

[6] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] J Ranstam and JA Cook. Lasso regression. *Journal of British Surgery*, 105(10):1348–1348, 2018.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Leonardo Noriega. Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, 2005.

[11] Greg Hamerly and Charles Elkan. Learning the k in k-means. *Advances in neural information processing systems*, 16:281–288, 2003.

[12] *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[13] Stan Salvador and Philip Ka-Fai Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. 2004.

[14] Alessandro Bigi, Michael Mueller, Stuart K Grange, Grazia Ghermandi, and Christoph Hueglin. Performance of no, no 2 low cost sensors and three calibration approaches within a real world application. *Atmospheric Measurement Techniques*, 11(6):3717–3735, 2018.

[15] David C. Carslaw and Karl Ropkins. openair — an r package for air quality data analysis. *Environmental Modelling and Software*, 27–28(0):52–61, 2012.

[16] Jakob Erdmann. Online-kalibrierung einer mikroskopischen verkehrssimulation. In *ViMOS 2012*, November 2012.

[17] Nils Gustaf Eissfeldt. *Vehicle-based modelling of traffic . Theory and application to environmental impact modelling*. PhD thesis, Universität zu Köln, 2004.

[18] Debashish Chowdhury, Ludger Santen, and Andreas Schadschneider. Statistical physics of vehicular traffic and some related systems. *Physics Reports*, 329, 07 2000.

[19] Dipa Soni and Ashwin Makwana. A survey on mqtt: A protocol of internet of things(iot). 04 2017.

[20] Norbert Blenn and Fernando Kuipers. Lorawan in the wild: Measurements from the things network, 2017.

[21] Clement Nedelcu. *Nginx HTTP Server*, volume 75. Packt Publishing, 2010.

[22] Roger A Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2017.

[23] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

Anexos

Anexo A

Métricas obtenidas en el entrenamiento del predictor de tráfico

En la tabla A.1 se muestran los resultados de entrenamiento para el predictor de tráfico.

Tabla A.1: Resultados de entrenamiento del predictor de tráfico.

RF (num_trees=700, num_leaves=8)	hour;weekday;type_day;month	142.1201	77.5722	0.9164	0.953
RF (num_trees=300, num_leaves=12)	hour;weekday;type_day;month	142.2588	77.8251	0.9163	0.9529
RF (num_trees=900, num_leaves=8)	hour;weekday;type_day;month	142.2283	77.3161	0.9162	0.953
RF (num_trees=700, num_leaves=12)	hour;weekday;type_day;month	142.4037	77.9013	0.9161	0.9531
RF (num_trees=900, num_leaves=12)	hour;weekday;type_day;month	142.7527	78.1634	0.9156	0.9527
RF (num_trees=900, num_leaves=16)	hour;weekday;type_day;month	143.6033	79.1645	0.9147	0.9522
RF (num_trees=100, num_leaves=16)	hour;weekday;type_day;month	143.6544	79.1189	0.9146	0.9523
RF (num_trees=500, num_leaves=16)	hour;weekday;type_day;month	143.5754	79.2039	0.9146	0.9523
RF (num_trees=300, num_leaves=16)	hour;weekday;type_day;month	143.7105	79.1671	0.9145	0.9523
RF (num_trees=700, num_leaves=16)	hour;weekday;type_day;month	144.0134	79.3308	0.9142	0.952
RF (num_trees=500, num_leaves=20)	hour;weekday;type_day;month	144.5843	80.2274	0.9135	0.9516

RF (num_trees=300, num_leaves=8)	hour;weekday;month	144.6272	78.402	0.9134	0.9512
RF (num_trees=500, num_leaves=8)	hour;weekday;month	144.7354	78.2136	0.9133	0.9512
RF (num_trees=900, num_leaves=12)	hour;weekday;month	144.6956	78.8961	0.9133	0.9514
RF (num_trees=700, num_leaves=20)	hour;weekday;type_day;month	144.7628	80.277	0.9133	0.9516
RF (num_trees=300, num_leaves=12)	hour;weekday;month	144.9116	78.8058	0.9131	0.9511
RF (num_trees=700, num_leaves=8)	hour;weekday;month	144.8702	78.5449	0.9131	0.9509
RF (num_trees=100, num_leaves=8)	hour;weekday;month	144.932	78.5637	0.913	0.9509
RF (num_trees=300, num_leaves=20)	hour;weekday;type_day;month	144.8958	80.3186	0.913	0.9514
RF (num_trees=900, num_leaves=20)	hour;weekday;type_day;month	144.927	80.3553	0.913	0.9514
RF (num_trees=500, num_leaves=12)	hour;weekday;month	145.2168	79.0524	0.9127	0.9508
RF (num_trees=700, num_leaves=12)	hour;weekday;month	145.2343	79.0709	0.9126	0.9506
RF (num_trees=100, num_leaves=20)	hour;weekday;type_day;month	145.2689	80.7321	0.9126	0.9512
RF (num_trees=900, num_leaves=8)	hour;weekday;month	145.3439	78.5413	0.9125	0.9506
RF (num_trees=100, num_leaves=12)	hour;weekday;month	145.5084	79.1064	0.9123	0.9506
RF (num_trees=900, num_leaves=16)	hour;weekday;month	145.9348	80.0461	0.9119	0.9506
RF (num_trees=300, num_leaves=16)	hour;weekday;month	146.0032	80.0214	0.9118	0.9506
RF (num_trees=500, num_leaves=16)	hour;weekday;month	146.1118	79.93	0.9117	0.9505
RF (num_trees=100, num_leaves=16)	hour;weekday;month	146.3764	80.1995	0.9113	0.9501
RF (num_trees=700, num_leaves=16)	hour;weekday;month	146.3757	80.2459	0.9113	0.9501
RF (num_trees=300, num_leaves=20)	hour;weekday;month	147.4327	81.3042	0.9101	0.9497
RF (num_trees=500, num_leaves=20)	hour;weekday;month	147.3798	81.256	0.9101	0.9497
RF (num_trees=100, num_leaves=20)	hour;weekday;month	147.6943	81.3787	0.9098	0.9495
RF (num_trees=700, num_leaves=20)	hour;weekday;month	147.5497	81.2738	0.9098	0.9496
RF (num_trees=900, num_leaves=20)	hour;weekday;month	147.707	81.3729	0.9097	0.9491
RF (num_trees=900, num_leaves=20)	hour;weekday;trimester	153.9934	86.133	0.9018	0.944
RF (num_trees=300, num_leaves=20)	hour;weekday;trimester	154.0715	86.1804	0.9017	0.9438
RF (num_trees=500, num_leaves=20)	hour;weekday;trimester	154.2518	86.2614	0.9016	0.9436
RF (num_trees=100, num_leaves=20)	hour;weekday;trimester	154.2623	86.3326	0.9015	0.9435
RF (num_trees=900, num_leaves=16)	hour;weekday;trimester	154.2489	86.0943	0.9015	0.9435
RF (num_trees=100, num_leaves=16)	hour;weekday;trimester	154.3305	86.0915	0.9013	0.9435

RF (num_trees=500, num_leaves=16)	hour;weekday;trimester	154.6139	86.3274	0.9011	0.9434
RF (num_trees=700, num_leaves=20)	hour;weekday;trimester	154.5549	86.3733	0.9011	0.9434
RF (num_trees=700, num_leaves=16)	hour;weekday;trimester	154.6591	86.3672	0.901	0.9433
RF (num_trees=300, num_leaves=16)	hour;weekday;trimester	154.8006	86.4999	0.9008	0.9431
RF (num_trees=700, num_leaves=12)	hour;weekday;trimester	154.824	86.4469	0.9008	0.9428
RF (num_trees=100, num_leaves=12)	hour;weekday;trimester	155.1736	86.494	0.9003	0.9426
RF (num_trees=300, num_leaves=12)	hour;weekday;trimester	155.2375	86.6347	0.9002	0.9424
RF (num_trees=500, num_leaves=12)	hour;weekday;trimester	155.35	86.7304	0.9001	0.9425
RF (num_trees=900, num_leaves=12)	hour;weekday;trimester	155.3295	86.521	0.9001	0.9425
RF (num_trees=100, num_leaves=8)	hour;weekday;trimester	155.8979	86.9818	0.8994	0.9418
RF (num_trees=500, num_leaves=8)	hour;weekday;trimester	156.0251	86.978	0.8992	0.9416
RF (num_trees=300, num_leaves=8)	hour;weekday;trimester	156.4012	87.29	0.8987	0.9414
RF (num_trees=700, num_leaves=8)	hour;weekday;trimester	156.4234	87.4444	0.8987	0.941
RF (num_trees=900, num_leaves=8)	hour;weekday;trimester	156.4182	87.2331	0.8987	0.9413
NN (num_layers=11, num_perceptrons=12)	hour;weekday;type_day;month	163.3254	103.3227	0.8896	0.9237
RF (num_trees=500, num_leaves=16)	hour;type_day;month	166.1542	99.3937	0.8857	0.9352
RF (num_trees=100, num_leaves=16)	hour;type_day;month	166.2377	99.4646	0.8856	0.9348
RF (num_trees=700, num_leaves=16)	hour;type_day;month	166.2485	99.3878	0.8856	0.9352
RF (num_trees=900, num_leaves=12)	hour;type_day;month	166.2344	99.4633	0.8856	0.935
RF (num_trees=900, num_leaves=16)	hour;type_day;month	166.2492	99.4899	0.8856	0.9352
RF (num_trees=300, num_leaves=12)	hour;type_day;month	166.2719	99.4429	0.8855	0.935
RF (num_trees=700, num_leaves=12)	hour;type_day;month	166.307	99.4729	0.8855	0.9349
RF (num_trees=700, num_leaves=20)	hour;type_day;month	166.3402	99.3417	0.8855	0.935
RF (num_trees=300, num_leaves=16)	hour;type_day;month	166.3798	99.4115	0.8854	0.935
RF (num_trees=500, num_leaves=12)	hour;type_day;month	166.4045	99.4047	0.8854	0.9347
RF (num_trees=500, num_leaves=20)	hour;type_day;month	166.3678	99.4433	0.8854	0.9351
RF (num_trees=900, num_leaves=20)	hour;type_day;month	166.4785	99.4464	0.8853	0.935
RF (num_trees=100, num_leaves=12)	hour;type_day;month	166.6232	99.5807	0.885	0.9347
NN (num_layers=5, num_perceptrons=10)	hour;weekday;type_day;month	166.8224	109.4137	0.8849	0.9177
RF (num_trees=100, num_leaves=8)	hour;type_day;month	166.7358	99.5404	0.8849	0.9342

RF (num_trees=300, num_leaves=20)	hour;type_day;month	166.8189	99.4382	0.8848	0.9347
RF (num_trees=100, num_leaves=20)	hour;type_day;month	166.9123	99.5503	0.8847	0.9344
RF (num_trees=300, num_leaves=8)	hour;type_day;month	166.9017	99.7432	0.8847	0.9341
RF (num_trees=900, num_leaves=8)	hour;type_day;month	166.9052	99.846	0.8846	0.934
RF (num_trees=700, num_leaves=8)	hour;type_day;month	167.0092	99.8616	0.8845	0.9339
RF (num_trees=500, num_leaves=8)	hour;type_day;month	167.2019	99.9804	0.8843	0.934
RF (num_trees=700, num_leaves=12)	hour;weekday	167.2664	99.3467	0.8843	0.926
RF (num_trees=900, num_leaves=8)	hour;weekday	167.187	99.3941	0.8843	0.926
RF (num_trees=900, num_leaves=20)	hour;weekday	167.2214	99.4169	0.8843	0.9259
RF (num_trees=100, num_leaves=20)	hour;weekday	167.2535	99.4477	0.8842	0.9259
RF (num_trees=500, num_leaves=20)	hour;weekday	167.2873	99.4314	0.8842	0.9258
RF (num_trees=700, num_leaves=16)	hour;weekday	167.2585	99.4051	0.8842	0.9259
RF (num_trees=900, num_leaves=16)	hour;weekday	167.2491	99.3968	0.8842	0.9258
RF (num_trees=100, num_leaves=12)	hour;weekday	167.3661	99.5244	0.884	0.926
RF (num_trees=100, num_leaves=16)	hour;weekday	167.4097	99.5596	0.884	0.9259
RF (num_trees=300, num_leaves=12)	hour;weekday	167.3736	99.5911	0.884	0.9259
RF (num_trees=300, num_leaves=20)	hour;weekday	167.3969	99.4902	0.884	0.9258
RF (num_trees=500, num_leaves=16)	hour;weekday	167.3795	99.4739	0.884	0.9258
RF (num_trees=700, num_leaves=8)	hour;weekday	167.3966	99.4992	0.884	0.9259
RF (num_trees=500, num_leaves=12)	hour;weekday	167.4683	99.5194	0.8839	0.9255
RF (num_trees=700, num_leaves=20)	hour;weekday	167.5109	99.5832	0.8839	0.9256
RF (num_trees=300, num_leaves=8)	hour;weekday	167.5316	99.697	0.8838	0.9258
RF (num_trees=100, num_leaves=8)	hour;weekday	167.5682	99.6363	0.8837	0.9255
RF (num_trees=500, num_leaves=8)	hour;weekday	167.6499	99.6658	0.8837	0.9254
RF (num_trees=900, num_leaves=12)	hour;weekday	167.5532	99.6085	0.8837	0.9257
RF (num_trees=300, num_leaves=16)	hour;weekday	167.6724	99.6014	0.8836	0.9256
NN (num_layers=8, num_perceptrons=10)	hour;weekday;month	168.6359	108.6365	0.8823	0.9164
RF (num_trees=500, num_leaves=20)	hour;type_day;trimester	172.8454	104.7902	0.8764	0.9279
RF (num_trees=300, num_leaves=16)	hour;type_day;trimester	172.8539	104.7679	0.8763	0.928
RF (num_trees=300, num_leaves=20)	hour;type_day;trimester	172.8834	104.8883	0.8763	0.9281

RF (num_trees=300, num_leaves=12)	hour;type_day;trimester	172.9571	104.875	0.8762	0.9277
RF (num_trees=900, num_leaves=16)	hour;type_day;trimester	172.9216	104.8521	0.8762	0.9279
RF (num_trees=100, num_leaves=12)	hour;type_day;trimester	172.9959	104.9181	0.8761	0.928
RF (num_trees=100, num_leaves=20)	hour;type_day;trimester	173.0557	105.0519	0.876	0.9279
RF (num_trees=300, num_leaves=8)	hour;type_day;trimester	173.0782	104.9569	0.876	0.9278
RF (num_trees=700, num_leaves=12)	hour;type_day;trimester	173.1023	105.031	0.876	0.928
RF (num_trees=900, num_leaves=12)	hour;type_day;trimester	173.031	104.9949	0.876	0.9277
RF (num_trees=500, num_leaves=8)	hour;type_day;trimester	173.1135	105.0189	0.8759	0.9276
RF (num_trees=500, num_leaves=12)	hour;type_day;trimester	173.1282	105.0098	0.8759	0.9279
RF (num_trees=500, num_leaves=16)	hour;type_day;trimester	173.1129	105.0444	0.8759	0.9278
RF (num_trees=700, num_leaves=20)	hour;type_day;trimester	173.1487	104.9446	0.8759	0.9276
RF (num_trees=900, num_leaves=20)	hour;type_day;trimester	173.1985	105.0386	0.8759	0.9277
RF (num_trees=700, num_leaves=8)	hour;type_day;trimester	173.2162	105.1008	0.8758	0.9276
RF (num_trees=100, num_leaves=16)	hour;type_day;trimester	173.3172	105.0887	0.8757	0.9274
RF (num_trees=700, num_leaves=16)	hour;type_day;trimester	173.3142	105.1343	0.8756	0.9276
RF (num_trees=900, num_leaves=8)	hour;type_day;trimester	173.3873	105.209	0.8755	0.9274
RF (num_trees=100, num_leaves=8)	hour;type_day;trimester	173.7204	105.3761	0.8751	0.9272
NN (num_layers=8, num_perceptrons=10)	hour;weekday;trimester	177.8312	114.4003	0.8691	0.9059
RF (num_trees=100, num_leaves=16)	hour;type_day	184.9105	116.3528	0.8585	0.9097
RF (num_trees=100, num_leaves=12)	hour;type_day	184.943	116.4081	0.8584	0.9098
RF (num_trees=300, num_leaves=16)	hour;type_day	184.9431	116.347	0.8584	0.91
RF (num_trees=500, num_leaves=8)	hour;type_day	184.9842	116.4531	0.8584	0.9101
RF (num_trees=700, num_leaves=12)	hour;type_day	184.9467	116.3891	0.8584	0.9101
RF (num_trees=900, num_leaves=8)	hour;type_day	184.969	116.4505	0.8584	0.9095
RF (num_trees=300, num_leaves=8)	hour;type_day	184.9885	116.4392	0.8583	0.9098
RF (num_trees=300, num_leaves=12)	hour;type_day	184.9911	116.4079	0.8583	0.9101
RF (num_trees=300, num_leaves=20)	hour;type_day	185.0098	116.4203	0.8583	0.9098
RF (num_trees=700, num_leaves=8)	hour;type_day	185.0373	116.3955	0.8583	0.9098
RF (num_trees=700, num_leaves=20)	hour;type_day	185.0089	116.4168	0.8583	0.9101
RF (num_trees=900, num_leaves=12)	hour;type_day	184.9995	116.4141	0.8583	0.9101

RF (num_trees=900, num_leaves=16)	hour;type_day	185.1196	116.5014	0.8582	0.9096
RF (num_trees=900, num_leaves=20)	hour;type_day	185.0807	116.4461	0.8582	0.9095
RF (num_trees=100, num_leaves=8)	hour;type_day	185.1492	116.4445	0.8581	0.91
RF (num_trees=100, num_leaves=20)	hour;type_day	185.155	116.5034	0.8581	0.9097
RF (num_trees=500, num_leaves=12)	hour;type_day	185.1247	116.4186	0.8581	0.9099
RF (num_trees=500, num_leaves=16)	hour;type_day	185.1346	116.4047	0.8581	0.9102
RF (num_trees=700, num_leaves=16)	hour;type_day	185.1137	116.439	0.8581	0.9101
RF (num_trees=500, num_leaves=20)	hour;type_day	185.103	116.476	0.858	0.91
NN (num_layers=11, num_perceptrons=2)	hour;weekday;month	192.337	127.5247	0.8471	0.8954
NN (num_layers=11, num_perceptrons=4)	hour;weekday;trimester	195.6659	129.7832	0.8418	0.8917
NN (num_layers=11, num_perceptrons=6)	hour;weekday;trimester	196.9694	129.9447	0.8392	0.8899
LA (alpha=0)	hour;weekday;type_day;month	201.7062	147.9426	0.8316	0.9029
LA (alpha=0)	hour;weekday;month	203.468	148.4364	0.8287	0.9005
LA (alpha=0)	hour;weekday;trimester	206.749	150.3298	0.8231	0.8956
NN (num_layers=11, num_perceptrons=10)	hour;type_day;trimester	211.3334	141.3822	0.8151	0.88
LA (alpha=0)	hour;weekday	212.1196	155.4371	0.8138	0.884
LA (alpha=0)	hour;type_day;month	212.591	157.1789	0.8129	0.8869
LA (alpha=0)	hour;type_day;trimester	215.7034	158.5946	0.8074	0.8822
NN (num_layers=11, num_perceptrons=2)	hour;weekday;type_day;month	219.553	149.8258	0.8001	0.8748
NN (num_layers=11, num_perceptrons=8)	hour;weekday;trimester	219.9674	151.6118	0.7992	0.869
LA (alpha=0)	hour;type_day	221.0911	163.6829	0.7976	0.8705
NN (num_layers=8, num_perceptrons=2)	hour;weekday;month	234.8034	162.0849	0.7716	0.857
NN (num_layers=5, num_perceptrons=12)	hour;weekday;month	235.6822	153.5711	0.7705	0.8525
NN (num_layers=11, num_perceptrons=10)	hour;weekday;month	236.4381	153.5607	0.7704	0.8544
NN (num_layers=5, num_perceptrons=8)	hour;weekday;type_day;month	236.2936	155.9491	0.7701	0.8525
NN (num_layers=11, num_perceptrons=4)	hour;type_day;month	237.7759	167.1746	0.7653	0.8544
NN (num_layers=11, num_perceptrons=12)	hour;weekday;trimester	241.1163	159.1965	0.7608	0.8477
RF (num_trees=300, num_leaves=12)	hour	245.78	168.414	0.75	0.8337
RF (num_trees=100, num_leaves=8)	hour	245.8351	168.464	0.7499	0.8335
RF (num_trees=100, num_leaves=16)	hour	245.8457	168.45	0.7498	0.8336

RF (num_trees=700, num_leaves=8)	hour	245.8399	168.4786	0.7498	0.8338
RF (num_trees=500, num_leaves=12)	hour	245.9328	168.5509	0.7497	0.8337
RF (num_trees=500, num_leaves=16)	hour	245.9085	168.484	0.7497	0.8332
RF (num_trees=500, num_leaves=20)	hour	245.9553	168.4942	0.7497	0.8335
RF (num_trees=700, num_leaves=12)	hour	245.8834	168.4543	0.7497	0.8337
RF (num_trees=900, num_leaves=8)	hour	245.922	168.4699	0.7497	0.8333
RF (num_trees=900, num_leaves=12)	hour	245.8871	168.4583	0.7497	0.8332
RF (num_trees=900, num_leaves=20)	hour	245.9079	168.5129	0.7497	0.8336
RF (num_trees=700, num_leaves=16)	hour	245.9527	168.5181	0.7496	0.8332
RF (num_trees=700, num_leaves=20)	hour	245.9515	168.4645	0.7496	0.8328
RF (num_trees=900, num_leaves=16)	hour	245.8946	168.5369	0.7496	0.8331
RF (num_trees=100, num_leaves=12)	hour	246.0073	168.514	0.7495	0.8331
RF (num_trees=300, num_leaves=20)	hour	246.0151	168.5568	0.7495	0.8332
RF (num_trees=500, num_leaves=8)	hour	246.013	168.5514	0.7495	0.8336
RF (num_trees=100, num_leaves=20)	hour	246.0302	168.5368	0.7494	0.8333
RF (num_trees=300, num_leaves=16)	hour	246.0499	168.5443	0.7494	0.8329
LA (alpha=0)	hour	246.1117	168.6106	0.7493	0.8337
RF (num_trees=300, num_leaves=8)	hour	246.1473	168.63	0.7492	0.8329
NN (num_layers=8, num_perceptrons=10)	hour;weekday;type_day;month	247.7828	166.367	0.7448	0.8383
NN (num_layers=8, num_perceptrons=4)	hour;weekday;month	250.0375	168.7826	0.7408	0.8361
NN (num_layers=11, num_perceptrons=4)	hour;weekday;month	252.8501	170.0511	0.7336	0.831
NN (num_layers=5, num_perceptrons=6)	hour;weekday;trimester	254.7281	173.6321	0.733	0.8287
NN (num_layers=11, num_perceptrons=12)	hour;type_day;month	258.8729	177.4067	0.7215	0.8258
NN (num_layers=11, num_perceptrons=2)	hour;type_day;month	265.7553	196.8482	0.7079	0.8206
NN (num_layers=8, num_perceptrons=4)	hour;weekday;trimester	267.5428	186.0518	0.705	0.8155
NN (num_layers=11, num_perceptrons=8)	hour;type_day	269.9025	191.3417	0.6977	0.8031
NN (num_layers=5, num_perceptrons=6)	hour;weekday;type_day;month	272.1427	190.992	0.6949	0.8165
NN (num_layers=5, num_perceptrons=10)	hour;type_day;month	275.4767	196.7941	0.6846	0.8037
NN (num_layers=2, num_perceptrons=12)	hour;weekday;month	275.9144	198.4512	0.6842	0.7954
NN (num_layers=11, num_perceptrons=2)	hour;weekday	279.6071	200.3418	0.6781	0.7908

NN (num_layers=8, num_perceptrons=8)	hour;type_day;trimester	279.4586	199.129	0.6771	0.7979
NN (num_layers=2, num_perceptrons=10)	hour;weekday;month	280.4143	199.8526	0.6753	0.7986
NN (num_layers=11, num_perceptrons=2)	hour;type_day;trimester	280.499	203.0677	0.6743	0.7961
NN (num_layers=2, num_perceptrons=10)	hour;weekday;type_day;month	280.8635	200.8145	0.6735	0.7833
NN (num_layers=8, num_perceptrons=12)	hour;weekday;month	283.8416	192.7363	0.668	0.8004
NN (num_layers=5, num_perceptrons=2)	hour;type_day;month	283.338	205.8371	0.6675	0.7901
NN (num_layers=8, num_perceptrons=8)	hour;weekday;type_day;month	288.5277	202.2089	0.6559	0.7887
NN (num_layers=11, num_perceptrons=10)	hour;weekday;trimester	289.4913	200.1813	0.6531	0.7846
NN (num_layers=8, num_perceptrons=12)	hour;weekday;trimester	292.3682	204.9102	0.6443	0.7766
NN (num_layers=11, num_perceptrons=10)	hour;weekday;type_day;month	295.3111	207.0034	0.6373	0.7817
NN (num_layers=8, num_perceptrons=12)	hour;type_day;month	295.9398	210.8976	0.6356	0.775
NN (num_layers=11, num_perceptrons=6)	hour;weekday;type_day;month	296.5587	209.2728	0.6352	0.7771
NN (num_layers=5, num_perceptrons=12)	hour;weekday;type_day;month	297.5802	211.5867	0.6345	0.7777
NN (num_layers=11, num_perceptrons=6)	hour;weekday;month	299.2802	213.4636	0.6297	0.7727
NN (num_layers=8, num_perceptrons=12)	hour	299.3353	220.4477	0.629	0.7482
NN (num_layers=11, num_perceptrons=12)	hour;weekday	300.0432	211.8785	0.6278	0.7567
NN (num_layers=5, num_perceptrons=2)	hour;type_day	301.0228	226.0858	0.6242	0.767
NN (num_layers=8, num_perceptrons=6)	hour;weekday;month	301.8356	217.5457	0.6226	0.7674
NN (num_layers=5, num_perceptrons=12)	hour;weekday;trimester	302.3835	217.2208	0.6196	0.7649
NN (num_layers=8, num_perceptrons=10)	hour;weekday	305.7052	221.7869	0.6139	0.752
NN (num_layers=11, num_perceptrons=4)	hour;weekday	306.1057	223.6562	0.6138	0.7513
NN (num_layers=8, num_perceptrons=6)	hour;type_day;month	306.7125	223.7305	0.6113	0.763
NN (num_layers=5, num_perceptrons=12)	hour;type_day;month	307.1403	222.9301	0.608	0.7592
NN (num_layers=5, num_perceptrons=4)	hour;type_day;month	309.2147	227.1294	0.6061	0.7579
NN (num_layers=8, num_perceptrons=4)	hour;type_day;month	307.8968	224.0841	0.606	0.7572
NN (num_layers=11, num_perceptrons=4)	hour;weekday;type_day;month	307.947	223.475	0.6049	0.7588
NN (num_layers=5, num_perceptrons=10)	hour;weekday;trimester	308.0895	227.09	0.6046	0.7552
NN (num_layers=11, num_perceptrons=10)	hour;weekday	308.871	225.2743	0.603	0.745
NN (num_layers=11, num_perceptrons=8)	hour;type_day;month	310.1306	225.2953	0.602	0.7553
NN (num_layers=5, num_perceptrons=4)	hour;weekday;month	309.8882	229.815	0.6017	0.76

NN (num_layers=11, num_perceptrons=10)	hour;type_day	315.7995	230.9535	0.5892	0.738
NN (num_layers=11, num_perceptrons=6)	hour;weekday	319.5307	237.1555	0.5772	0.732
NN (num_layers=5, num_perceptrons=8)	hour;weekday;month	322.3085	240.4884	0.5694	0.7392
NN (num_layers=5, num_perceptrons=2)	hour;weekday;month	322.8589	242.5324	0.5687	0.7374
NN (num_layers=8, num_perceptrons=4)	hour;type_day;trimester	324.446	245.4653	0.5632	0.7272
NN (num_layers=2, num_perceptrons=10)	hour;type_day;trimester	325.9751	247.0819	0.56	0.7255
NN (num_layers=2, num_perceptrons=12)	hour;type_day;trimester	329.9644	247.1187	0.5497	0.7109
NN (num_layers=5, num_perceptrons=4)	hour;weekday;type_day;month	332.2346	245.9707	0.5468	0.7232
NN (num_layers=11, num_perceptrons=8)	hour;weekday;month	331.2948	244.8653	0.5435	0.7191
NN (num_layers=11, num_perceptrons=6)	hour;type_day;trimester	331.7821	250.5798	0.543	0.7207
NN (num_layers=8, num_perceptrons=4)	hour;weekday;type_day;month	334.1852	251.2329	0.5379	0.7195
NN (num_layers=8, num_perceptrons=8)	hour;weekday;trimester	334.6117	248.8165	0.5377	0.7167
NN (num_layers=5, num_perceptrons=6)	hour;weekday;month	334.4834	248.5077	0.5373	0.7173
NN (num_layers=2, num_perceptrons=8)	hour;weekday;type_day;month	335.1914	253.2494	0.5357	0.7152
NN (num_layers=2, num_perceptrons=6)	hour;type_day;trimester	335.9886	260.2245	0.5328	0.7105
NN (num_layers=11, num_perceptrons=10)	hour;type_day;month	337.3384	252.2649	0.5279	0.7127
NN (num_layers=8, num_perceptrons=12)	hour;weekday	338.0602	253.7624	0.5248	0.6962
NN (num_layers=11, num_perceptrons=12)	hour;type_day;trimester	338.9052	254.5087	0.5246	0.7108
NN (num_layers=2, num_perceptrons=10)	hour	339.9699	260.3196	0.5242	
NN (num_layers=5, num_perceptrons=8)	hour;type_day;trimester	339.5416	255.296	0.5231	0.7084
NN (num_layers=8, num_perceptrons=12)	hour;weekday;type_day;month	339.3076	256.5772	0.5228	0.7139
NN (num_layers=8, num_perceptrons=6)	hour;weekday;trimester	340.44	260.4624	0.5215	0.7097
NN (num_layers=5, num_perceptrons=12)	hour	340.232	265.9456	0.5206	0.6842
NN (num_layers=2, num_perceptrons=12)	hour;weekday	340.6864	258.0789	0.5195	0.6946
NN (num_layers=5, num_perceptrons=10)	hour;weekday	340.7754	254.0875	0.5188	0.6938
NN (num_layers=5, num_perceptrons=10)	hour;weekday;month	341.508	258.5913	0.5185	0.7087
NN (num_layers=11, num_perceptrons=6)	hour;type_day;month	340.4477	255.3439	0.5182	0.7077
NN (num_layers=8, num_perceptrons=12)	hour;type_day;trimester	341.7404	256.4229	0.516	0.6996
NN (num_layers=5, num_perceptrons=10)	hour;type_day;trimester	341.7022	256.8396	0.515	0.7002
NN (num_layers=2, num_perceptrons=10)	hour;type_day	343.5327	262.4157	0.5129	0.682

NN (num_layers=8, num_perceptrons=12)	hour;type_day	343.7143	259.1379	0.5114	0.6918
NN (num_layers=8, num_perceptrons=4)	hour;type_day	343.8586	269.7817	0.5113	0.6944
NN (num_layers=5, num_perceptrons=8)	hour;weekday;trimester	344.6269	262.1557	0.5092	0.7003
NN (num_layers=2, num_perceptrons=10)	hour;weekday;trimester	344.6805	264.6013	0.5079	0.6967
NN (num_layers=2, num_perceptrons=12)	hour;weekday;trimester	345.4999	265.8923	0.5066	0.6982
NN (num_layers=5, num_perceptrons=4)	hour;weekday;trimester	348.3539	269.3104	0.4975	0.6905
NN (num_layers=8, num_perceptrons=6)	hour;type_day;trimester	349.031	266.9292	0.4956	0.6905
NN (num_layers=5, num_perceptrons=4)	hour;type_day	349.2806	270.5048	0.4953	0.6841
NN (num_layers=5, num_perceptrons=4)	hour;weekday	349.2248	267.0577	0.4953	0.6836
NN (num_layers=5, num_perceptrons=2)	hour;weekday;type_day;month	350.0083	271.44	0.4942	0.6977
NN (num_layers=8, num_perceptrons=6)	hour;weekday;type_day;month	349.8746	270.2616	0.4935	0.6969
NN (num_layers=8, num_perceptrons=10)	hour;type_day;trimester	350.3309	268.2245	0.4931	0.6904
NN (num_layers=11, num_perceptrons=8)	hour;weekday;type_day;month	350.1275	270.0409	0.493	0.6959
NN (num_layers=11, num_perceptrons=8)	hour;type_day;trimester	350.2844	269.0091	0.4925	0.6914
NN (num_layers=11, num_perceptrons=4)	hour;type_day;trimester	351.8468	273.0148	0.4894	0.6875
NN (num_layers=5, num_perceptrons=12)	hour;type_day;trimester	351.4617	271.5253	0.4877	0.6861
NN (num_layers=8, num_perceptrons=8)	hour;type_day	352.6495	272.9276	0.4864	0.6764
NN (num_layers=2, num_perceptrons=8)	hour;type_day;trimester	352.6126	272.5729	0.4859	0.6824
NN (num_layers=2, num_perceptrons=4)	hour;type_day;month	352.935	274.5968	0.4843	0.677
NN (num_layers=11, num_perceptrons=12)	hour;type_day	354.3149	273.2764	0.4837	0.6798
NN (num_layers=2, num_perceptrons=6)	hour;weekday;month	354.9987	275.7616	0.4783	0.6916
NN (num_layers=5, num_perceptrons=6)	hour;type_day	355.2305	274.557	0.4782	0.6746
NN (num_layers=2, num_perceptrons=8)	hour;weekday;month	355.4341	282.9051	0.4767	0.6865
NN (num_layers=11, num_perceptrons=2)	hour;type_day	355.7934	273.9261	0.4766	0.6733
NN (num_layers=8, num_perceptrons=8)	hour;weekday	356.9057	276.1724	0.4737	0.6753
NN (num_layers=11, num_perceptrons=6)	hour;type_day	357.8635	280.3059	0.4688	0.671
NN (num_layers=2, num_perceptrons=6)	hour;weekday;trimester	358.8428	284.4329	0.4667	0.6778
NN (num_layers=8, num_perceptrons=2)	hour;type_day	359.6385	283.1786	0.4648	0.6715
NN (num_layers=11, num_perceptrons=4)	hour;type_day	362.9083	286.2033	0.4538	0.6599
NN (num_layers=11, num_perceptrons=12)	hour	367.357	290.0207	0.4404	0.6309

NN (num_layers=8, num_perceptrons=8)	hour;weekday;month	369.3632	290.1622	0.4352	0.6595
NN (num_layers=11, num_perceptrons=12)	hour;weekday;month	370.6371	288.9432	0.4325	0.6601
NN (num_layers=8, num_perceptrons=6)	hour	371.1605	297.9326	0.4269	0.6208
NN (num_layers=2, num_perceptrons=8)	hour;weekday;trimester	373.4393	296.8939	0.4231	0.6493
NN (num_layers=5, num_perceptrons=12)	hour;weekday	375.0557	297.1738	0.4208	0.6409
NN (num_layers=11, num_perceptrons=8)	hour;weekday	374.0282	293.1159	0.4197	0.6415
NN (num_layers=8, num_perceptrons=8)	hour;type_day;month	375.7958	297.9977	0.4166	0.6472
NN (num_layers=5, num_perceptrons=6)	hour;type_day;month	375.9035	297.935	0.4161	0.6466
NN (num_layers=11, num_perceptrons=8)	hour	376.0918	301.9546	0.4153	0.61
NN (num_layers=11, num_perceptrons=4)	hour	376.5282	303.0224	0.414	0.6153
NN (num_layers=2, num_perceptrons=4)	hour	376.0255	302.3821	0.4139	0.6156
NN (num_layers=8, num_perceptrons=10)	hour;type_day;month	375.9268	297.1194	0.4131	0.6443
NN (num_layers=2, num_perceptrons=4)	hour;weekday	376.6732	298.8388	0.4129	0.6334
NN (num_layers=2, num_perceptrons=6)	hour;weekday;type_day;month	377.5771	306.6343	0.4098	0.6492
NN (num_layers=8, num_perceptrons=8)	hour	377.8447	307.6586	0.4095	0.6127
NN (num_layers=2, num_perceptrons=12)	hour;weekday;type_day;month	377.4745	303.2875	0.4094	0.6482
NN (num_layers=8, num_perceptrons=4)	hour	378.4234	308.7665	0.4067	0.6108
NN (num_layers=5, num_perceptrons=6)	hour;type_day;trimester	379.6244	301.0011	0.4021	0.6365
NN (num_layers=2, num_perceptrons=6)	hour	380.9272	309.8924	0.3982	0.6025
NN (num_layers=5, num_perceptrons=12)	hour;type_day	382.1988	301.9746	0.3949	0.623
NN (num_layers=8, num_perceptrons=10)	hour;type_day	382.3859	307.8518	0.3943	0.6284
NN (num_layers=8, num_perceptrons=6)	hour;weekday	383.2193	308.5002	0.3927	0.6263
NN (num_layers=2, num_perceptrons=4)	hour;type_day;trimester	383.4129	309.0738	0.3923	
NN (num_layers=8, num_perceptrons=4)	hour;weekday	384.0264	307.7333	0.3921	0.628
NN (num_layers=2, num_perceptrons=10)	hour;type_day;month	384.1171	312.021	0.3883	0.6298
NN (num_layers=5, num_perceptrons=8)	hour;weekday	384.0571	307.7569	0.3883	0.6249
NN (num_layers=2, num_perceptrons=12)	hour;type_day;month	384.7587	310.1139	0.3882	0.6277
NN (num_layers=5, num_perceptrons=6)	hour;weekday	384.8218	311.5067	0.3868	0.6234
NN (num_layers=2, num_perceptrons=6)	hour;weekday	385.3434	315.3035	0.3856	0.6262
NN (num_layers=5, num_perceptrons=4)	hour;type_day;trimester	386.0868	310.9355	0.384	0.6254

NN (num_layers=2, num_perceptrons=2)	hour;type_day;trimester	387.2669	309.4176	0.3798	0.6549
NN (num_layers=5, num_perceptrons=10)	hour;type_day	389.4658	314.3224	0.3728	0.6098
NN (num_layers=2, num_perceptrons=2)	hour;weekday	389.6053	314.2816	0.3702	0.6148
NN (num_layers=8, num_perceptrons=6)	hour;type_day	390.8172	318.122	0.3683	0.6097
NN (num_layers=2, num_perceptrons=6)	hour;type_day	391.3463	317.3222	0.3659	0.6063
NN (num_layers=2, num_perceptrons=8)	hour;type_day	391.7456	315.9078	0.3645	0.5994
NN (num_layers=2, num_perceptrons=8)	hour;type_day;month	394.7359	319.9945	0.3532	
NN (num_layers=5, num_perceptrons=10)	hour	399.7304	328.9456	0.3396	0.5652
NN (num_layers=2, num_perceptrons=4)	hour;weekday;month	399.4289	326.7177	0.3367	
NN (num_layers=8, num_perceptrons=10)	hour	400.2348	329.7067	0.3362	0.564
NN (num_layers=11, num_perceptrons=2)	hour;weekday;trimester	403.7859	245.9801	0.3239	
NN (num_layers=5, num_perceptrons=4)	hour	405.4153	339.1047	0.319	0.5572
NN (num_layers=5, num_perceptrons=8)	hour	405.3973	339.5865	0.3187	0.5564
NN (num_layers=2, num_perceptrons=8)	hour	405.4809	339.1323	0.3186	0.5559
NN (num_layers=2, num_perceptrons=2)	hour	406.5008	340.4504	0.3178	0.5563
NN (num_layers=2, num_perceptrons=8)	hour;weekday	406.6488	333.2946	0.3148	
NN (num_layers=5, num_perceptrons=6)	hour	407.3434	343.7088	0.3145	0.5561
NN (num_layers=2, num_perceptrons=10)	hour;weekday	408.715	338.4793	0.3085	0.5779
NN (num_layers=5, num_perceptrons=8)	hour;type_day;month	409.1889	340.1278	0.307	0.5826
NN (num_layers=2, num_perceptrons=6)	hour;type_day;month	409.2109	340.4866	0.3068	0.582
NN (num_layers=5, num_perceptrons=8)	hour;type_day	413.4328	343.1885	0.2926	0.5639
NN (num_layers=2, num_perceptrons=2)	hour;type_day	413.4582	343.1553	0.2925	0.5637
NN (num_layers=2, num_perceptrons=4)	hour;type_day	413.4497	343.0748	0.2924	0.5638
NN (num_layers=2, num_perceptrons=12)	hour;type_day	413.5506	343.3819	0.2917	0.5639
NN (num_layers=2, num_perceptrons=12)	hour	418.8176	353.4209	0.2737	
LA (alpha=20)	hour;weekday;type_day;month	423.6275	380.6377	0.2573	0.7572
LA (alpha=20)	hour;weekday;trimester	425.5881	383.703	0.2504	0.8003
LA (alpha=20)	hour;weekday	425.7815	383.8287	0.2497	0.7997
LA (alpha=20)	hour;weekday;month	425.8104	383.8703	0.2496	0.8015
LA (alpha=20)	hour;type_day;month	426.5025	383.3701	0.247	0.7651

LA (alpha=20)	hour;type_day	426.5708	383.401	0.2469	0.7573
LA (alpha=20)	hour;type_day;trimester	426.5132	383.4071	0.2467	0.7645
NN (num_layers=8, num_perceptrons=2)	hour	427.2398	366.3197	0.2446	0.5096
NN (num_layers=11, num_perceptrons=6)	hour	427.2706	366.3072	0.2443	0.5091
NN (num_layers=11, num_perceptrons=10)	hour	427.3056	366.4588	0.2441	0.5088
NN (num_layers=8, num_perceptrons=2)	hour;weekday;trimester	430.3599	278.8259	0.2345	
NN (num_layers=8, num_perceptrons=2)	hour;weekday;type_day;month	436.0891	286.6131	0.2149	
LA (alpha=20)	hour	437.7559	397.7421	0.2068	0.7861
NN (num_layers=5, num_perceptrons=2)	hour;type_day;trimester	441.8936	301.7102	0.1856	
NN (num_layers=8, num_perceptrons=2)	hour;weekday	445.9154	306.4963	0.1729	
NN (num_layers=2, num_perceptrons=4)	hour;weekday;trimester	460.2006	325.7682	0.1244	
NN (num_layers=8, num_perceptrons=2)	hour;type_day;month	460.2524	328.0916	0.1183	0.5205
NN (num_layers=5, num_perceptrons=2)	hour;weekday	463.0309	329.236	0.1093	
NN (num_layers=5, num_perceptrons=2)	hour	485.7051	365.1156	0.0287	
LA (alpha=40)	hour;type_day	487.6669	439.0063	0.0158	0.1943
LA (alpha=40)	hour;type_day;trimester	487.67	439.0056	0.0158	0.1944
NN (num_layers=2, num_perceptrons=4)	hour;weekday;type_day;month	487.0803	363.7084	0.0156	
LA (alpha=40)	hour;type_day;month	487.6776	439.022	0.0156	0.1943
LA (alpha=40)	hour;weekday;type_day;month	487.6933	439.032	0.0156	0.1942
LA (alpha=40)	hour;weekday;trimester	490.9901	442.7998	0.0024	0.2159
LA (alpha=40)	hour;weekday	491.0201	442.8282	0.002	0.2158
LA (alpha=40)	hour;weekday;month	491.0641	442.8796	0.0016	0.2158
LA (alpha=60)	hour;type_day	491.5907	443.4728	0	
LA (alpha=80)	hour;weekday;type_day;month	491.599	443.4809	-0.0001	
LA (alpha=80)	hour;type_day;month	491.61	443.491	-0.0002	
LA (alpha=60)	hour;type_day;trimester	491.6071	443.4936	-0.0002	
LA (alpha=80)	hour;type_day;trimester	491.6044	443.4846	-0.0002	
LA (alpha=100)	hour;type_day;trimester	491.6118	443.4888	-0.0002	
LA (alpha=60)	hour;weekday	491.6054	443.4811	-0.0002	
LA (alpha=60)	hour;weekday;month	491.6081	443.4892	-0.0002	

LA (alpha=60)	hour;weekday;trimester	491.6048	443.4905	-0.0002	
LA (alpha=100)	hour;weekday;trimester	491.6103	443.4897	-0.0002	
LA (alpha=60)	hour;weekday;type_day;month	491.6072	443.4778	-0.0002	
LA (alpha=100)	hour;weekday;month	491.6161	443.4967	-0.0003	
LA (alpha=100)	hour;weekday;type_day;month	491.6175	443.5017	-0.0003	
LA (alpha=40)	hour	491.6282	443.5164	-0.0004	
LA (alpha=60)	hour	491.6241	443.5032	-0.0004	
LA (alpha=80)	hour	491.6213	443.4935	-0.0004	
LA (alpha=80)	hour;type_day	491.6215	443.5008	-0.0004	
LA (alpha=80)	hour;weekday;trimester	491.6228	443.4953	-0.0004	
LA (alpha=80)	hour;weekday	491.6308	443.5071	-0.0005	
LA (alpha=80)	hour;weekday;month	491.6383	443.514	-0.0005	
LA (alpha=100)	hour	491.6392	443.5212	-0.0006	
LA (alpha=100)	hour;type_day	491.6393	443.5142	-0.0006	
LA (alpha=60)	hour;type_day;month	491.6406	443.5186	-0.0006	
LA (alpha=100)	hour;type_day;month	491.6545	443.5265	-0.0007	
LA (alpha=100)	hour;weekday	491.6519	443.5239	-0.0007	
NN (num_layers=8, num_perceptrons=2)	hour;type_day;trimester	495.4623	376.1748	-0.0153	0.4506
NN (num_layers=2, num_perceptrons=2)	hour;weekday;type_day;month	499.1812	380.1843	-0.0386	
NN (num_layers=11, num_perceptrons=2)	hour	506.6407	394.9236	-0.0612	
NN (num_layers=2, num_perceptrons=2)	hour;type_day;month	512.3853	400.7018	-0.0896	
NN (num_layers=2, num_perceptrons=2)	hour;weekday;trimester	523.7011	419.3229	-0.1354	
NN (num_layers=5, num_perceptrons=2)	hour;weekday;trimester	545.4534	371.3924	-0.2316	
NN (num_layers=2, num_perceptrons=2)	hour;weekday;month	581.9448	433.7337	-0.4057	

Anexo B

Despliegue de una red de sensores

En este anexo se va a explicar cómo se ha configurado una red de sensores de bajo coste para la medida de diferentes contaminantes atmosféricos, con el objetivo de recoger datos sobre los niveles de contaminación de toda la ciudad de Zaragoza y poder analizar dichos datos.

B.1. Introducción

En la ciudad de Zaragoza, dentro del proyecto TRAFAIR, se ha desplegado una red de sensores LoRa de bajo coste para así poder tomar medidas de la concentración de varios contaminantes atmosféricos. Concretamente, se han desplegado 10 sensores repartidos por toda la ciudad. El objetivo de este trabajo es el desarrollo de una infraestructura que permita administrar dichos sensores, de forma que se pueda:

- Recoger las medidas de los valores de contaminación realizadas por los sensores. La frecuencia con la que un sensor envía una medida es 10 minutos. Concretamente permitirán tomar medidas de la concentración de diferentes compuestos: NO, NO_2, CO y O_3.

- Gestionar los sensores de la red.

- Almacenar las medidas de los sensores.

- Visualizar los datos recogidos por los sensores.

Para la realización de este trabajo se parte con la red de sensores ya desplegada en la ciudad y emitiendo por la red TTN [20]. Es decir, se va a realizar una migración de una infraestructura externa pública a una interna privada.

B.2. Infraestructura del sistema

En la figura B.1 se ilustra la infraestructura del sistema de sensores desarrollado. Los componentes a desplegar en el presente trabajo son los situados dentro de la red interna, mientras que, tanto los sensores como los gateways, se encuentran ya desplegados. Además, puede observarse que la solución está basada en el stack ChirpStack[1].

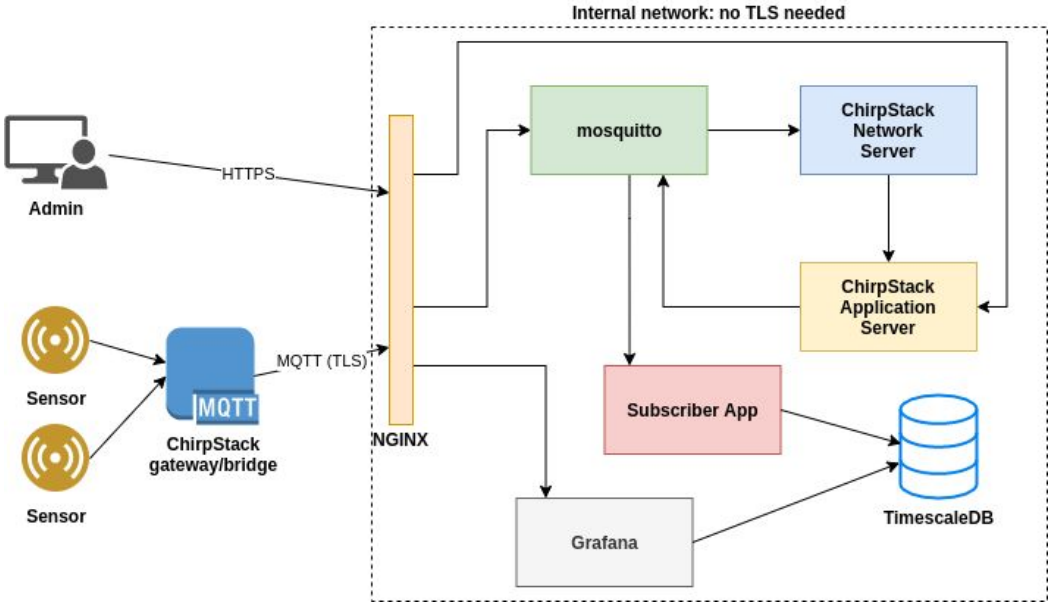

Figura B.1: Infraestructura del sistema de sensores.

Por otro lado se observa que el sistema dispone de dos puntos de entrada: uno tanto para la administración de ChirpStack como para la visualización de los datos vía web, y otro, para la recepción de datos de los sensores mediante el protocolo MQTT [19].

Así pues, el camino que sigue una medida de contaminación de un sensor es el siguiente: el sensor que realiza la medida la envía al gateway correspondiente. Dicho gateway la manda al servidor mediante el protocolo MQTT. Esta medida atraviesa el componente NGINX [21] y es publicada en el broker Mosquitto [22], para seguidamente ser leída por el componente ChirpStack Network Server el cual, tras realizar una serie de comprobaciones, se la envía al componente ChirpStack Application Server. Éste procesa dicha medida y la publica en el broker de mensajes. Finalmente, dicha medida procesada es leída por el componente Subscriber App y almacenada en la base de datos.

Por otro lado, tal y como se ha comentado en la introducción, los sensores se encuentran emitiendo por la red TTN. Dichas medidas son procesadas y publicadas en un broker de mensajes al que el proyecto TRAFAIR tiene acceso para así poder almacenar dichas medidas en una base de datos. Sin embargo, gracias a la nueva

[1] `https://www.chirpstack.io/`

infraestructura privada se ha ganado tanto privacidad como flexibilidad para realizar los cambios que se consideren oportunos.

B.3. Configuración y despliegue

En esta sección se detalla la configuración de cada uno de los componentes del sistema y cómo se han desplegado dichos componentes.

B.3.1. Sensores y gateways

Como ya se ha comentado, en este trabajo no se ha realizado la configuración de los gateways ni la de los sensores. Sin embargo, hay que comentar que ya se encuentran desplegados y que los gateways tienen la posibilidad de enviar los datos tanto cifrados (sobre TLS) como sin cifrar.

B.3.2. NGINX

El componente NGINX es un proxy inverso que actúa como único punto de entrada del sistema. Se encarga de realizar las distintas operaciones en nombre de los clientes. Por lo tanto, los gateways del sistema nunca llegan a interaccionar directamente con los componentes internos de la aplicación, sino que se comunican con NGINX y este se comunica con los otros componentes. Por otra parte, al ofrecer NGINX un único punto de entrada al sistema, es posible tener una IP única para acceder a todos los servicios, aunque se encuentren en máquinas físicas distintas.

Por otra parte, tal y como se observa en la figura B.1, este componente dispone de dos puertos abiertos al exterior: uno por el que recibir las peticiones HTTP que dan acceso a la aplicación de visualización y a la aplicación de administración de ChirpStack, y otro, por el cual los gateways envían las medidas realizadas por los sensores.

Además, NGINX es el punto donde se realiza buena parte de la configuración de seguridad, puesto que es el único componente que está conectado con internet. En él se configuran los certificados para poder usar HTTPS y MQTT sobre TLS, para que el acceso a las aplicaciones web desde el exterior sea seguro y para que los gateways puedan enviar datos cifrados. También se han realizado otras configuraciones de seguridad para las aplicaciones web:

- Se ha limitado el número de peticiones que puede realizar un cliente para así intentar mitigar los ataques DDoS.

- Se impiden los ataques XSS.

– Otras configuraciones menores.

Por otro lado, como el resto de componentes se encuentran en una red interna no es necesario aplicarles estas políticas de seguridad, ya que el propio NGINX actúa como cortafuegos.

Además, se ha configurado el componente para que el administrador pueda acceder a una aplicación web u otra en función del path. Es decir, si el usuario accede a **/grafana** accederá a la aplicación de monitorización, mientras que si accede a **/chirpstack** accederá a la aplicación de administración de ChirpStack.

Finalmente, cabe destacar que se ha decidido utilizar este software y no otro ya que es *open source* y está ampliamente utilizado y testeado.

B.3.3. Mosquitto

Mosquitto es un broker de mensajería que implementa el protocolo MQTT. En él, los gateways publican las medidas de los sensores en un determinado tópico y estas son leídas por el ChirpStack Network Server.

En cuanto a las configuraciones, lo único a resaltar es que se ha activado la persistencia en el broker, para que, en caso de caída, no se pierdan los mensajes que aún no han sido consumidos.

Finalmente, como en el caso anterior, se ha empleado este software porque es *open source* y su uso es generalizado en la red.

B.3.4. ChirpStack Network Server

Es una implementación del **LoRaWAN® Network Server**. Entre otras cosas, se encarga de:

– La autenticación, es decir, de comprobar que un mensaje ha sido generado por alguno de los sensores de la red.

– La deduplicación de mensajes, es decir, de eliminar mensajes duplicados.

Como en los casos anteriores, se ha utilizado este software porque es *open source*.

B.3.5. ChirpStack Application Server

Es una implementación del **LoRaWAN® Application Server**. Entre otras cosas, se encarga de:

– Gestionar sensores y gateways del sistema: dar de alta, de baja, modificar parámetros...

– Decodificar los mensajes enviados por los sensores, transformándolos al formato JSON. Una vez decodificados, son publicados a través de un tópico en el componente mosquitto del sistema.

Además, ofrece una interfaz web a través de la cual es posible gestionar los dispositivos, gateways y toda la infraestructura en general.

En cuanto a la configuración de este componente, ha sido necesario realizar una serie de pasos para poner en marcha la infraestructura. En primer lugar ha sido necesario conectarlo con el ChirpStack Network Server, tal y como se muestra en la figura B.2.

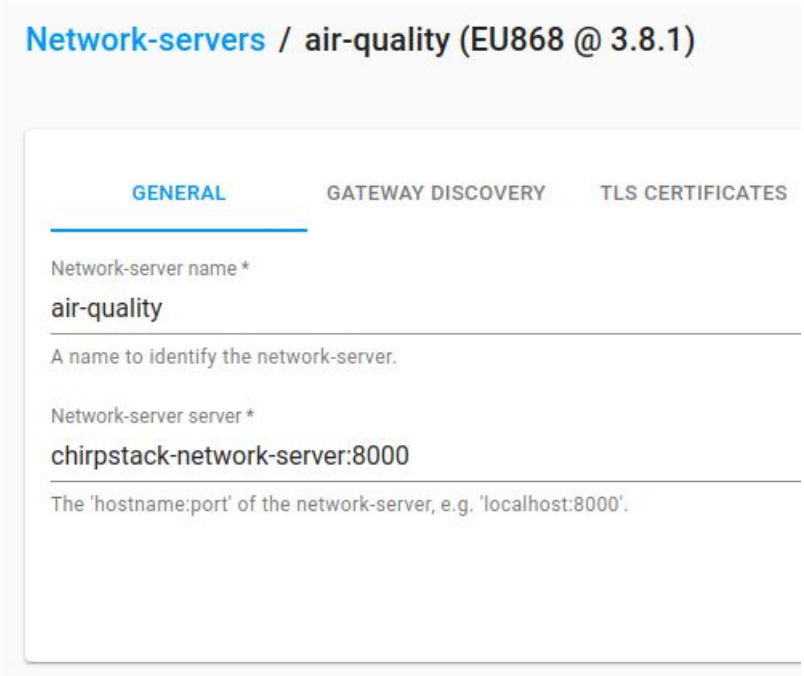

Figura B.2: Conexión del Application Server con el Network Server.

Una vez realizada la conexión con el Network Server, es necesario crear una aplicación. Una aplicación consiste básicamente en una agrupación de dispositivos LoRa.

Posteriormente hay que dar de alta los gateways de la aplicación. Es obligatorio conocer el identificador de cada gateway para poderlo dar de alta en el sistema. Además, hay que configurar el gateway para que envíe los datos a la IP del sistema. En la figura B.3 se muestran los gateways creados en el sistema.

Una vez creada la aplicación y dados de altas los gateways del sistema ya es posible dar de alta los sensores en la aplicación. Para ello, en primer lugar es necesario crear un *device profile* en el que hay que indicar varios parámetros:

– La versión MAC de los dispositivos que van a hacer uso de dicho perfil. Los sensores del sistema usan la versión 1.0.

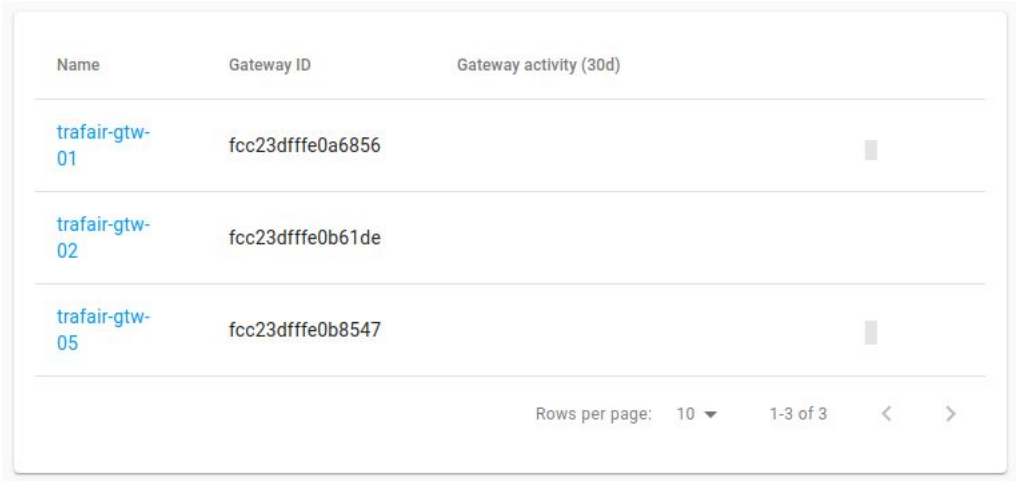

Figura B.3: Gateways dados de alta.

- Si los sensores soportan OTAA (Over-the air activation) o no. En caso afirmativo, los sensores y ChirpStack generan las claves de sesión cuando el sensor es dado de alta. Los sensores del sistema sí lo soportan.
- El decodificador. Este consiste en un código javascript que se encarga de transformar los datos generados por los sensores. En este caso, se transforman los datos enviados por los sensores al formato JSON.

Cabe destacar que como los sensores del sistema son iguales solo se ha creado un único perfil. Si hubiera diferencias (por ejemplo, un sensor que no soporta OTAA) hubiera sido necesario crear al menos dos perfiles.

Una vez creado el perfil ya puede procederse a dar de alta los dispositivos en la aplicación. Para ello hay que indicar:

- El device-profile que usará el dispositivo.
- El EUI del dispositivo. Es un identificador que viene fijado por el fabricante del sensor.
- La Application Key, que tiene que ser conocida tanto por el sensor como por ChirpStack. Sirve para generar las claves de sesión.

En la figura B.4 puede verse los dispositivos que tiene la aplicación creada. Finalmente, una vez dado de alta un sensor se generan las claves de sesión para validar tanto la integridad como la autoría de los mensajes.

B.3.6. Subscriber App y TimescaleDB

El componente Subscriber App es una aplicación escrita en Python que se suscribe al tópico en el que es componente ChirpStack Application Server publica los mensajes

Last seen	Device name	Device EUI	Device profile	Link margin	Battery
n/a	trafair_zgz_04017	70b3d57ba0000fb1	trafair-devices	n/a	n/a
n/a	trafair_zgz_04018	70b3d57ba0000fb2	trafair-devices	n/a	n/a
n/a	trafair_zgz_04019	70b3d57ba0000fb3	trafair-devices	n/a	n/a
n/a	trafair_zgz_04020	70b3d57ba0000fb4	trafair-devices	n/a	n/a
n/a	trafair_zgz_04021	70b3d57ba0000fb5	trafair-devices	n/a	n/a
n/a	trafair_zgz_04022	70b3d57ba0000fb6	trafair-devices	n/a	n/a
n/a	trafair_zgz_04023	70b3d57ba0000fb7	trafair-devices	n/a	n/a
n/a	trafair_zgz_04024	70b3d57ba0000fb8	trafair-devices	n/a	n/a
7 months ago	trafair_zgz_04025	70b3d57ba0000fb9	trafair-devices	n/a	n/a
n/a	trafair_zgz_04026	70b3d57ba0000fba	trafair-devices	n/a	n/a

Figura B.4: Dispositivos creados en ChirpStack.

decodificados. Es decir, se encarga de leer los mensajes decodificados e insertarlos en una base de datos.

Como sistema gestor de bases de datos se utiliza TimescaleDB, que básicamente es un PostgreSQL con una extensión para el manejo de series de datos temporales. Esta es la base de datos que se viene utilizando en todo el proyecto.

B.3.7. Grafana

Grafana[2] es un software que permite la visualización vía web de datos de múltiples fuentes mediante la creación de dashboards.

En las figura B.5, B.6 y B.7 se ilustran los dashboards creados. En el primero de ellos, se visualizan los datos en crudo recogidos por los sensores y su estado, así como la batería restante. En las otras dos figuras, se ven las medidas de estos sensores una vez calibrados y un mapa con la localización de los sensores en la ciudad.

B.3.8. Despliegue de los componentes

Se ha decidido desplegar toda la infraestructura desarrollada, excepto la base de datos, en una única máquina virtual puesto que la carga de trabajo del sistema es pequeña: solamente hay 10 sensores que emiten un dato cada 10 minutos y únicamente 3 ó 4 personas van a tener acceso tanto a las aplicaciones de administración como a

[2] `https://grafana.com/`

Figura B.5: Dashboards en Grafana de datos en crudo y estado.

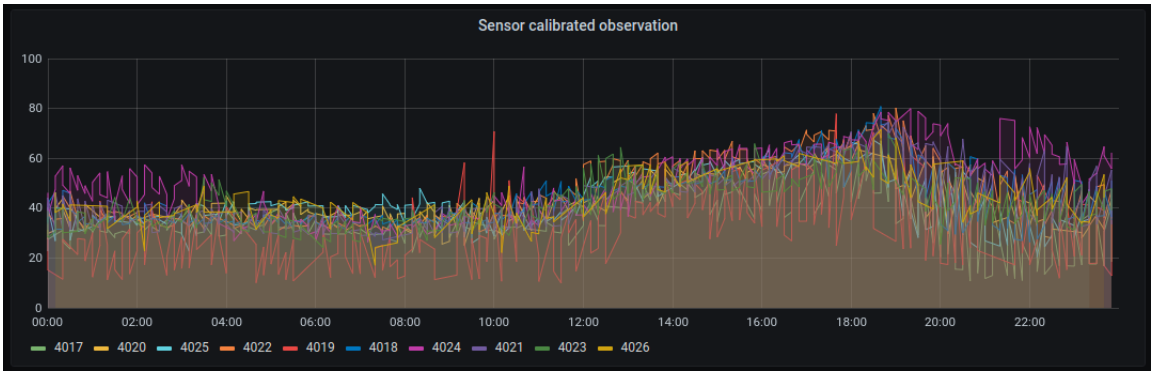

Figura B.6: Dashboards en Grafana de datos calibrados.

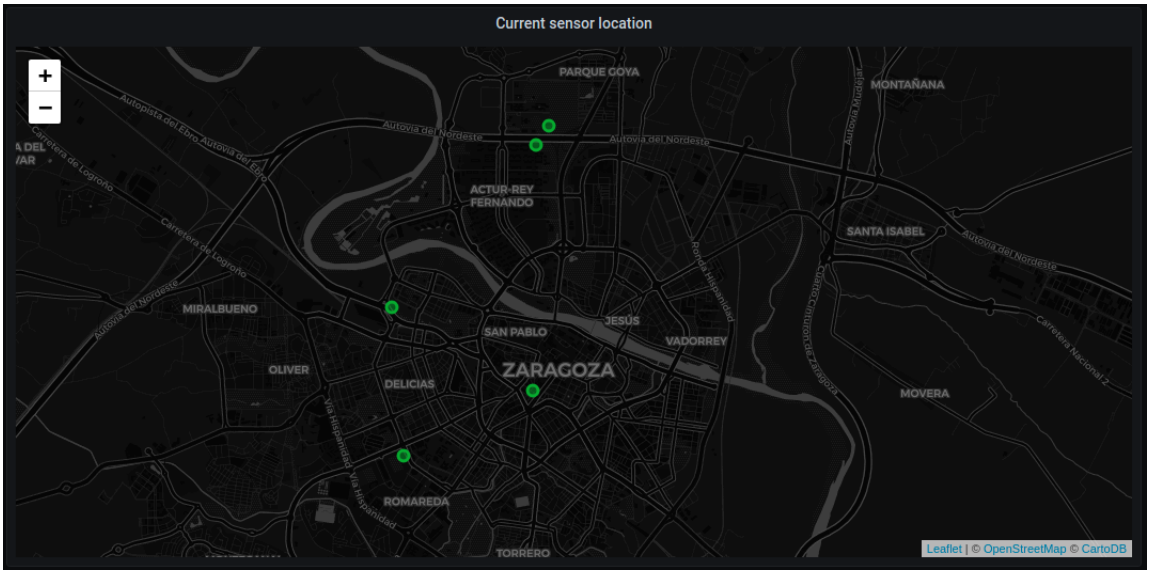

Figura B.7: Dashboards en Grafana con la ubicación de los sensores.

la de visualización. Como base de datos se utiliza una instancia el PostgreSQL que se viene usando a lo largo del trabajo. El nombre de TimescaleDB viene porque se le ha añadido una extensión para facilitar el tratamiento de datos temporales.

Para el despliegue de los componentes se ha seguido una aproximación basada en contenedores de Docker [23], en la que cada uno de los componentes vistos en la figura B.1 se despliega dentro de un contenedor. Además, para agilizar dicho despliegue se ha utilizado Docker Compose[3].

[3] `https://docs.docker.com/compose/`