



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Implementación de un Sistema de Información de soporte  
al análisis de emisiones relacionadas con el tráfico de la  
ciudad de València.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Romero García, Carlos

Tutor/a: Mateo Pla, Miguel Ángel

Cotutor/a: Lemus Zúñiga, Lenin Guillermo

CURSO ACADÉMICO: 2021/2022



## Resumen

---

Hoy en día, cada vez son más frecuentes los efectos causados por el calentamiento global. Ante el aumento de la preocupación, un grupo de investigadores del grupo ICTvsCC, han desarrollado un software capaz de aproximar las emisiones de gases de una zona en concreto mediante el cálculo del tráfico. Posteriormente, dicha tecnología ha sido implementada en la ciudad de Valencia mediante un sistema de espiras de medición. Con el objetivo de mejorar la recogida de información y su comprensión, en este proyecto, se pretende realizar un sistema de información el cual permita almacenar los datos y facilitar el posterior uso de éstos, por ejemplo, su visualización. En este TFG se mostrarán los distintos pasos realizados para la obtención de dicho sistema de información, desde la creación de una base de datos que albergue los datos obtenidos hasta la creación de una API que posibilite acceder a ellos de una forma segura.

**Palabras clave:** SI , emisiones, API, CORS, EntryPoints

## Resum

---

Hui en dia, cada vegada són més freqüents els efectes causats pel calfament global. Davant de l'augment de la preocupació , un grup d'investigadors del grup ICTvsCC, han desenrotllat un programari capaç d'aproximar les emissions de gasos d'una zona en concret per mitjà del càlcul del tràfic. Posteriorment, la dita tecnologia ha sigut implementada en la ciutat de València per mitjà d'un sistema d'espires de mesurament. Amb l'objectiu de millorar l'arreglega d'informació i la seua comprensió, en este projecte, es pretén realitzar un sistema d'informació el qual permet emmagatzemar les dades i facilitar el posterior ús d'estos, per exemple, la seua visualització. En este TFG es mostraran els distints passos realitzats per a l'obtenció del dit sistema d'informació. des de la creació d'una base de dades que alberg les dades obtinguts fins a la creació d'una API que possibilita accedir a ells d'una forma segura.

**Paraules clau:** SI , emissions, API, CORS, EntryPoints

# Abstract

---

Today, the effects caused by global warming are becoming more frequent. Given the growing concern, a group of researchers from the ICTvsCC group have developed software capable of approximating gas emissions in a specific area by calculating traffic. Subsequently, this technology has been implemented in the city of Valencia through a system of measuring loops. With the aim of improving the collection of information and its understanding, in this project, it is intended to create an information system which allows data to be stored and facilitates its subsequent use, for example, its visualization. In this TFG the different steps carried out to obtain said information system will be shown, from the creation of a database that houses the data obtained to the creation of an API that makes it possible to access them in a secure way.

**Keywords :** IS, emissions, API, CORS, EntryPoints

# Tabla de contenidos

---

1.	Introducción .....	9
1.1.	Motivación.....	10
1.2.	Objetivos.....	10
1.3.	Impacto esperado .....	10
1.4.	Estructura.....	11
1.5.	Colaboraciones.....	12
2.	Estado del arte .....	13
2.1.	Sistemas de información .....	13
2.2.	API .....	15
3.	Análisis del problema.....	18
3.1.	Identificación y análisis de soluciones posibles.....	19
3.2.	Solución propuesta .....	20
3.3.	Plan de trabajo .....	22
4.	Diseño e implementación .....	23
4.1.	Tecnologías empleadas.....	23
4.2.	Diseño de la base de datos .....	24
4.3.	Diseño API.....	27
4.3.1.	Definición de EntryPoints .....	27
4.3.2.	Diseño clases .....	29
5.	Implantación y validación .....	41
5.1.	Ejemplos prácticos.....	41
6.	Conclusiones.....	45
7.	Trabajo futuro.....	46
8.	Bibliografía .....	47
	Anexo I. Relación con ODS.....	50

# Tabla de figuras, esquemas y tablas

---

## Índice de Figuras

<b>Figura 1:</b> Esquema del funcionamiento básico de un sistema de información .....	14
<b>Figura 2:</b> Componentes de un sistema de información .....	14
<b>Figura 3:</b> Esquema del funcionamiento básico de una API .....	15
<b>Figura 4:</b> Tipos de API.....	16
<b>Figura 5:</b> Métodos básicos HTTP dentro de una API.....	17
<b>Figura 6:</b> Esquema de la solución propuesta .....	21
<b>Figura 7:</b> Plan de trabajo .....	22
<b>Figura 8:</b> Opciones de codificación en NotePad++ .....	24
<b>Figura 9:</b> Datos preparados para almacenar .....	24
<b>Figura 10:</b> Conexión a la base de datos .....	25
<b>Figura 11:</b> Formulario de creación de una base de datos en MongoDB Compass .....	25
<b>Figura 12:</b> Importación del fichero en la base de datos .....	26
<b>Figura 13:</b> Base de datos creada .....	27
<b>Figura 14:</b> Instalación NestJS.....	29
<b>Figura 15:</b> Creación del proyecto .....	29
<b>Figura 16:</b> Instalación de los paquetes y librerías de mongoose .....	30
<b>Figura 17:</b> Instalación librería Swagger.....	30
<b>Figura 18:</b> Creación módulo database.....	30
<b>Figura 19:</b> Creación módulo DatosEspiras .....	31
<b>Figura 20:</b> Creación clase myConfig.ts .....	31
<b>Figura 21:</b> Creación clase database-providers .....	31
<b>Figura 22:</b> Clase database.module.ts .....	31
<b>Figura 23:</b> Definición EntryPoints en la clase controladora (Parte 1).....	32
<b>Figura 24:</b> Definición EtryPoints en la clase controladora (Parte 2).....	33
<b>Figura 25:</b> Declaración del formato del DTO .....	33
<b>Figura 26:</b> Interfaz i-datos-espigas-interface.ts .....	34
<b>Figura 27:</b> Interfaz i-return-inteface.ts .....	34
<b>Figura 28:</b> Clase datos-espigas-schema.ts .....	35
<b>Figura 29:</b> Clase datos-espigas.ts.....	35
<b>Figura 30:</b> Método finTramos() .....	36
<b>Figura 31:</b> Método findAllEspirasInTramo().....	36
<b>Figura 32:</b> Método ocupaciónTramo().....	37
<b>Figura 33:</b> Método intensidadTramo() .....	38
<b>Figura 34:</b> Método deleteTramo().....	38
<b>Figura 35:</b> Estructura final del proyecto .....	39
<b>Figura 36:</b> Ejecución del proyecto.....	39
<b>Figura 37:</b> Sitio web de la API realizado con Swagger .....	40
<b>Figura 38:</b> Consulta Entrypoint 3 ( Listar espigas pertenecientes a un tramo) .....	41
<b>Figura 39:</b> Respuesta a la petición de listar todas las espigas relacionadas con cierto tramo .....	42
<b>Figura 40:</b> Consulta Entrypoint 7 ( Intensidad media en una espiga determinada en una franja horaria determinada .....	42
<b>Figura 41:</b> Cuerpo del formulario o DTO con las horas deseadas .....	42
<b>Figura 42:</b> Respuesta a la petición de la intensidad media .....	43

<b>Figura 43:</b> Petición de eliminación de una espira .....	43
<b>Figura 44:</b> Respuesta a la petición de eliminación .....	43
<b>Figura 45:</b> Petición de obtención de la intensidad media de una espira eliminada .....	44

## ***Índice de Tablas***

<b>Tabla 1:</b> Formato ficheros generados por las espiras.....	18
<b>Tabla 2:</b> Ventajas y desventajas del uso de bases de datos relacionales y no relacionales .....	20
<b>Tabla 3:</b> Ventajas y desventajas del uso de APIs SOAP o REST .....	20
<b>Tabla 4:</b> Definición de los EntryPoints de la API a desarrollar .....	28
<b>Tabla 5:</b> Grado de relación del proyecto realizado con los ODS .....	50



Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

# 1. Introducción

---

El aumento de la población mundial y de la demanda de productos (entre otros muchos factores) han generado un gran número de fenómenos que, hasta hace escasas décadas, se desconocían o no se consideraban de gran importancia. Durante los últimos años, la comunidad científica internacional ha investigado y analizado dichos fenómenos y sus efectos en el planeta y en la humanidad. Se ha observado un aumento de las emisiones de gases de efecto invernadero (GEI) producidos por la humanidad [1], que están produciendo un aumento de la temperatura global del planeta. También se ha detectado la aparición de sustancias peligrosas para la salud y el medio ambiente [2]. Otros efectos que se achacan hoy día a la actividad humana son: son el deshielo de los polos, la subida de los niveles del mar y la desaparición de ciertas especies.

La comunidad científica ha agrupado la suma de dichos fenómenos y sus consecuencias en el concepto de **cambio climático**. Según la Real Academia Española (RAE), se define el cambio climático [3] como: *“Cambio del clima, atribuido directa o indirectamente a la actividad humana, que altera la composición de la atmósfera mundial y que se suma a la variabilidad natural del clima observada durante períodos de tiempo comparables”*

El cambio climático se ha convertido en un reto principal para la humanidad y todas las naciones, por lo que, con el afán de reducir sus efectos, las instituciones mundiales han explorado e invertido en soluciones innovadoras.

Una de las medidas sugeridas, es la aplicación de tecnologías de la información en las ciudades con la finalidad de medir, controlar y disminuir las emisiones de GEI generadas. A esta fusión de tecnología y ciudad se la conoce con el nombre de SmartCity. Desde la Oficina de Ciudad Inteligente [4], definen SmartCity como: *“Una Ciudad Inteligente y Sostenible es una ciudad innovadora que aprovecha las Tecnologías de la Información y la Comunicación (TIC) y otros medios para mejorar la calidad de vida, la competitividad, la eficiencia del funcionamiento y los servicios urbanos, al tiempo que se asegura de que responde a las necesidades de las generaciones presente y futuras en lo que respecta a los aspectos económicos, sociales, medioambientales y culturales.”*

Con el objetivo de digitalizar la ciudad de Valencia y convertirla en una SmartCity, se han llevado a cabo diversos proyectos. En los últimos años, investigadores del grupo de Tecnologías de la Información y Comunicación contra el Cambio Climático (ICTvsCC), han desarrollado un sistema de estimación de contaminación con GEI en Valencia basándose en los datos de tráfico de una vía [5], utilizando para ello el sistema de control de tráfico de la ciudad. Este sistema de control de tráfico está compuesto por balizas de medición distribuidas en las vías de la ciudad. Cada baliza genera una gran cantidad de datos durante su funcionamiento. Teniendo en cuenta esto y el número de espiras, más de mil, aparece un problema de almacenamiento y análisis de los datos en bruto y de los derivados.

En este contexto, este TFG persigue mejorar la recogida de datos del sistema de control de tráfico, la generación de información a partir de los mismos y, finalmente, su análisis. Para lo cual se pretende la realización de un sistema de información o SI. Un sistema de información se define como [6]: *“Un conjunto de componentes que permiten recoger, almacenar y procesar datos con el objetivo de obtener conocimiento, información o productos digitales”*.

Para ello, en este trabajo, se mostrará el proceso de creación de una base de datos en MongoDB, la cual almacenará los datos generados por las espiras de medición. Por otro lado, se desarrollará una API REST que permitirá acceder a dicha base de datos, facilitando las tareas de análisis y comprensión de ellos mismos.

## 1.1. Motivación

---

El almacenamiento y procesado de datos se ha convertido en parte esencial en cualquier proceso empresarial o de investigación. En consecuencia, las empresas y las instituciones han aumentado la inversión e investigación en esta área de la informática. Para realizar el almacenamiento y su posterior procesado de datos, los expertos pueden hacer uso de los llamados Sistemas de información o SI, los cuales facilitan las labores de recolección y análisis de los datos.

En lo personal, dicho campo me resulta sumamente interesante debido a la gran cantidad de aplicaciones que disponen los SI en distintas áreas de la informática. Utilidades diversas tanto para entidades públicas como privadas, como, por ejemplo, facilitar la toma de decisiones, ayudar a controlar el flujo de información o procesar rutinas diarias dentro de la organización [6].

En esta ocasión, se planea usar el SI para facilitar la recogida y la comprensión de los datos generados por una serie de espiras de medición, con el afán de comprender la contaminación que genera la ciudad de Valencia. Trabajar en un proyecto que pretende investigar la contaminación de mi ciudad para así generar nuevas soluciones para reducirla, también me resulta motivador.

Finalmente, la realización del proyecto pondrá a prueba mis conocimientos adquiridos a lo largo de mis estudios universitarios e indagaré aún más en los relacionados con mi especificación.

## 1.2. Objetivos

---

El objetivo principal del proyecto es diseñar, implementar y evaluar un sistema de información dedicado a la captura y exposición de los datos obtenidos por un sistema de medición de tráfico.

Para alcanzar satisfactoriamente dicho objetivo principal de este trabajo, se han planteado los siguientes objetivos específicos:

- Preparar los datos generados por las espiras, para su inclusión en la base de datos aplicando los conceptos ETL.
- Diseñar e implementar un sistema de base de datos utilizando MongoDB para almacenar los datos ya preparados.
- Diseñar e implementar una API REST que permita acceder a los datos almacenados en la base de datos.
- Evaluar la efectividad del sistema mediante la realización de pruebas piloto.

## 1.3. Impacto esperado

---

Una vez alcanzados los objetivos planteados en este trabajo, se espera obtener un sistema de información el cual suponga una mejora respecto a la situación actual.

La correcta implementación de un sistema de información permitirá controlar de forma más efectiva los datos obtenidos por el sistema de medición de tráfico, permitiendo un posterior uso de ellos con distintas

finalidades. También supondrá una integración de distintas metodologías en una sola herramienta, lo que aumentará considerablemente la seguridad y efectividad del proceso.

Por otro lado, este proyecto contribuye con algunos de los objetivos de desarrollo sostenible (ODS) establecidos por la ONU en 2015 [7]. Siendo, los más relacionados con el trabajo, el objetivo de transformar las ciudades y comunidades para que sean sostenibles y el objetivo de acción por el clima. Al final de la memoria se adjuntará en el anexo una tabla en la cual se relacionará el TFG con los ODS acompañados de una reflexión personal acerca de su relación (ver Anexo I. Relación con ODS).

## 1.4. Estructura

---

El contenido del trabajo y su estructura se dividirá en 7 capítulos, una bibliografía y finalmente un Anexo. A continuación, se muestran cada una de las partes de la memoria, acompañados de una breve descripción:

- **Capítulo 1, Introducción:** Se expone brevemente el problema inicial, la motivación que me ha llevado a realizar el trabajo, los objetivos planteados, el impacto esperado tras la realización del proyecto y las colaboraciones realizadas.
- **Capítulo 2, Estado del arte:** En esta sección se mostrará la situación actual del procesado y almacenamiento de datos, de los sistemas de información y de las API, así como de los problemas a los que se enfrentan.
- **Capítulo 3, Análisis del problema:** Explicación de la problemática del sistema actual. Además, se identificarán las posibles soluciones al problema y las herramientas que se finalmente se han utilizado.
- **Capítulo 4, Diseño e implementación:** Se expondrá de forma detallada el desarrollo y diseño de la base de datos, así como de la API REST y su funcionalidad.
- **Capítulo 5, Implantación y validación:** Realización y exposición de pruebas piloto del sistema de información desarrollado.
- **Capítulo 6, Conclusiones:** Resumen de la fase de desarrollo y análisis de la solución obtenida. Además, se relacionará el trabajo con el grado cursado acompañado de una reflexión personal.
- **Capítulo 7, Trabajo futuro:** Se incluirán posibles mejoras en el sistema para enfrentar a posibles problemas que surgen durante el paso del tiempo.
- **Bibliografía:** Las referencias bibliográficas utilizadas y consultadas a lo largo de la realización del proyecto.
- **Anexo I. Relación con ODS:** Se adjuntará una tabla la cual relaciona el trabajo realizado con los distintos objetivos del desarrollo sostenible (ODS) acompañados con una reflexión.

## 1.5. Colaboraciones

---

Como se ha comentado previamente, este trabajo colabora con el grupo de investigación de Tecnologías de la Información y Comunicación contra el cambio climático (ICTvsCC), puesto que el sistema de espiras de medición, así como del origen de los datos, ha sido desarrollado por ellos en un trabajo previo.

Por otro lado, en este trabajo colaborará con mi compañera Elena López Valero, estudiante de la UPV. Ambos trabajaremos acerca del mismo sistema de espiras de tráfico y con los mismos datos. El TFG de Elena está enfocado al análisis minucioso de los datos generados por las espiras de medición, mientras que mi objetivo es el desarrollar un sistema que mejore la captación y exposición de dichos datos. Durante el desarrollo del trabajo se han realizado colaboraciones en algunos aspectos del proyecto, tales como la preparación de los datos o la selección de los datos más relevantes para el análisis posterior de ellos.

## 2. Estado del arte

---

En la introducción del proyecto, se ha comentado como el área informática del procesado y almacenamiento de datos ha sufrido un gran aumento de demanda por parte de empresas, grupos de investigación, instituciones públicas, etc.

Según un estudio de IBM de 2017 [8], los empleos relacionados con la ciencia de datos y su análisis crecerán en un 28% para 2020. Dicha previsión se ha convertido en una realidad en la actualidad, puesto que cada vez más organismos, empresas y grupos de investigación demandan trabajadores con conocimientos en el área del procesado y almacenamiento de datos. El aumento de demanda es comprensible teniendo en cuenta la gran cantidad de ventajas que aporta esta rama de la informática. De entre ellas destacan la toma de decisiones fundamentadas en los datos, la unificación de diversas fuentes de datos en una sola y la obtención de un registro histórico de los datos.

Para realizar un procesamiento de los datos y posteriormente obtener información útil, los analistas usan herramientas de inteligencia empresarial (BI), SAS, Excel y otras distintas aplicaciones de análisis. En cuanto al almacenamiento, se pueden emplear bases de datos, almacenes de datos, etc. De entre las herramientas o mecanismos usados para almacenar los datos, destacan los sistemas de información o SI. El uso de los SI ha ido creciendo en popularidad y cada vez son más empleados por los expertos en el área debido a las ventajas que ofrecen.

### 2.1. Sistemas de información

---

El uso de los SI abarca muchas áreas y actividades diarias [9]. algún ejemplo de sistema de información son los sistemas utilizados por las entidades bancarias para la realización de operaciones de transacción, registro, nominas, envíos, etc. Los SI también son utilizados como sistemas de apoyo a las decisiones en empresas, sistemas de control de calidad, como sistemas de gestión documental, como archivos de bibliotecas o como sistemas de mensajería.

Existen varias formas de definir que es un sistema de información o SI. De acuerdo con [10], un SI se define como: *“Un conjunto de componentes que permiten recoger, almacenar y procesar datos con el objetivo de obtener conocimiento, información o productos digitales”*. Como se menciona en la cita anterior, el objetivo de un SI es el de obtener conocimiento e información, lo que quiere decir que el elemento principal el cual se ha de diseñar un SI es su uso posterior y sus futuros usuarios.

Según Stair y Reynolds en su obra Principios de Sistemas de información [11], el funcionamiento de un sistema de información está formado por diversas fases. A continuación, se explican en que consiste cada una de ellas:

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.



*Figura 1: Esquema del funcionamiento básico de un sistema de información*

- **Entrada de datos:** Consiste en la recopilación y captura de los datos que deseamos almacenar y posteriormente procesar. Los datos pueden proceder de distintas fuentes como, por ejemplo, documentos físicos, bases de datos independientes o dispositivos que generan datos de interés.
- **Procesamiento de los datos almacenados:** Se realizan las conversiones o transformaciones necesarias de los datos de entrada (conocido como ETL). Una vez realizados dichos procedimientos, se almacena de manera conjunta y unificada todos los datos de entrada.
- **Salida de la información:** A partir de los datos unificados y tratados, se obtiene información útil para el usuario que utilice el SI. Esta información puede ser presentada en forma de documentos, reportes o incluso puede ser utilizada como entrada de otro sistema o programa, como por ejemplo una API.
- **Retroalimentación del sistema:** Parte de la información generada por la salida del SI, puede ser utilizada para realizar cambios en las fases de entrada y de procesamiento en caso de error o problema.

Una vez definido el funcionamiento de un SI, es importante indicar las distintas partes que lo componen. De acuerdo con [12] un sistema de información está compuesto por 6 partes:



*Figura 2: Componentes de un sistema de información*

- **Hardware:** Equipo de cómputo que se utiliza para llevar a cabo las distintas actividades de entrada, procesamiento y salida.

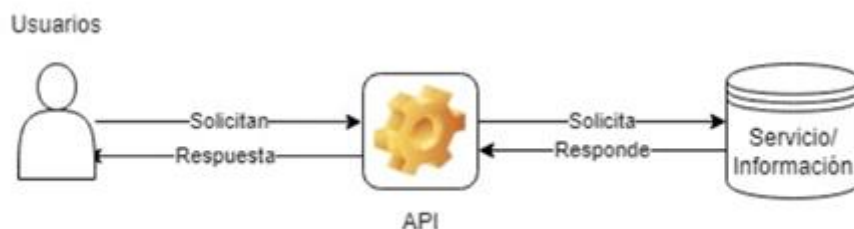
- **Software:** Programas de cómputo que rigen las operaciones del SI.
- **Redes:** Recursos de comunicación.
- **Datos/Hechos:** Conjunto organizado de hechos que precisa organizarse.
- **Gente:** El elemento más importante en la mayoría de SI. Personal encargado de administrar, operar, programar y mantener el sistema de información.
- **Procedimientos:** Políticas, métodos, reglas, pasos a seguir, estaciones de trabajo, etc.

Como anteriormente se ha mencionado, la información generada en la fase de salida por un SI puede ser utilizada como entrada para otro SI o programa. En este proyecto se va a utilizar una API la cual accederá a la información generada por el sistema de información.

## 2.2. API

---

De acuerdo con la guía de Buenas Prácticas en el diseño de APIs y Linked Data [13] creada por el ministerio de asuntos económicos y transformación digital de España, el concepto de API se define como *“La interfaz de programación de aplicaciones, abreviada del término inglés API (Application Programming Interface), es la suma de subrutinas, funciones, procedimientos o métodos que un sistema de información ofrece para ser utilizado por otro sistema”*.



**Figura 3:** Esquema del funcionamiento básico de una API

Existen innumerables ejemplos de API que utilizamos en el día a día. La API de Google Maps [14] que permite a otras aplicaciones acceder a sus mapas y su sistema de geolocalización, la API de YouTube [15] que permite integrar videos en una página web, APIs de cartera digital como PayPal [16], la API de Amazon [17] que permite enlazar un producto, etc.

El extenso uso de las APIs como mecanismo de intercambio de información entre dos componentes de software independientes es debido a las diversas ventajas que aportan. Según la plataforma de desarrollo colaborativo de software, GitHub, los programadores y las empresas encuentran algunas de las siguientes ventajas al utilizar las APIs [18] frente a otros mecanismos:

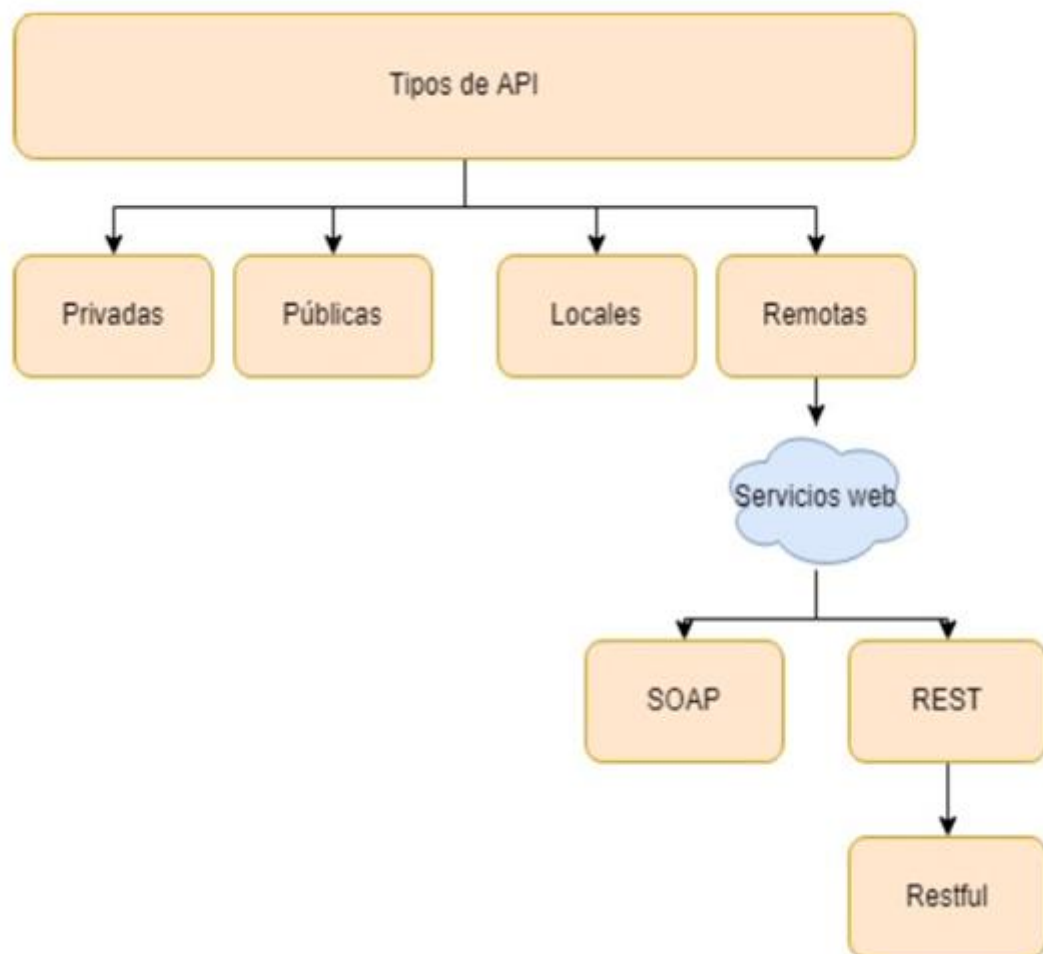
- **Eficiencia:** los contenidos basados se publican de forma automática y en diversos canales, aumentando la reutilización de los recursos y reduciendo la sobrecarga.
- **Seguridad/Independencia:** la APIs son seguras ya que actúan como una barrera entre las solicitudes de acceso y la información almacenada por el sistema.



## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

- **Flexibilidad:** En caso de error o cambio, las APIs poseen una gran capacidad de adaptación debido a su sistema de migración de datos.
- **Compatibilidad:** Cualquier dispositivo, usuario, sistema o programa puede acceder a los contenidos de la API independientemente de su sistema de soporte.
- **Personalización:** Los usuarios que acceden a la API pueden personalizar los datos que contengan de acuerdo con sus intereses, sin comprometer la información original, permitiendo adaptar protocolos o funciones según el usuario.

A la hora de diseñar una API, nos encontramos con distintos tipos dependiendo de diversos tipos de factores y características. De entre todos los tipos, se destacan los siguientes:



*Figura 4: Tipos de API*

Dependiendo de su política de uso:

- **Privadas:** Únicamente son empleadas por los sistemas internos de una organización.
- **Publicas:** Están disponibles para que otros usuarios puedan acceder a ellas de forma total o parcialmente restringida.

Dependiendo de su ubicación:

- **Locales:** Los recursos a los que accede el usuario están en el mismo dispositivo desde que los solicita.
- **Remotas:** Los recursos están situados en un dispositivo distinto. Para conectarse de forma remota, las APIs emplean las redes y los servicios red mediante protocolos de conexión.

Dependiendo del protocolo de acceso:

- **SOAP** (*Simple Object Access Protocol*): Define como dos objetos/entidades se comunican mediante el intercambio de datos en formato XML.
- **REST** (*Representational State Transfer*): A diferencia de SOAP, la comunicación se realiza mediante el uso de los métodos de HTTP, permitiendo el envío de los datos en cualquier tipo, ya sea XML, texto, documentos o JSON (El más empleado). A las API que usan el protocolo REST se les conoce como API Restful.

Método HTTP	Operación
GET	Leer todas las entidades dentro de la colección
PUT	Actualización múltiple y/o masiva
DELETE	Borrar la colección y todas sus entidades
POST	Crear una nueva entidad dentro de la colección

**Figura 5:** Métodos básicos HTTP dentro de una API

Recalcar la existencia de más protocolos de acceso, como lo son XML-RPC[19] y JSON-RPC [20], pero debido a su antigüedad y sus problemas de seguridad han sufrido un gran desuso por parte de la comunidad.

### 3. Análisis del problema

Una vez detallado el estado actual del almacenamiento y procesado de datos, así como del estado y funcionamiento básico de los sistemas de información y de las API, se procederá a analizar el problema actual y las posibles soluciones al mismo.

Como se ha comentado en la introducción de este trabajo, el grupo de investigación Tecnologías de la Información y Comunicación contra el Cambio Climático (ICTvsCC), desarrolló un sistema de medición de contaminación de Valencia en base al tráfico de una vía [5]. El sistema está basado en la existencia de balizas o espiras distribuidas a lo largo de la ciudad que detectan el paso de vehículos. Dado el número de espiras y de vehículos, se generan una gran cantidad de datos por minuto durante su funcionamiento.

Los datos de una espira se recogen en la central del Ayuntamiento, donde son procesados y almacenados en dos ficheros “.csv” (*Comma Separated Values*), dependiendo si los vehículos son bicicletas o no. Estos ficheros se generan al final de cada mes. Una vez disponibles ambos documentos, se pueden fusionar con los producidos por el resto de las espiras ese mismo mes. El resultado final es un fichero con las medidas para las bicicletas y otro para el resto de los vehículos. El formato de cada uno de estos ficheros es el descrito en las dos primeras columnas de la **Tabla 1**.

**Tabla 1:** Formato ficheros generados por las espiras

Columnas en .csv	Contenido de las columnas	Nombre del valor almacenado en MongoDB
Columna A	Identificador del punto de medida	IdPM
Columna B	Nombre del punto de medida	Nombre
Columna C	Descripción del punto de medida	Descripción
Columna D	Dirección a la que se dirige la vía actual	Sentido
Columna E	Identificador del tramo	IdTA
Columna F	Fecha y Hora	FechaHora
Columna G	Intensidad de Tráfico por hora	Intensidad
Columna H	Fiabilidad de intensidad de Tráfico	FiabilidadIntensidad
Columna I	Ocupación (Si hay automóviles)	Ocupación
Columna J	Fiabilidad de Ocupación	FiabilidadOcupación
Columna K	Velocidad (Promedio del flujo)	Velocidad
Columna L	Fiabilidad de velocidad	FiabilidadVelocidad

Al disponer de un gran volumen de datos, el grupo ICTvsCC, se enfrenta a tres problemas: su almacenamiento, su acceso y su análisis. Es por ello, por lo que solicitan una solución la cual cumpla una serie de requerimientos.

El sistema cuenta con más de mil espiras. las cuales generan un gran volumen de datos cada 3 minutos. Es por ello por lo que la solución a implementar ha de ser capaz de unificar todos los datos en un solo registro, de forma que se facilite la consulta de estos.

Además, ha de hacerlo de forma eficiente y consumiendo pocos recursos durante su funcionamiento. Esto es debido a que, al disponer de una gran cantidad de datos, pueden presentar un gasto excesivo en los recursos ralentizando las operaciones y las consultas de los datos.

Por otro lado, se desea hacer un análisis de estos en el futuro, por lo que la solución a implementar ha de poseer algún método de acceso al conjunto de datos de forma que permita realizar consultas de una forma amigable con el usuario.

La seguridad es un aspecto fundamental a la hora de trabajar con datos y el acceso a ellos. Si no se presta atención a este aspecto, pueden aparecer problemas como los accesos no autorizados a los datos o una posible corrupción de ellos durante su ciclo de vida. En consecuencia, la seguridad de la solución a desarrollar será un factor clave para un correcto funcionamiento del sistema. La solución ha de definir privilegios y acciones en base a una serie de roles (Autorización) y ha de poseer algún mecanismo que controle el acceso a los datos almacenados (Autenticación)

## 3.1. Identificación y análisis de soluciones posibles

---

En la introducción se ha definido el concepto de sistema de información, siendo éste un conjunto de componentes que permiten recoger, almacenar y procesar datos con el objetivo de obtener conocimiento, información o productos digitales. De entre todas las alternativas disponibles. desde el principio del desarrollo de este proyecto, se ha planteado como solución al problema debido a su capacidad de cumplir el requisito de unificar y almacenar datos provenientes de distintas fuentes.

Para cumplir con este requerimiento, los SI utilizan bases de datos. En la actualidad existen multitud de tipos de bases de datos fundamentadas en una gran variedad de lenguajes de programación. Todas ellas tienen el objetivo fundamental de almacenar información, pero dependiendo de cómo se pretenda almacenar y utilizar dichos datos, de acuerdo con [21] nos podemos encontrar algunas de las siguientes bases de datos:

- **Bases de datos relacionales:** Son las más utilizadas actualmente. Los elementos de la base de datos se organizan mediante el uso de un conjunto de tablas con columnas y filas.
- **Bases de datos orientadas a objetos:** Basada en la programación orientada a objetos, los datos son representados como objetos.
- **Bases de datos distribuidas:** Consta de varias bases de datos situadas en lugares diferentes, ya sea en varios ordenadores o redes.
- **Almacenes de datos:** Diseñados para realizar consultas y tareas de análisis sobre una gran cantidad de datos históricos.
- **Bases de datos NoSQL o no relacionales:** En lugar de utilizar un esquema tabular basado en filas y columnas, utilizan un modelo de almacenamiento de acuerdo unos requisitos específicos de forma no estructurado o semiestructurada.

Entre todos estos tipos de bases de datos expuestas, destacan las bases de datos relacionales y no relacionales debido a su gran uso en la actualidad por parte de la comunidad. En la **Tabla 2** podemos observar las virtudes e inconvenientes del uso de ellas y los requisitos que cumplen.

Para cumplir con el requerimiento de permitir realizar consultas sobre el conjunto de dichos datos de forma segura y amigable con el usuario, se ha optado desarrollar una API. En la actualidad existen una gran

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

variedad de API, pero puesto que la conexión entre el SI y la API se realizará mediante el uso de los recursos web, entre todas las opciones disponibles, destacan las API basadas en la arquitectura REST o SOAP. En la **Tabla 3** se aprecian algunas de las ventajas y desventajas que ofrecen cada una de ellas.

**Tabla 2:** Ventajas y desventajas del uso de bases de datos relacionales y no relacionales

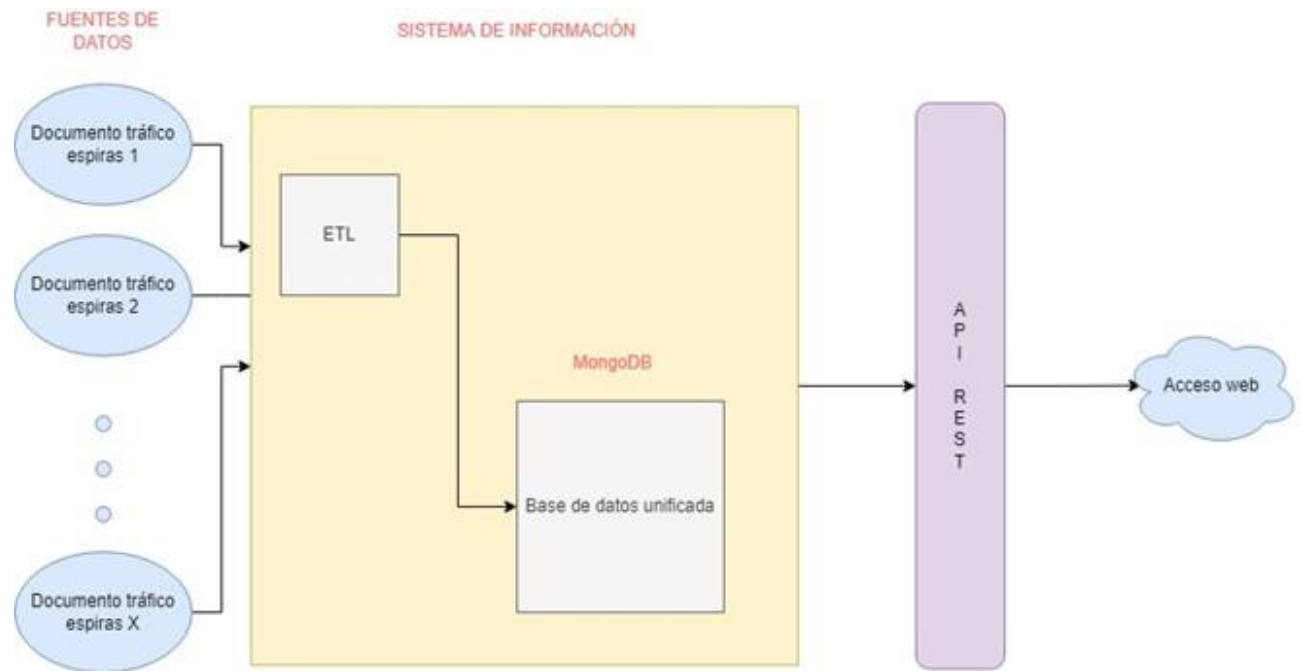
RELACIONALES	NO RELACIONALES
<ul style="list-style-type: none"> <li>- Uso extendido dentro de la comunidad, siendo éstas las más utilizadas actualmente.</li> <li>- Atomicidad en las operaciones sobre la base de datos.</li> <li>- Estándares definidos en las distintas operaciones.</li> <li>- Sencillez en su estructura.</li> <li>- Pérdida de prestaciones con el manejo de grandes cantidades de datos.</li> <li>- Poca flexibilidad a la hora de cambiar la estructura de la base de datos.</li> <li>- Dificultad en su instalación en algunos sistemas operativos.</li> </ul>	<ul style="list-style-type: none"> <li>- Versatilidad ante los cambios en la estructura de almacenamiento.</li> <li>- Poco consumo de recursos.</li> <li>- Optimización en sus operaciones.</li> <li>- Compatible con sistemas distribuidos</li> <li>- Difícil aprendizaje debido a la falta de documentación.</li> <li>- No hay un estándar definido.</li> <li>- Algunas carecen de interfaz gráfica</li> </ul>

**Tabla 3:** Ventajas y desventajas del uso de APIs SOAP o REST

API SOAP	API REST
<ul style="list-style-type: none"> <li>- Utiliza un formato simple en sus mensajes (XML).</li> <li>- Más estandarizado.</li> <li>- Alta seguridad entre origen y destino.</li> <li>- Simpleza en su funcionamiento.</li> <li>- Mayor tiempo de carga, por lo tanto, peor eficiencia.</li> <li>- Carece de almacenamiento en caché.</li> <li>- Ciertos lenguajes no dan soporte a su uso.</li> </ul>	<ul style="list-style-type: none"> <li>- Escaso uso de los recursos. REST es más eficiente.</li> <li>- Fácil de integrar.</li> <li>- Puede enviar el mensaje en muchos tipos de formato.</li> <li>- Son más flexibles y se configuran con mayor facilidad.</li> <li>- No importa la tecnología que el usuario utiliza.</li> <li>- Dispone de almacenamiento en caché.</li> <li>- La seguridad puede ser un punto débil si no se sabe cómo gestionar con cautela.</li> <li>- No hay estándares de respuesta.</li> <li>- Mayor coste de desarrollo.</li> </ul>

## 3.2. Solución propuesta

Una vez expuestas las distintas alternativas que se han considerado para la creación de la base de datos del SI y del desarrollo de la API que lo accederá, se procederá a explicar lo solución final propuesta. En la **Figura 6** se muestra un esquema de la solución propuesta:



**Figura 6:** Esquema de la solución propuesta

Los datos provenientes de distintas fuentes pasarán por un proceso **ETL** previamente a ser almacenados en la base de datos unificada. Los procesos ETL [22] son parte esencial en el manejo de datos y constan de tres fases definidas: Extraer, Transformar y Cargar. Al realizarse estos tres procesos sobre los datos originales, nos aseguraremos de que los datos están preparados para ser almacenados en la base de datos del SI.

La base de datos del SI estará basada en una **arquitectura no relacional**. Se ha decantado por esta opción debido al gran volumen de datos que generan las espiras y necesitan almacenarse. Como hemos comentado anteriormente (*Tabla 2*) las bases de datos no relacionales disponen de un menor consumo de recursos y de un mejor algoritmo de optimización en sus operaciones, por lo que supone la opción que cumple con los requisitos deseados.

Dicha base de datos será desarrollada e implementada mediante **MongoDB** [23]. MongoDB es un sistema de base de datos no SQL orientado a documentos, es decir, que en lugar de guardar los datos en tablas (bases de datos relacionales) u otras arquitecturas, lo hace mediante documentos BSON (similares en formato a JSON). MongoDB, al igual que otras bases de datos no relacionales, ofrece una gran escalabilidad y flexibilidad y es ampliamente utilizado.

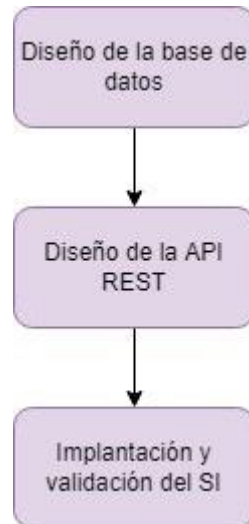
Por otro lado, para el diseño de la API, se ha optado por utilizar la arquitectura REST, empleando así una **API Restful**. Las API Restful usan menor cantidad de recursos y son fáciles de implementar (*Tabla 3*). Además, permiten enviar los mensajes en una gran cantidad de formatos y posibilitan operar con dispositivos de diversos sistemas operativos. Entre todos los lenguajes de programación disponibles para la creación de APIs, la API Restful será desarrollada mediante **NestJS** [24]. NestJS es un framework progresivo de Node.js, el cual destaca por su arquitectura escalable y su gran versatilidad. También hay que destacar su gran capacidad para trabajar con bases de datos basadas en MongoDB debido a su módulo **Mongoose** [25].

Se ha hecho hincapié en lo importante que es la seguridad del acceso a la base de datos para el correcto funcionamiento del sistema. Es cierto que las API REST pueden presentar problemas en la seguridad respecto a las API SOAP, pero en la actualidad existen mecanismos y buenas formas que permiten paliar este problema. Uno de esos mecanismos es **CORS** [26], el cual consiste en el uso de cabeceras HTTP adicionales para obtener los permisos necesarios permitiendo acceder a los recursos deseados.

### 3.3. Plan de trabajo

---

Para desarrollar con éxito la solución planteada, se seguirá un plan de trabajo dividido en diversas fases. En la **Figura 7**, podremos apreciar la planificación realizada, acompañado posteriormente de una breve explicación de cada fase:



*Figura 7: Plan de trabajo*

- **Diseño de la base de datos:** En esta fase se tratarán los datos recibidos de distintas fuentes aplicándoles un proceso ETL. Posteriormente se mostrará la creación de la base de datos a desarrollar y la unificación de todos los datos en ella.
- **Diseño de la API REST:** Se muestra el proceso de desarrollo de la API REST, tanto el diseño de sus *EntryPoints* como el de sus clases.
- **Implantación y validación del SI:** Finalmente para verificar el correcto funcionamiento del sistema, se implementará y se validará mediante el uso de un ejemplo.

## 4. Diseño e implementación

---

Tras identificar los requisitos necesarios de las herramientas a implementar, se procederá a desarrollar la solución al problema planteada. Para ello, teniendo en cuenta el plan de trabajo propuesto, se comenzará a diseñar la base de datos, posteriormente se desarrollará la API REST encargada de acceder a la base de datos y finalmente se realizará una serie de pruebas que validen el correcto funcionamiento de ambas herramientas.

Debido a la gran cantidad de datos almacenados hasta la fecha por el sistema de medición, tanto para el diseño de la base de datos como para el desarrollo de la API, se utilizarán los datos generados por las espiras en el mes de Junio de 2022 como guía en su diseño y desarrollo. Una vez comprobado el correcto funcionamiento de la solución y el cumplimiento de los requisitos solicitados, se realizará un posterior escalado con el resto de los datos generados.

### 4.1. Tecnologías empleadas

---

La base de datos, la cual almacenará la información generada por las espiras, será diseñada en base a una arquitectura datos NoSQL mediante el uso de MongoDB. Para la introducción de los datos almacenados en los ficheros en la base de datos y su posterior visualización, se empleará la herramienta **MongoDB Compass** [27]. MongoDB Compass es una herramienta gratuita e interactiva la cual nos permite consultar, optimizar y analizar los datos de una base de datos MongoDB.

Como hemos comentado con anterioridad, para el desarrollo de la API REST se empleará el lenguaje de programación NestJs. Dicho lenguaje posee una gran serie de librerías a su disposición, las cuales nos aportan multitud de funcionalidades dependiendo de la situación a la que nos enfrentemos. Durante el desarrollo de la solución se emplearán diversas librerías, pero cabe recalcar el uso de las librerías Mongoose y Swagger.

**Mongoose** es una librería para Node.js que permite escribir consultas para una base de datos de MongoDB de forma sencilla. Incluye una conversión de tipos incorporada, validación, creación de consultas y más. El empleo de la librería mongoose, nos permitirá conectar la API REST con la base de datos en MongoDB.

Por otro lado, **Swagger** [28] también es una librería para Node.js la cual permite establecer reglas, especificaciones y herramientas para documentar la API. Swagger permitirá crear una interfaz intuitiva de nuestra API REST, desde la cual podremos acceder a los distintos EntryPoints o métodos que dispone.

Para visualizar los ficheros originales generados por las espiras de medición, se empleará la herramienta gratuita de visualización de documentos, **NotePad++** [29], la cual dispone de diversas funcionalidades que permitirán la realización de los procesos ETL y preparación de los datos.

Finalmente, para la realización de pruebas API se empleará la herramienta **Postman** [30]. Postman es una herramienta gratuita la cual consiste en un servidor HTTP que posibilita probar el funcionamiento de las peticiones realizadas a la API mediante una interfaz gráfica de usuario.

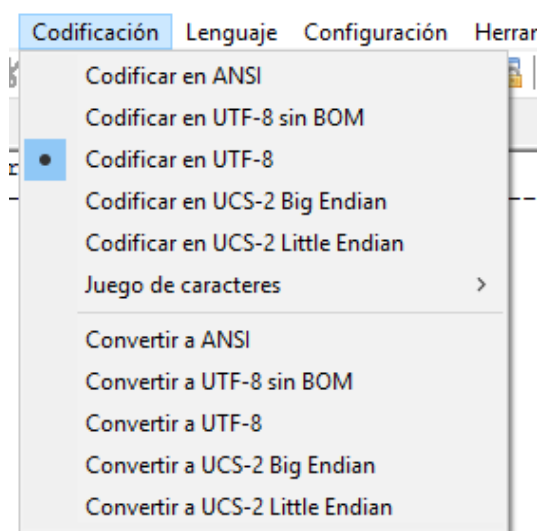


## 4.2. Diseño de la base de datos

El primer paso que realizar será la preparación de los datos para su correcta inserción en la base de datos. Para ello, se harán uso de los procesos ETL. Los procesos de extracción, transformación y carga también conocidos como ETL, son utilizados por las empresas y las organizaciones para preparar los datos procedentes de varias fuentes en una base o almacén de datos. Dichos procedimientos serán realizados con el objetivo de aseguraremos de que los datos están preparados para ser almacenados en la base de datos del SI.

Actualmente, los ficheros generados por las espiras se encuentran codificados en formato ANSI [31] el cual es una modificación de ASCII, un estándar para la codificación de los caracteres en Windows. Dicha codificación no es compatible con una base de datos MongoDB y puede generar errores a la hora de importar los datos en la base, así como modificación del contenido original. Es por eso por lo que se precisa cambiar la codificación de estos archivos, aplicando un mecanismo de transformación. La codificación más utilizada a la hora de tratar con bases de datos implementadas con MongoDB el UTF-8 [32], un estándar de HTML y compatible con MongoDB.

Para realizar estos procesos utilizaremos la herramienta Notepad ++ la cual dispone de metodologías transformar los datos procedentes de las espiras de medición. Para ello, usaremos la función de “Convertir codificación” de Notepad++ y seleccionaremos como nueva codificación UTF-8. Una vez aplicada la transformación, se observa en la **Figura 9** el correcto uso de los caracteres en los datos.



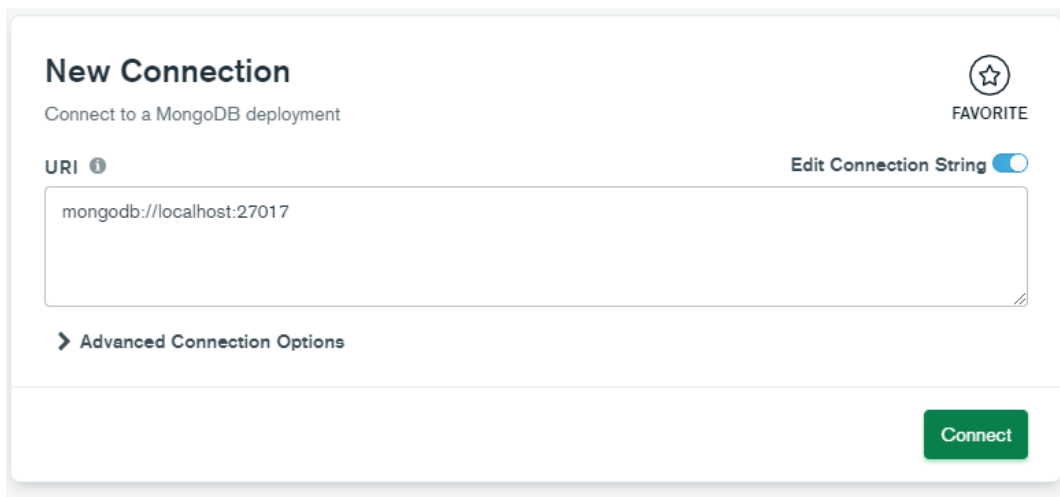
**Figura 8:** Opciones de codificación en NotePad++

```
3 101;P101;PINTOR SOROLLA, N° 1 (de Poeta Querol a Universidad);Pl. Alfonso el Magnánimo;1;2022-06-01 00:05:00.000;174;100;0;100;9;25
4 101;P101;PINTOR SOROLLA, N° 1 (de Poeta Querol a Universidad);Pl. Alfonso el Magnánimo;1;2022-06-01 00:10:00.000;141;100;0;100;8;25
5 101;P101;PINTOR SOROLLA, N° 1 (de Poeta Querol a Universidad);Pl. Alfonso el Magnánimo;1;2022-06-01 00:15:00.000;298;100;0;100;26;25
6 101;P101;PINTOR SOROLLA, N° 1 (de Poeta Querol a Universidad);Pl. Alfonso el Magnánimo;1;2022-06-01 00:20:00.000;120;100;0;100;9;25
7 101;P101;PINTOR SOROLLA, N° 1 (de Poeta Querol a Universidad);Pl. Alfonso el Magnánimo;1;2022-06-01 00:25:00.000;90;100;0;100;7;25
8 101;P101;PINTOR SOROLLA, N° 1 (de Poeta Querol a Universidad);Pl. Alfonso el Magnánimo;1;2022-06-01 00:30:00.000;87;100;0;100;4;25
```

**Figura 9:** Datos preparados para almacenar

Una vez preparados los datos, nos dispondremos a crear la base de datos en MongoDB y cargar la información en ella. Utilizaremos la herramienta gratuita MongoDB Compass, la cual nos permitirá crear la base de datos y realizar consultas sobre ella de forma sencilla.

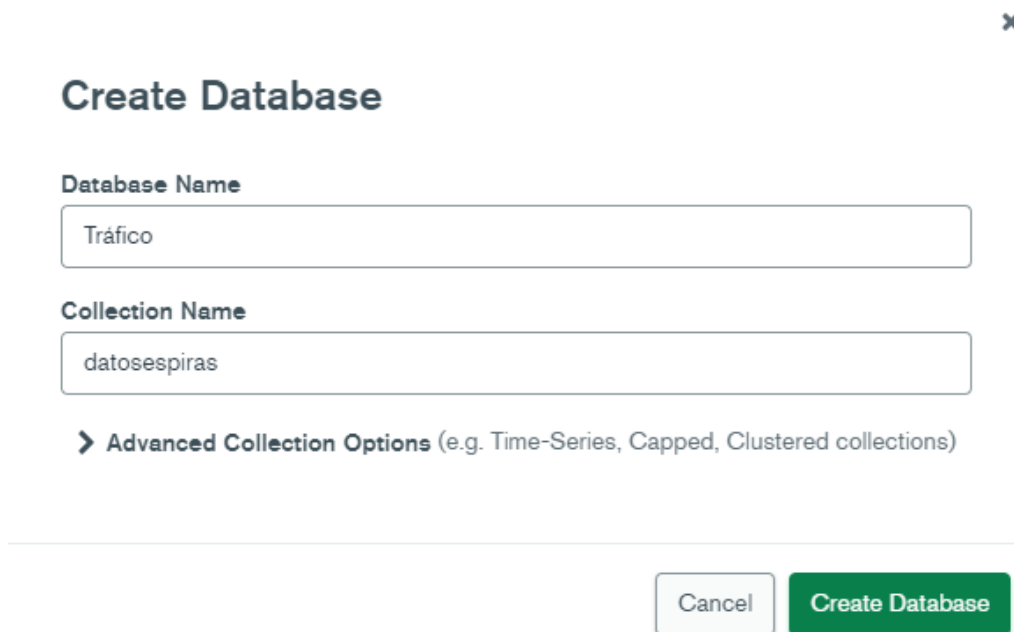
Ya instalada la herramienta, como observamos en la Figura 10, debemos de indicar en que puerto de nuestro ordenador deseamos alojar la base de datos en MongoDB. Por defecto, el puerto seleccionado es el 27017, el cual también usaremos a lo largo del desarrollo de la solución. A continuación, procedemos a conectarnos al puerto mediante la opción “Connect”.



The screenshot shows a 'New Connection' dialog box. At the top, it says 'Connect to a MongoDB deployment'. There is a 'FAVORITE' icon in the top right. Below that, there is a 'URI' field with a help icon and an 'Edit Connection String' toggle switch. The URI field contains 'mongodb://localhost:27017'. Below the URI field is a link for 'Advanced Connection Options'. At the bottom right, there is a green 'Connect' button.

**Figura 10:** Conexión a la base de datos

Después de conectarnos al puerto, nos dirigimos a la sección “Databases” y nos mostrará las bases de datos que se están alojando en la actualidad. Puesto que no disponemos de ninguna, seleccionaremos la opción “Create Database”, generando el siguiente formulario:



The screenshot shows a 'Create Database' form. At the top right, there is a close button (X). The form has two input fields: 'Database Name' with the value 'Tráfico' and 'Collection Name' with the value 'datosespiras'. Below these fields is a link for 'Advanced Collection Options (e.g. Time-Series, Capped, Clustered collections)'. At the bottom, there are two buttons: 'Cancel' and 'Create Database'.

**Figura 11:** Formulario de creación de una base de datos en MongoDB Compass

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

Como se ha comentado en el análisis del problema, MongoDB es una base de datos orientada a documentos, lo que significa que, en lugar de guardar los datos en registros, son almacenados en documentos BSON, una extensión de JSON. En vez de utilizar registros como estructura de la base de datos, MongoDB trabaja con las llamadas colecciones, las cuales consisten en un documento que almacenará cualquier tipo de objeto o dato. En el formulario generado en la Figura 11, introducimos el nombre deseado a nuestra base de datos y a la colección que se va a almacenar.

En este momento la base de datos ya está creada, pero carece de datos almacenados. A continuación, procederemos a cargar en la base de datos los datos previamente tratados y preparados. Para ello nos dirigiremos a la opción “Add Data” en nuestra base de datos e indicaremos el origen del fichero. Indicamos que se trata de un archivo con la extensión .csv y que tiene los distintos campos separados por punto y coma. También, como proceso de transformación, indicamos que elimine o ignore las filas que tengan un campo vacío, mejorando significativamente la calidad de la información que se extraerá de ellos. Posteriormente, indicamos el tipo de dato que alberga cada una de las columnas generadas, en base al formato del documento original (Tabla 1).

Import To Collection Tráfico.datosespiras

Select File

2022-06.csv

Select Input File Type

JSON CSV

Options

Select delimiter SEMICOLON

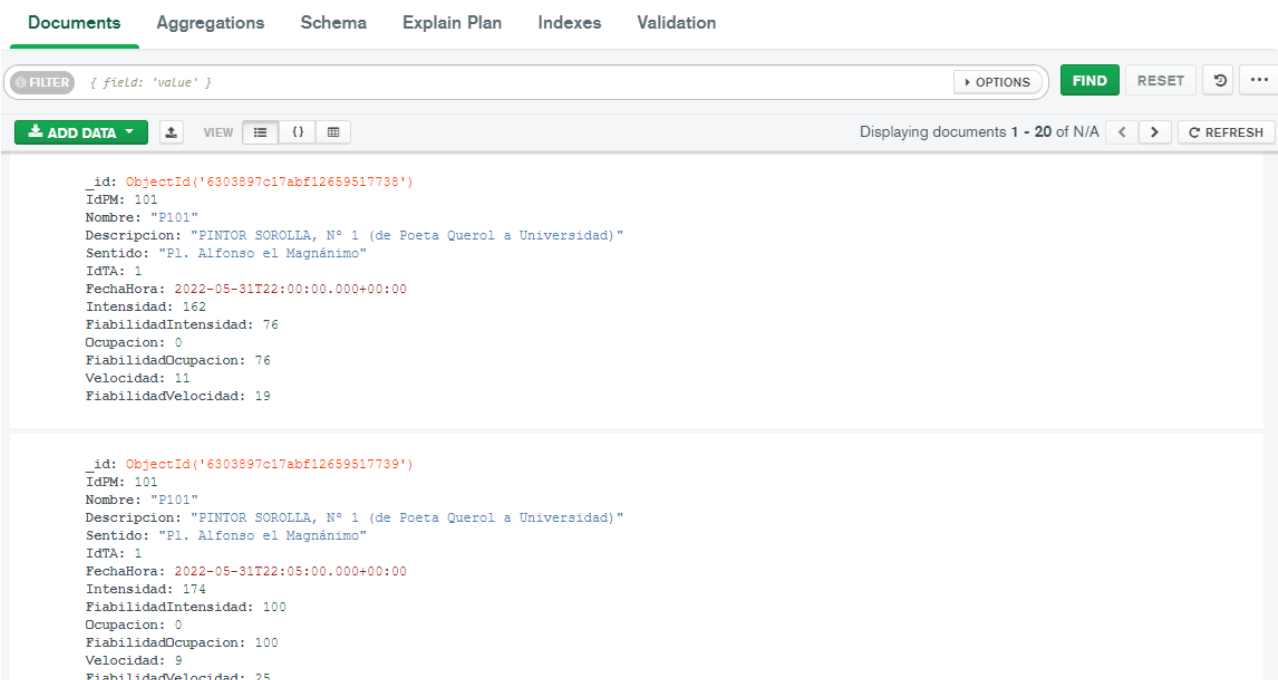
Ignore empty strings

Stop on errors

tensidad	Ocupacion	FiabilidadOcupacion	Velocidad	FiabilidadVelocidad
0	76	11	19	
0	100	9	25	
0	100	8	25	
0	100	26	25	
0	100	9	25	
0	100	7	25	
0	100	4	25	
0	100	14	25	
0	100	3	25	
0	100	7	25	

Importing documents... Stop 3117000 / ~8711281.400997223

Figura 12: Importación del fichero en la base de datos



The screenshot shows a MongoDB web interface. At the top, there are navigation tabs: Documents, Aggregations, Schema, Explain Plan, Indexes, and Validation. Below the tabs is a filter bar with a placeholder '{ field: 'value' }' and buttons for OPTIONS, FIND, RESET, and a refresh icon. Below the filter bar is a toolbar with ADD DATA, VIEW, and other icons. The main area displays a list of documents, with the first two visible. Each document is a JSON object with the following fields: \_id, IdPM, Nombre, Descripcion, Sentido, IdTA, FechaHora, Intensidad, FiabilidadIntensidad, Ocupacion, FiabilidadOcupacion, Velocidad, and FiabilidadVelocidad. The first document has a FechaHora of 2022-05-31T22:00:00.000+00:00 and an Intensidad of 162. The second document has a FechaHora of 2022-05-31T22:05:00.000+00:00 and an Intensidad of 174.

Figura 13: Base de datos creada

## 4.3. Diseño API

Una vez creada la base de datos que almacenará los datos generados por las espiras y posteriormente introducido los datos del fichero de guía, nos disponemos a diseñar la API REST encargada de acceder a ellos de forma segura y amigable con el usuario.

A la hora de diseñarla, previamente debemos identificar y definir los diferentes puntos de acceso o EntryPoints que va a disponer la API.

### 4.3.1. Definición de EntryPoints

Un EntryPoint o punto de acceso es una URL la cual describe la función o método que la API puede realizar. Como norma general, por cada funcionalidad de la API, ha de a ver definido un EntryPoint que aporte información acerca del método que ejecuta e indique la URL asignada a él.

En las APIs REST, existe un estándar aceptado por la comunidad a la hora de definir los EntryPoints de la API. Dicho modelo es conocido como HATEOAS (*Hypermedia As The Engine Of Application State*) el cual indica que la API ha de poseer una interfaz de servicio universal. Esto permite que el usuario que accede a nuestra API pueda navegar a través de sus EntryPoints y métodos simplemente conociendo las URI y sin la necesidad de comprender la estructura interna de la API.

Ya aclarados los conceptos de EntryPoint y de HATEOAS, nos disponemos a definir los distintos EntryPoints que dispondrá nuestra API REST, los cuales se ven reflejados en la **Tabla 4**:

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

**Tabla 4:** Definición de los EntryPoints de la API a desarrollar

URI	Protocolo	Verbo	Parámetro de entrada	Respuesta	Formato respuesta
1_Locahost:3000/datosEspiras/tramos/readALL	HTTP	GET	-	Mensaje + Lista	JSON
2_Locahost:3000/datosEspiras/espiras/readALL	HTTP	GET	-	Mensaje + Lista	JSON
3_Locahost:3000/datosEspiras /tramos/espiras/readAll/?idTA	HTTP	GET	idTA	Mensaje + Lista	JSON
4_Locahost:3000/datosEspiras /tramos/intensidad/?idTA	HTTP	POST	idTA, fecha 1, fecha 2	Mensaje + Valor	JSON
5_Locahost:3000/datos_espiras /tramos/ocupación/?idTA	HTTP	POST	idTA, fecha1, fecha2	Mensaje + Valor	JSON
6_Locahost:3000/datosEspiras /espiras/intensidad/?idPM	HTTP	POST	idPM, fecha1, fecha2	Mensaje + Valor	JSON
7_Locahost:3000/datosEspiras /espiras/ocupacion/?idPM	HTTP	POST	IdPM, fecha1, fecha2	Mensaje + Valor	JSON
8_Locahost:3000/datos_espiras /tramos/delete/?idTA	HTTP	DELETE	idTA	Mensaje	JSON
9_Locahost:3000/datos_espiras /espiras/delete/?idPM	HTTP	DELETE	idPM	Mensaje	JSON

Dentro de los datos que generan las espiras de medición, encontramos 2 objetos o entes de interés para posteriores análisis. Estos son los distintos puntos de media o espiras y los tramos que forman el sistema. Para futuros análisis de la contaminación en base al tráfico generado, se necesita saber la ocupación y la intensidad de cada tramo o cada punto de medición a lo largo del tiempo. Es por ello por lo que se centrará el diseño de las API en satisfacer futuras consultas relacionadas con estos entes.

De los tramos se desea saber cuáles están siendo monitorizados por el sistema de información. Para ello definimos un método el cual devuelva un listado con los nombres de los tramos registrados en el sistema (**EntryPoint 1**). Del mismo modo, se desea saber el número de espiras o puntos de medición registradas en el sistema, por lo que determinamos un método que devuelva un listado con los nombres de los puntos de medición almacenados en la base de datos (**EntryPoint 2**). Por otro lado, también es de interés saber qué puntos de medida posee cierto tramo, por eso se define un EntryPoint que dado un id de un tramo, devuelva el conjunto de tramos asociado a él (**EntryPoint 3**).

Como hemos comentado anteriormente, para realizar un futuro análisis de los datos generados por el sistema, es necesario conocer la intensidad y la ocupación media de los tramos. Es por eso por lo que definiremos dos puntos de acceso (**EntryPoints 4 y 5**) los cuales devuelven la intensidad o la ocupación media entre dos fechas que se solicitarán al usuario. De los puntos de medidas, también se desea conocer la intensidad y la ocupación de los datos que generan, en consecuencia, definimos dos puntos de acceso (**EntryPoints 6 y 7**) los cuales permitirán conocer la ocupación o intensidad media de los datos producidos por el punto de medida entre dos fechas seleccionada.

Finalmente, en caso de carencia de interés de un tramo o de avería de un punto de medición, debemos habilitar la posibilidad de eliminar cierto tramo o espira del sistema (**EntryPoints 8 y 9**).

## 4.3.2. Diseño clases

Ya habiendo definido los distintos métodos o EntryPoints que va a disponer nuestra API REST, procedemos a la creación del proyecto y al diseño de las diferentes clases que lo compondrán. El primer paso consistirá en la instalación del lenguaje NestJS en la carpeta que almacenará el proyecto de ahora en adelante. Para ello ,en nuestra terminal, utilizaremos el comando mostrado en la **Figura 14**:

```
C:\Proyecto TFG
λ npm install -g @nestjs/cli

added 10 packages, removed 14 packages, changed 236 packages, and audited 247 packages in 47s

37 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.5.0 -> 8.12.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.12.1
npm notice Run npm install -g npm@8.12.1 to update!
npm notice

C:\Proyecto TFG
λ |
```

*Figura 14: Instalación NestJS*

Una vez instalado NestJS como lenguaje de desarrollo de la API, creamos el proyecto que albergara el conjunto de métodos y carpetas de la API REST a desarrollar .Usaremos el siguiente comando en nuestra terminal y posteriormente nos preguntarán que manejador de paquetes deseamos emplear (**Figura 15**). En nuestro caso utilizaremos el manejador de paquetes npm debido a su gran popularidad y su facilidad de uso.

```
C:\Proyecto TFG
λ nest new API Tráfico
  We will scaffold your app in a few seconds..

CREATE api_tráfico/.eslintrc.js (665 bytes)
CREATE api_tráfico/.prettierrc (51 bytes)
CREATE api_tráfico/nest-cli.json (118 bytes)
CREATE api_tráfico/package.json (1997 bytes)
CREATE api_tráfico/README.md (3340 bytes)
CREATE api_tráfico/tsconfig.build.json (97 bytes)
CREATE api_tráfico/tsconfig.json (546 bytes)
CREATE api_tráfico/src/app.controller.spec.ts (617 bytes)
CREATE api_tráfico/src/app.controller.ts (274 bytes)
CREATE api_tráfico/src/app.module.ts (249 bytes)
CREATE api_tráfico/src/app.service.ts (142 bytes)
CREATE api_tráfico/src/main.ts (208 bytes)
CREATE api_tráfico/test/app.e2e-spec.ts (630 bytes)
CREATE api_tráfico/test/jest-e2e.json (183 bytes)

? Which package manager would you ♥ to use? (Use arrow keys)
> npm
  yarn
  pnpm
```

*Figura 15: Creación del proyecto*

A continuación, nos dispondremos a instalar las librerías Mongoose y Swagger, las cuales utilizaremos durante la realización del diseño de la API REST para las operaciones relacionadas con la base de datos desarrollada.

```
C:\Proyecto TFG\api_tráfico(master)
λ npm i mongoose@latest

up to date, audited 743 packages in 3s

82 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Proyecto TFG\api_tráfico(master)
λ npm i @nestjs/mongoose@latest

up to date, audited 743 packages in 14s

82 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Proyecto TFG\api_tráfico(master)
λ npm i @types/mongoose

up to date, audited 743 packages in 4s

82 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

*Figura 16: Instalación de los paquetes y librerías de mongoose*

```
C:\Proyecto TFG\api_tráfico(master)
λ npm i @nestjs/swagger@5.2.1

added 2 packages, and audited 745 packages in 14s

82 packages are looking for funding
  run `npm fund` for details
```

*Figura 17: Instalación librería Swagger*

El proyecto constará de dos módulos a diseñar. Un módulo llamado database el cual se encargará de conectar la API REST con la base de datos MongoDB previamente creada. El otro módulo, DatosEspiras, definirá los diferentes métodos y servicios que deseamos que contenga nuestra API. Seguidamente se creará la estructura y el sistema de clases que albergaran las funcionalidades de la API REST.

```
C:\Proyecto TFG
λ nest g mo Database
CREATE src/database/database.module.ts (85 bytes)

C:\Proyecto TFG
λ nest g pr database/databaseProviders
CREATE src/database/database-providers.spec.ts (491 bytes)
CREATE src/database/database-providers.ts (94 bytes)
UPDATE src/database/database.module.ts (177 bytes)

C:\Proyecto TFG
```

*Figura 18: Creación módulo database*

```

C:\Proyecto TFG\api_tráfico\src(master)
λ nest g mo DatosEspiras
CREATE datos-espigas/datos-espigas.module.ts (89 bytes)
UPDATE app.module.ts (342 bytes)

```

*Figura 19: Creación módulo DatosEspiras*

Primero trataremos de completar el módulo database. Como hemos comentado anteriormente, el módulo database será el encargado de conectar la base de datos en MongoDB con la aplicación. Para cumplir dicho objetivo, definimos en la clase *myConfig.ts* (Figura 20) el nombre de la base de datos que deseamos conectarnos y su servidor. Después, conectamos la API a la base de datos en la clase *database-providers.ts* (Figura 21) mediante el uso del método *connect*, el cual, dado una dirección de la base de datos definida previamente en la clase *myConfig.ts*, establece una conexión con ella. Finalmente, indicamos como proveedor de datos, en los manejadores generales de la API, la conexión creada (Figura 22).

```

C: > Proyecto TFG > TS myConfig.ts > ...
1  export const myConfig = {
2    mongoServer: 'localhost',
3    traficoDatabase: 'Trafico',
4  }

```

*Figura 20: Creación clase myConfig.ts*

```

1  import * as mongoose from 'mongoose';
2  import { myConfig } from '../myConfig';
3
4  export const databaseProviders = [
5    {
6      provide: 'DATABASE_CONNECTION',
7      useFactory: (): Promise<typeof mongoose> =>
8        mongoose.connect(`mongodb://${myConfig.mongoServer}/${myConfig.traficoDatabase}`),
9    }
10 ];
11

```

*Figura 21: Creación clase database-providers*

```

src > database > TS database.module.ts > ...
1  import { Module } from '@nestjs/common';
2  import { DatabaseProviders } from './database-providers';
3
4  @Module({
5    providers: [...DatabaseProviders],
6    exports: [...DatabaseProviders]
7  })
8  export class DatabaseModule {}
9

```

*Figura 22: Clase database.module.ts*



## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

Ya establecida la conexión mediante el módulo database, nos centraremos en el módulo DatosEspiras, el cual contiene los distintos métodos o EntryPoints de nuestra API. El primer paso será definir cada una de las clases que necesitaremos para cumplir con todos los EntryPoints establecidos. A continuación, detallamos el propósito y contenido de cada una:

- **Controlador:** El controlador es la clase que contiene la declaración de los EntryPoints y será desde esta clase donde se llamen a los métodos y servicios de cada uno de los EntryPoints definidos. Como podemos ver en la , cada uno de los EntryPoints detalla el método HTTP que utiliza ( GET, POST, DELETE) y ruta o URL que dirige a dicho punto de acceso.

```
src > datos-espiras > api > datos-espiras > TS datos-espiras.controller.ts > DatosEspirasController
1  import { Body, Controller, Delete, Get, Param, Post } from '@nestjsjs/common/decorators';
2  import { DatosEspirasService } from 'src/datos-espiras/services/datos-espiras/datos-espiras.service';
3  import { DatosEspirasDto } from 'src/datos-espiras/dto/datos-espiras-dto';
4
5
6  @Controller('datos-espiras')
7  export class DatosEspirasController {
8      constructor(private readonly DatosEspirasService: DatosEspirasService) {}
9
10
11     @Get('/tramos/readAll') readAllTramos() {
12         return this.DatosEspirasService.findTramos();
13     }
14
15
16     @Get('/espiras/readAll') readAllEspiras() {
17         return this.DatosEspirasService.findEspiras();
18     }
19
20     @Get('/tramos/espiras/readALL/:idTA')
21     readAllEspirasInTramo(@Param('idTA') idTA: number) {
22         return this.DatosEspirasService.findAllEspirasInTramo(idTA).then(r => {
23             return(r);
24         } )
25     }
26
27     @Post('/tramos/ocupacion/:idTA')
28     ocupacionTramos(@Param('idTA') idTA: number, @Body() DTO: DatosEspirasDto){
29         return this.DatosEspirasService.ocupacionTramo(idTA,DTO).then(r => {
30             return(r);
31         } )
32     }
33
34     @Post('/tramos/intensidad/:idTA')
35     intensidadTramos(@Param('idTA') idTA: number, @Body() DTO: DatosEspirasDto){
36         return this.DatosEspirasService.intensidadTramo(idTA,DTO).then(r => {
37             return(r);
38         } )
39     }
}
```

Figura 23: Definición EntryPoints en la clase controladora (Parte 1)

```

@Post('/espiras/ocupacion/:idPM')
ocupacionEspiras(@Param('idPM') idPM: number, @Body() DTO: DatosEspirasDto){
  return this.DatosEspirasService.ocupacionEspira(idPM,DTO).then(r => {
    return(r);
  } )
}

@Post('/espiras/intensidad/:idPM')
intensidadEspiras(@Param('idPM') idPM: number, @Body() DTO: DatosEspirasDto){
  return this.DatosEspirasService.intensidadEspira(idPM,DTO).then(r => {
    return(r);
  } )
}

@Delete('/tramos/delete/:idTA')
deleteTramo(@Param('idTA') idTA: number) {
  return this.DatosEspirasService.deleteTramo(idTA).then(r => {
    return (r);
  });
}

@Delete('/espiras/delete/:idPM')
deleteEspira(@Param('idPM') idPM: number) {
  return this.DatosEspirasService.deleteEspira(idPM).then(r => {
    return (r);
  });
}

```

Figura 24: Definición EtryPoints en la clase controladora (Parte 2)

- **DTO:** Del inglés *Data Transfer Object*, define el formato de los servicios que requieran el uso del método HTTP POST. El método POST permite enviar datos al servidor por medio de un cuerpo (body) o formulario. En nuestra API, necesitamos enviar las fechas deseadas en ciertos métodos, por lo que deberemos definir el formato del objeto que queremos enviar. Como vemos en la , definimos un formulario que solicite una fecha de inicio y una fecha final, ambas en formato String.

```

src > datos-espiras > dto > TS datos-espiras-dto.ts > ...
1  import { Date } from "mongoose";
2
3  export class DatosEspirasDto {
4
5      FechaI: string;
6      FechaF: string;
7
8  }

```

Figura 25: Declaración del formato del DTO

- **Interfaces:** Definimos dos interfaces, las cuales utilizaremos en diversos métodos de nuestra API y en los mensajes de respuesta. La interfaz *i-datos-espiras.interface.ts* declara la estructura que posee cada uno de los objetos almacenados en la base de datos. Como hemos comentado en el análisis del problema, las filas o objetos disponen de una serie de campos, es por ello por lo que en esta clase definiremos cada uno de ellos y su correspondiente tipo (**Figura 26**).

```
src > datos-espiras > interfaces > TS i-datos-espiras.interface.ts > ...
 1  import { Document } from 'mongoose';
 2
 3  export interface IDatosEspiras extends Document {
 4
 5      readonly IdPM: number;
 6      readonly Nombre: string;
 7      readonly Descripcion: string;
 8      readonly Sentido: string;
 9      readonly IdTA: number;
10     readonly FechaHora: Date;
11     readonly Intensidad: number;
12     readonly FiabilidadIntensidad: number;
13     readonly Ocupacion: number;
14     readonly FiabilidadOcupacion: number;
15     readonly Velocidad: number;
16     readonly FiabilidadVelocidad: number;
17 }
```

*Figura 26: Interfaz i-datos-espiras-interface.ts*

Por otro lado, la interfaz *i-return.interface.ts* define el formato del mensaje que vamos a enviar al usuario tras realizar un servicio de nuestra API (**Figura 27**).

```
src > datos-espiras > interfaces > TS i-return.interface.ts > IReturn
 1  export interface IReturn {
 2
 3      readonly msg: string;
 4      readonly status: boolean;
 5      readonly data: any;
 6      readonly code: string;
 7
 8  }
```

*Figura 27: Interfaz i-return-inteface.ts*

- **Schemas:** Un esquema es una interfaz especial la cual permite que dos partes de software o servicios realicen transacciones mediante un sistema de solicitud y respuesta. En nuestro caso, el Schema facilitará el intercambio de información entre la API y la base de datos. En la **Figura 28** definimos la clase encargada de mandar la información a la base de datos MongoDB.

```
src > datos-espiras > schemas > TS datos-espiras-schema.ts > ...
1  import * as mongoose from 'mongoose'
2
3  export const DatosEspirasSchema = new mongoose.Schema({
4
5      IdPM: Number,
6      Nombre: String,
7      Descripcion: String,
8      Sentido: String,
9      IdTA: Number,
10     FechaHora: Date,
11     Intensidad: Number,
12     FiabilidadIntensidad: Number,
13     Ocupacion: Number,
14     FiabilidadOcupacion: Number,
15     Velocidad: Number,
16     FiabilidadVelocidad: Number,
17
18 });
```

Figura 28: Clase datos-espiras-schema.ts

- **Providers:** Al igual que en el módulo database, necesitaremos enlazar el módulo DatosEspiras con la conexión previamente realizada. Para ello, usaremos el Schema previamente definido como mecanismo de intercambio de información.

```
src > datos-espiras > providers > TS datos-espiras.ts > ...
1  import { Connection } from 'mongoose';
2  import { DatosEspirasSchema } from '../schemas/datos-espiras-schema';
3
4  export const DatosEspirasProviders = [
5
6      {
7          provide: 'DATOSESPIRAS_MODEL',
8          useFactory: (connection: Connection) => connection.model('DatosEspiras', DatosEspirasSchema),
9          inject: ['DATABASE_CONNECTION']
10     },
11 ];
```

Figura 29: Clase datos-espiras.ts

- **Servicios:** En esta clase definiremos los métodos asignados a los EntryPoints, es decir, las distintas funciones que realiza la API REST. Posteriormente estos métodos serán invocados desde el controlador de la API. En la **Figura 30**, observamos el método *findTramos()* el cual al igual que el método *findEspiras()*, devuelve una lista con todos los tramos o espiras registrados en la base de datos.

```
// Métodos de listar espiras y tramos de la base de datos

async findTramos(): Promise<IReturn> {
  const myPromise = new Promise<IReturn>(async (resolve, reject) => {

    const datos = await this.datosEspirasModel.find();
    const tramos = [];

    datos.forEach(element => { tramos.push(element.IdTA)});

    let unique = [...new Set(tramos)];

    resolve({ msg: 'Lista con los tramos (IdTA) almacenados en la base de datos:', status: true, data: unique, code: '80xx' });});
  return myPromise;
}
```

*Figura 30: Método findTramos()*

Por otro lado, necesitamos un método el cual cumpla con el EntryPoint de mostrar cada una de las espiras asignadas a un determinado tramo. Para ello definimos el método *findAllEspirasInTramo()* el cual, dado el identificador de tramo, devuelve el nombre de cada uno de los puntos de medición que posee:

```
async findAllEspirasInTramo(IdTramo: Number): Promise<IReturn> {
  const tramos = await this.datosEspirasModel.find({IdTA: IdTramo});
  const myPromise = new Promise<IReturn>( (resolve, reject) => {
    if (!tramos.length) {
      resolve({
        msg: 'El tramo indicado no existe',
        status: false,
        data: undefined,
        code: '80xx',
      });
    } else {
      const espiras = [];

      tramos.forEach(element => { espiras.push(element.Nombre)});
      let unique = [...new Set(espiras)];

      resolve({
        msg: 'Las espiras o puntos de medición asignados al tramos indicado son:',
        status: true,
        data: unique,
        code: '80x1',
      });
    }
  });
  return myPromise;
}
```

*Figura 31: Método findAllEspirasInTramo()*

Los EntryPoints 4,5,6 y 7 consisten en devolver la intensidad u ocupación media de los tramos o puntos de medición registrados en el sistema de información. Es por ello por lo que debemos definir una serie de métodos que cumplan con tal propósito. En la **Figura 32** se muestra el método *ocupaciónTramo()* el cual dado el identificador de un tramo y un DTO conteniendo dos fechas, devuelve la ocupación media durante las horas especificadas. Por otro lado, en la **Figura 33** , definimos el método *intensidadTramo()* . Dicho método devuelve la intensidad media de un tramo entre las horas especificadas por el usuario. Finalmente, para el resto de los métodos relacionados los puntos de medición, el desarrollo es similar, sustituyendo el indicador del tramo por el indicador del punto de medición.

```

async ocupacionTramo(IdTramo: Number, DTO: DatosEspirasDto): Promise<IReturn> {
  if (DTO.FechaI === undefined || DTO.FechaI === '' || DTO.FechaI.trim() === '' || DTO.FechaF === undefined || DTO.FechaF === '' || DTO.FechaF.trim() === '') {
    return new Promise<IReturn>((resolve, reject) => {
      resolve({
        msg: 'Introduzca la fechas en el formato correcto. La API considera correctas las fechas con el siguiente formato -> dd-mm-aaaa hh:mm',
        status: false,
        data: undefined,
        code: '80x1',
      });
    });
  }
  const FechaI = new Date(DTO.FechaI);
  const FechaF = new Date(DTO.FechaF);
  if (FechaI.getTime() > Date.now() || FechaF.getTime() > Date.now() || FechaI.getTime() > FechaF.getTime()) {
    return new Promise<IReturn>((resolve, reject) => {
      resolve({
        msg: 'Fechas introducidas de forma incorrecta, por favor revise la fecha especificada',
        status: false,
        data: undefined,
        code: '80x1',
      });
    });
  }
  const tramos = await this.datosEspirasModel.find({IdTA: IdTramo});
  const myPromise = new Promise<IReturn>((resolve, reject) => {
    if (!tramos.length) {
      resolve({
        msg: 'El tramo indicado no existe',
        status: false,
        data: undefined,
        code: '80xx',
      });
    } else {
      const tramosH = tramos.filter(element => ( (element.FechaHora >= FechaI ) && (element.FechaHora <= FechaF)));
      var ocupacion = 0;
      tramosH.forEach(element => { ocupacion = ocupacion + element.Ocupacion; });
      var media = ocupacion/tramosH.length;
      resolve({
        msg: 'La ocupación media del tramo en las horas seleccionadas es: '+ media,
        status: true,
        data: DTO,
        code: '80x1',
      });
    }
  });
  return myPromise;
}

```

**Figura 32:** Método *ocupaciónTramo()*

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

```
async intensidadTramo(IdTramo: Number, DTO: DatosEspirasDto): Promise<IReturn> {
  if (DTO.FechaI === undefined || DTO.FechaI === '' || DTO.FechaI.trim() === '' || DTO.FechaF === undefined || DTO.FechaF === '' || DTO.FechaF.trim() === '') {
    return new Promise<IReturn>((resolve, reject) => {
      resolve({
        msg: 'Introduzca la fechas en el formato correcto. La API considera correctas las fechas con el siguiente formato -> dd-mm-aaaa hh:mm',
        status: false,
        data: undefined,
        code: '80x1',
      });
    });
  }
  const FechaI = new Date(DTO.FechaI);
  const FechaF = new Date(DTO.FechaF);
  if (FechaI.getTime() > Date.now() || FechaF.getTime() > Date.now() || FechaI.getTime() > FechaF.getTime()) {
    return new Promise<IReturn>((resolve, reject) => {
      resolve({
        msg: 'Fechas introducidas de forma incorrecta, por favor revise la fecha especificada',
        status: false,
        data: undefined,
        code: '80x1',
      });
    });
  }
  const tramos = await this.datosEspirasModel.find({IdTA: IdTramo});
  const myPromise = new Promise<IReturn>((resolve, reject) => {
    if (!tramos.length) {
      resolve({
        msg: 'El tramo indicado no existe',
        status: false,
        data: undefined,
        code: '80xx',
      });
    } else {
      const tramosH = tramos.filter(element => ( element.FechaHora >= FechaI ) && (element.FechaHora <= FechaF));
      var intensidad = 0;
      tramosH.forEach(element => {intensidad = intensidad + element.Intensidad; });
      var media = intensidad/tramosH.length;
      resolve({
        msg: 'La intensidad media del tramo en las horas seleccionadas es: '+ media,
        status: true,
        data: DTO,
        code: '80x1',
      });
    }
  });
  return myPromise;
}
```

Figura 33: Método intensidadTramo()

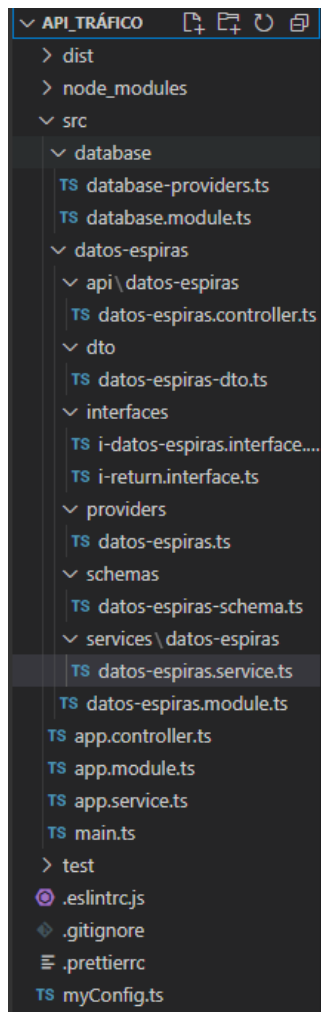
Para finalizar con la definición de los métodos de la clase Services, definimos 2 métodos los cuales dado su respectivo indicar, eliminaran en caso de avería o carencia de interés, la espira o el tramo indicado de la base de datos.

```
// Métodos borrar tramo y borrar espira

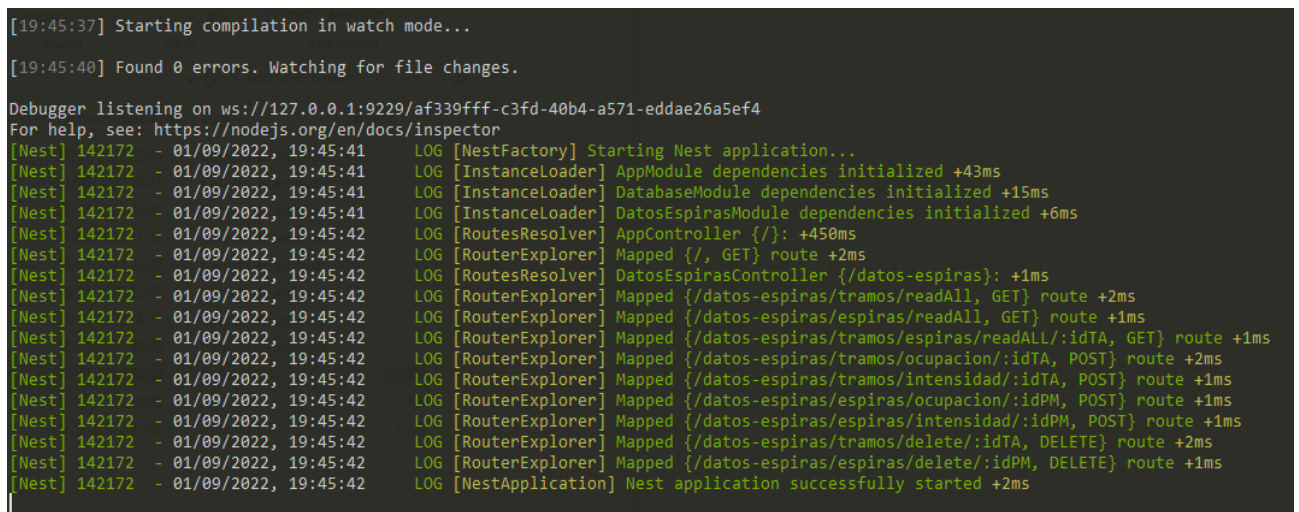
async deleteTramo(IdTramo: number): Promise<IReturn> {
  const existe = await this.datosEspirasModel.find({ IdTA: IdTramo });
  const myPromise = new Promise<IReturn>((resolve, reject) => {
    if (!existe.length) {
      resolve({
        msg: 'El tramo indicado no existe.',
        status: false,
        data: undefined,
        code: '80xx',
      });
    } else {
      this.datosEspirasModel.deleteMany({ IdTA: IdTramo }).exec();
      resolve({
        msg: 'El tramo indicado fue eliminado de la base de datos.',
        status: true,
        data: IdTramo,
        code: '80xx',
      });
    }
  });
  return myPromise;
}
```

Figura 34: Método deleteTramo()

Ya habiendo definido todos los métodos de todas las clases a desarrollar, observamos en la **Figura 35** la estructura final del proyecto. El siguiente paso consistiría en ejecutar el proyecto y verificar el correcto funcionamiento de cada una de las clases.



**Figura 35:** Estructura final del proyecto



**Figura 36:** Ejecución del proyecto



## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

Finalmente, accedemos al sitio web generado por el paquete Swagger. En dicha página web podemos observar los distintos EntryPoints de nuestra API, sus respectivas URL de acceso y los métodos HTTP que utilizan para comunicarse con el servidor. Swagger automáticamente localiza los métodos definidos en la clase controlador de la API.



*Figura 37: Sitio web de la API realizado con Swagger*

# 5. Implantación y validación

Una vez detallado el diseño de la base de datos y desarrollado la API REST, con el objetivo de validar el correcto funcionamiento de la solución planteada y obtener los resultados deseados, se procederá a realizar su implantación y se verificará el correcto funcionamiento de ella.

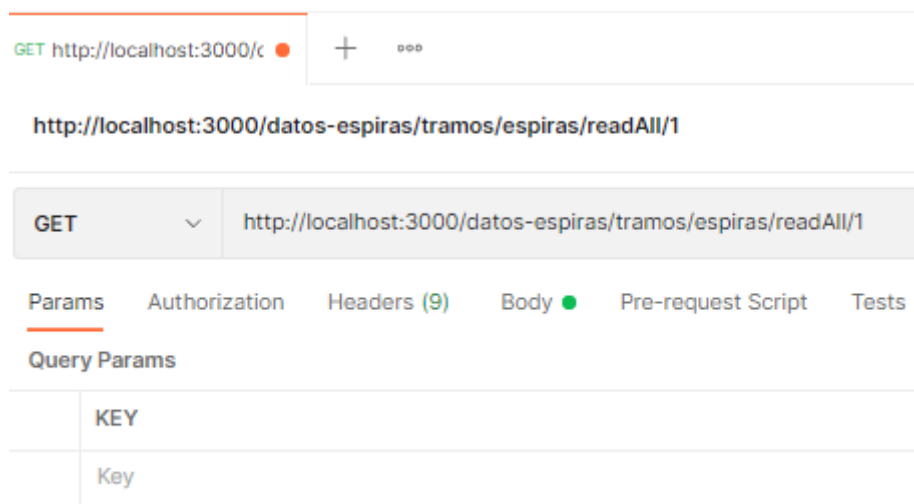
Para ello, se empleará la utilización de la herramienta Postman, la cual nos permite realizar solicitudes al sistema desarrollado. A continuación, se mostrará algunos ejemplos prácticos los cuales muestran el funcionamiento de los métodos definidos y su respuesta.

## 5.1. Ejemplos prácticos

Realizaremos 3 ejemplos prácticos los cuales mostraran el funcionamiento del sistema de información en 3 servicios o EntryPoints distintos. Debemos de tener en cuenta que las pruebas se realizaran a los datos obtenidos por las espiras de medición en el mes de Junio de 2022.

- **Obtener todos los puntos de medición de un determinado tramo:**

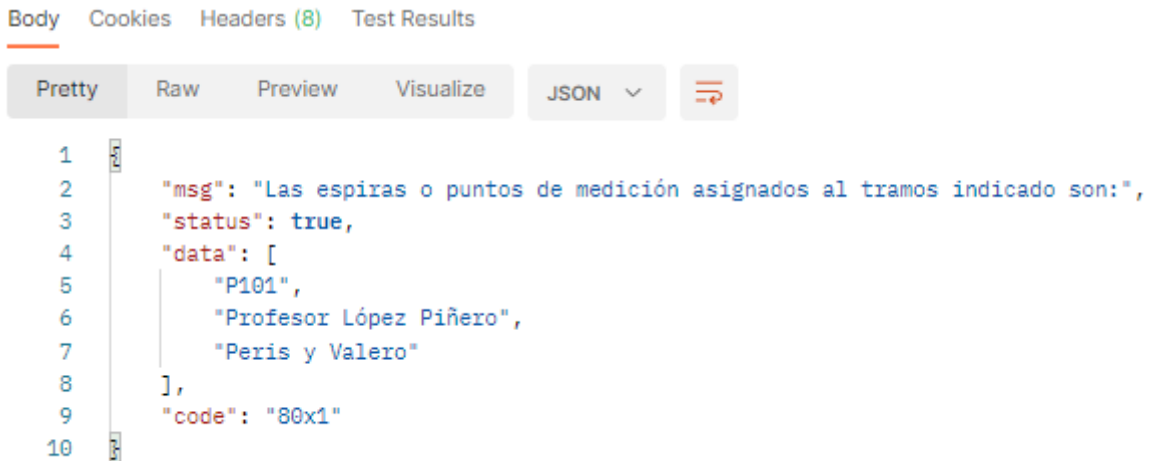
Primero insertamos en Postman la URL asignada a determinado método o EntryPoint e indicamos la acción HTTP a realizar. En el ejemplo, solicitamos todas las espiras o puntos de medición asignadas al tramo "1" usando el método GET.



**Figura 38:** Consulta Entrypoint 3 ( Listar espiras pertenecientes a un tramo)

Una vez realizada la petición, observamos en la **Figura 39** el mensaje de respuesta el cual además de confirmar la existencia del tramo indicado, nos devuelve el nombre de las espiras asignadas a él, siendo estos 3 en total.

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.



```
Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON
1
2   "msg": "Las espiras o puntos de medición asignados al tramos indicado son:",
3   "status": true,
4   "data": [
5     "P101",
6     "Profesor López Piñero",
7     "Peris y Valero"
8   ],
9   "code": "80x1"
10
```

**Figura 39:** Respuesta a la petición de listar todas las espiras relacionadas con cierto tramo

- **Obtener la intensidad media registrada por una espira de medición en una determinada franja horaria:**

Solicitamos al sistema de información que nos devuelva la intensidad media en la espira “101” entre las horas indicadas en la **Figura 40**, las cuales corresponden al formato indicado en el DTO.

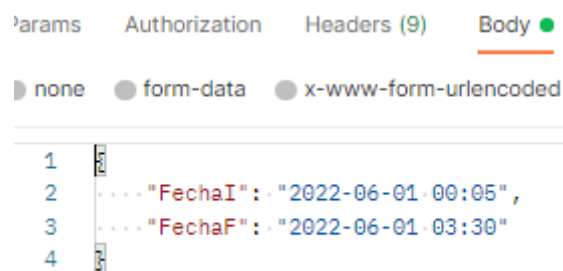


`http://localhost:3000/datos-espiras/espiras/intensidad/101`

```
POST http://localhost:3000/datos-espiras/espiras/intensidad/101
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Query Params
```

KEY	VALUE
Key	Value

**Figura 40:** Consulta Entrypoint 7 (Intensidad media en una espira determinada en una franja horaria determinada)



```
Params Authorization Headers (9) Body
none form-data x-www-form-urlencoded
1
2   ... "FechaI": "2022-06-01 00:05",
3   ... "FechaF": "2022-06-01 03:30"
4
```

**Figura 41:** Cuerpo del formulario o DTO con las horas deseadas

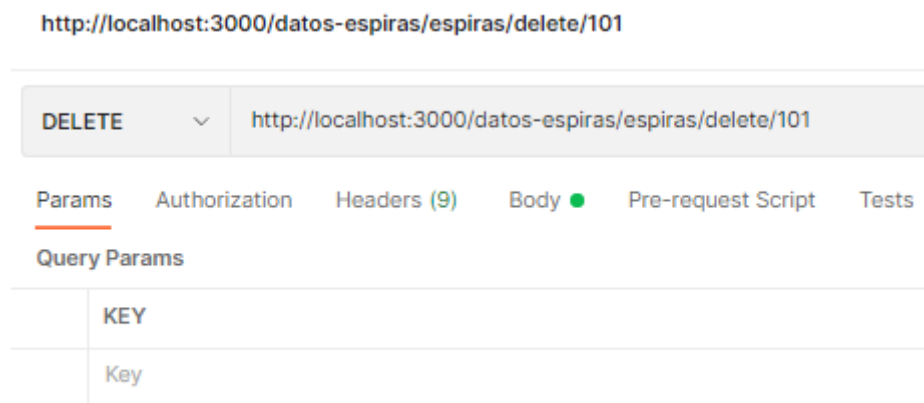
Tras esperar un momento, obtenemos una confirmación de la solicitud y la intensidad media de la espira en las horas seleccionadas, siendo esta 73,80.

```
1 |  
2 | "msg": "La intensidad media de la espira en las horas seleccionadas es: 73.80952380952381",  
3 | "status": true,  
4 | "data": {  
5 |   "FechaI": "2022-06-01 00:05",  
6 |   "FechaF": "2022-06-01 03:30"  
7 | },  
8 | "code": "80x1"  
9 |
```

*Figura 42: Respuesta a la petición de la intensidad media*

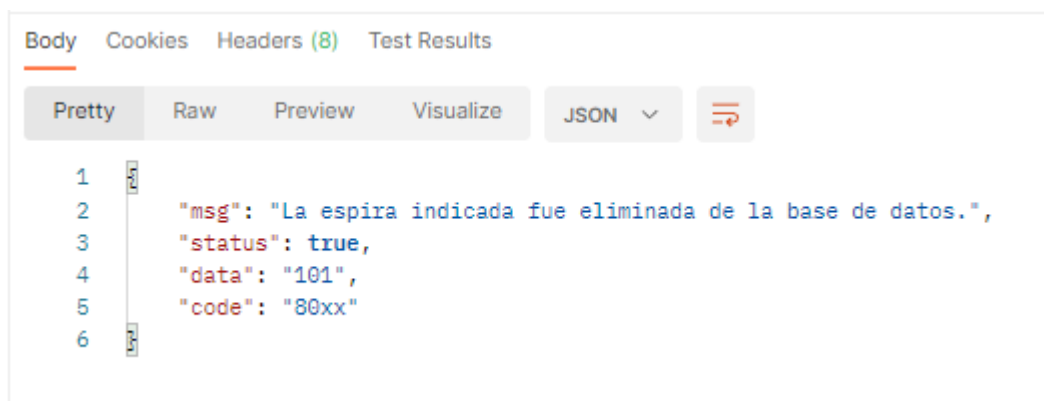
- **Borrar un punto de medición averiado:**

Supongamos que la espira “101” ha sufrido una avería y es necesario eliminarla del sistema. Para ello realizamos una petición indicando el EntryPoint deseado y el identificador de la espira, acompañado del método HTTP Delete.



*Figura 43: Petición de eliminación de una espira*

A continuación ,observamos en la **Figura 44** la respuesta obtenida.



*Figura 44: Respuesta a la petición de eliminación*

## Implementación de un Sistema de Información de soporte al análisis de emisiones relacionadas con el tráfico de la ciudad de València.

Para asegurarnos de su correcta eliminación del sistema, realizaremos una consulta de la intensidad media de la espira eliminada. Como observamos en la **Figura 45**, el sistema nos indica que la espira no existe en la base de datos.

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:3000/datos-espigas/espigas/intensidad/101`
- Method: `POST`
- Query Params table:

KEY	VALUE
Key	Value
- Body tab selected, showing JSON response:

```
1 {  
2   "msg": "La espira indicada no existe",  
3   "status": false,  
4   "code": "80xx"  
5 }
```

**Figura 45:** Petición de obtención de la intensidad media de una espira eliminada

## 6. Conclusiones

---

Ya desarrollada la solución, implementada y validada, debemos analizar si se han cumplido los objetivos establecidos al principio del proyecto.

El objetivo principal del proyecto consistía en diseñar, implementar y evaluar un sistema de información dedicado a la captura y exposición de los datos obtenidos por un sistema de medición de tráfico. La solución desarrollada permite la unificación de los datos generados por las espiras y posteriormente realizar consultas de estos.

Se ha cumplido el objetivo de preparar los datos generados por las espiras, para su inclusión en la base de datos aplicando los conceptos ETL mediante el uso de herramientas ofrecidas por NotePad++ y MongoDB Compass. También, se ha desarrollado una base de datos en MongoDB que almacena el conjunto de datos generados por las espiras en un solo registro. Al mismo tiempo, se ha conseguido diseñar e implementar una API REST mediante NestJS con el fin de acceder de una forma amigable y segura a los datos almacenados en la base de datos. Finalmente, se ha efectuado una evaluación del sistema mediante la realización de unos ejemplos prácticos.

El trabajo realizado mantiene relación con ciertas asignaturas estudiadas a lo largo de la carrera como lo son “Bases de datos” y “Diseño y gestión de bases de datos”, las cuales aportan unos cimientos de conocimiento en el campo del diseño de bases de datos relacionales. Por otro lado, la utilización de un sistema de información para la resolución de esta clase de problemas viene inspirado por la asignatura “Sistemas integrados de información en las organizaciones”. En dicha asignatura se exponen las ventajas de su uso y la gran utilización que gozan los sistemas de información en las organizaciones o en empresas.

Hay que destacar la importancia de la asignatura “Sistemas de información estratégicos” la cual pese a centrarse en el uso de los llamados almacenes de datos, aporta conceptos importantísimos para la utilización de los SI como las nuevas tecnologías de diseño y del correcto uso de las herramientas de consulta. La gran mayoría de las asignaturas previamente mencionadas pertenecen a la rama de Sistemas de Información, dentro del grado en Ingeniería Informática. Esta rama prepara a los estudiantes para concebir sistemas informáticos que den soporte a los procesos de negocio y de toda organización. Por otro lado, forma a los estudiantes en la técnica de definición de requerimiento y solucionar la mejor solución evaluando las posibles alternativas.

Personalmente considero la realización de este trabajo como una grata experiencia. Gracias a este trabajo, he conseguido aplicar los conceptos ofrecidos por las asignaturas previamente comentadas. Por otro lado, he indagado en el mundo del diseño de las API, de sus distintas alternativas y sus diversas funcionalidades. Finalmente, puedo afirmar que la elaboración del trabajo me ha supuesto una gran utilidad personal y ha satisfecho la motivación previa a su realización. He sentido como aplicaba de forma exitosa los distintos conocimientos adquiridos durante estos años y siento que he resuelto el problema como un graduado en informática debería. Todo ello me llena de satisfacción personal y de mayor motivación y experiencia para futuros retos.

## 7. Trabajo futuro

---

En el estado de arte y en el análisis del problema, comentamos la existencia de diversas formas de resolver la problemática que nos encontramos. Desde el principio de este proyecto, se ha optado por la combinación del uso de un sistema de información y una API como solución al problema, debido a su gran utilización en la actualidad para resolver situaciones semejantes. No obstante, la solución planteada no es perfecta, por lo que es posible implementar mejoras que faciliten ya aseguren el correcto funcionamiento del sistema a lo largo del tiempo.

Uno trabajo que supondría una gran mejora del sistema, sería la automatización de los procesos ETL en los datos generados por las espiras del sistema de medición. Anteriormente, en el desarrollo de la solución, se ha mostrado como los ficheros procedentes de las espiras han de ser tratados mediante estos procesos con el fin de asegurar una correcta integración de ellos. En la solución planteada, estos procesos se han realizado de forma manual y tediosa mediante funcionalidades de NotePad++ y MongoDB Compass, por lo que la utilización de una herramienta ETL, facilitaría su automatización, coste y entendimiento. En la actualidad existen multitud de herramientas ETL, utilizadas tanto por las grandes corporaciones tecnológicas como para uso personal. Entre ellas destacan Apache NIFI, Streamsets, Apache AirFlow o AWS Data Pipeline.

Como se ha comentado previamente, la solución propuesta usa como guía de desarrollo los datos generados por las espiras de medición en el mes de Junio de 2022. En el futuro, para garantizar un correcto funcionamiento de la solución desarrollada y un análisis de calidad, sería necesario almacenar todo el histórico de los datos procedentes de sistema de medición de tráfico. Esto supondría albergar todos los datos generados por las espiras desde 2016 hasta la actualidad. Debido a la enorme cantidad de datos a almacenar, un trabajo a realizar sería la sustitución de la base de datos por un almacén de datos o *Data Warehouse*. Estos mecanismos disponen de mayor capacidad de almacenaje y son utilizados por las grandes organizaciones con el propósito de crear un histórico de los datos.

Finalmente, la realización de una app o programa que sirva como interfaz de la API, supondría una mejora en la solución planteada. En la solución presentada en este trabajo, utilizamos la herramienta pública Postman para realizar las consultas a la API REST y la Interfaz estándar ofrecida por Swagger. Pese a que nos permita observar de forma clara los distintos EntryPoints y métodos que realiza nuestra API, la experiencia se queda escasa. La realización de una propia interfaz aportaría la personalización necesaria para satisfacer las necesidades del usuario de una forma más clara y visual.

## 8. Bibliografía

---

- [1] “¿Qué evidencia existe de que la Tierra se está calentando y que los humanos son la causa principal? | NOAA Climate.gov.” <https://www.climate.gov/news-features/climate-qa/%C2%BFqu%C3%A9-evidencia-existe-de-que-la-tierra-se-est%C3%A1-calentando-y-que-los> (accessed May 15, 2022).
- [2] C. Portier *et al.*, “A Human Health Perspective on Climate Change: A Report Outlining Research Needs on the Human Health Effects of Climate Change,” *Environ Health Perspect*, Apr. 2010, doi: 10.1289/EHP.1002272.
- [3] “Definición de calentamiento global - Diccionario panhispánico del español jurídico - RAE.” <https://dpej.rae.es/lema/calentamiento-global> (accessed Feb. 17, 2022).
- [4] “Oficina de Ciudad Inteligente - València Ciudad Inteligente.” <https://smartcity.valencia.es/oficina-ciudad-inteligente/> (accessed Feb. 17, 2022).
- [5] M. A. Mateo Pla *et al.*, “From traffic data to GHG emissions: A novel bottom-up methodology and its application to Valencia city,” *Sustain Cities Soc*, vol. 66, p. 102643, Mar. 2021, doi: 10.1016/J.SCS.2020.102643.
- [6] “Los 6 principales tipos de sistemas de información | Kyocera.” <https://www.kyoceradocumentsolutions.es/es/smarter-workspaces/business-challenges/the-cloud/los-6-principales-tipos-sistemas-informacion.html> (accessed May 25, 2022).
- [7] “Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible.” <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/> (accessed May 26, 2022).
- [8] “THE QUANT CRUNCH HOW THE DEMAND FOR DATA SCIENCE SKILLS IS DISRUPTING THE JOB MARKET,” 2017.
- [9] “Los seis tipos principales de sistemas de informacionMenu.” <https://altametrics.com/es/information-systems/information-system-types.html> (accessed Jun. 04, 2022).
- [10] Vladimir. Zwass, “Management information systems,” p. 896, 1992.
- [11] “(PDF) Principios de Sistemas de Informacion Un Enfoque Administrativo | Michael D Rincon Ocampo - Academia.edu.” [https://www.academia.edu/22302493/Principios\\_de\\_Sistemas\\_de\\_Informacion\\_Un\\_Enfoque\\_Administrativo](https://www.academia.edu/22302493/Principios_de_Sistemas_de_Informacion_Un_Enfoque_Administrativo) (accessed Jun. 04, 2022).
- [12] “LOS COMPONENTES DE LOS SISTEMAS DE INFORMACIÓN – Bienvenido a SIG5A – 5.” <https://sig5a5.wordpress.com/2015/04/24/los-componentes-de-los-sistemas-de-informacion/> (accessed Jun. 28, 2022).



- [13] “Buenas prácticas en el diseño de APIs y Linked Data | datos.gob.es.” <https://datos.gob.es/es/documentacion/buenas-practicas-en-el-diseno-de-apis-y-linked-data> (accessed Jun. 06, 2022).
- [14] “Google Maps Platform | Google Developers.” <https://developers.google.com/maps?hl=es-419> (accessed Jun. 06, 2022).
- [15] “YouTube Data API | Google Developers.” <https://developers.google.com/youtube/v3> (accessed Jun. 06, 2022).
- [16] “Get started with PayPal Developer.” <https://developer.paypal.com/api/rest/> (accessed Jun. 06, 2022).
- [17] “Amazon API Gateway | API Management | Amazon Web Services.” <https://aws.amazon.com/es/api-gateway/> (accessed Jun. 06, 2022).
- [18] “Benefits of APIs.” [http://18f.github.io/API-All-the-X/pages/benefits\\_of\\_apis/](http://18f.github.io/API-All-the-X/pages/benefits_of_apis/) (accessed Jun. 06, 2022).
- [19] “¿Qué es XMLRPC y cómo esta reliquia amenaza la seguridad de tu web?” <https://www.siteground.es/blog/xmlrpc/> (accessed Jun. 07, 2022).
- [20] “JSON-RPC 2.0 Specification.” <https://www.jsonrpc.org/specification> (accessed Jun. 07, 2022).
- [21] “Qué es una base de datos | Oracle México.” <https://www.oracle.com/mx/database/what-is-database/> (accessed Jun. 14, 2022).
- [22] “¿Qué son los procesos ETL?” <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/qu-son-los-procesos-etl> (accessed Jun. 16, 2022).
- [23] “MongoDB: La Plataforma De Datos Para Aplicaciones | MongoDB.” <https://www.mongodb.com/es> (accessed Jun. 16, 2022).
- [24] “NestJS - A progressive Node.js framework.” <https://nestjs.com/> (accessed Jun. 16, 2022).
- [25] “Mongoose ODM v6.3.8.” <https://mongoosejs.com/> (accessed Jun. 16, 2022).
- [26] “Habilitación de CORS para un recurso de la API de REST - Amazon API Gateway.” [https://docs.aws.amazon.com/es\\_es/apigateway/latest/developerguide/how-to-cors.html](https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/how-to-cors.html) (accessed Jun. 28, 2022).
- [27] “MongoDB Compass | MongoDB.” <https://www.mongodb.com/es/products/compass> (accessed Jun. 28, 2022).
- [28] “API Documentation & Design Tools for Teams | Swagger.” <https://swagger.io/> (accessed Aug. 30, 2022).
- [29] “Notepad++.” <https://notepad-plus-plus.org/> (accessed Jun. 28, 2022).
- [30] “Postman API Platform | Sign Up for Free.” <https://www.postman.com/> (accessed Jun. 28, 2022).

- [31] “[Resuelta] character-encoding | ¿Qué es el formato ANSI?”  
<https://www.iteramos.com/pregunta/14054/que-es-el-formato-ansi> (accessed Jun. 28, 2022).
- [32] “HTML UTF-8 Reference.” [https://www.w3schools.com/charsets/ref\\_html\\_utf8.asp](https://www.w3schools.com/charsets/ref_html_utf8.asp)  
(accessed Jun. 28, 2022).

## Anexo I. Relación con ODS

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS):

*Tabla 5: Grado de relación del proyecto realizado con los ODS*

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. <b>Fin de la pobreza.</b>				<b>X</b>
ODS 2. <b>Hambre cero.</b>				<b>X</b>
ODS 3. <b>Salud y bienestar.</b>	<b>X</b>			
ODS 4. <b>Educación de calidad.</b>				<b>X</b>
ODS 5. <b>Igualdad de género.</b>				<b>X</b>
ODS 6. <b>Agua limpia y saneamiento.</b>				<b>X</b>
ODS 7. <b>Energía asequible y no contaminante.</b>			<b>X</b>	
ODS 8. <b>Trabajo decente y crecimiento económico.</b>			<b>X</b>	
ODS 9. <b>Industria, innovación e infraestructuras.</b>		<b>X</b>		
ODS 10. <b>Reducción de las desigualdades.</b>				<b>X</b>
ODS 11. <b>Ciudades y comunidades sostenibles.</b>	<b>X</b>			
ODS 12. <b>Producción y consumo responsables.</b>	<b>X</b>			
ODS 13. <b>Acción por el clima.</b>	<b>X</b>			
ODS 14. <b>Vida submarina.</b>				<b>X</b>
ODS 15. <b>Vida de ecosistemas terrestres.</b>				<b>X</b>
ODS 16. <b>Paz, justicia e instituciones sólidas.</b>				<b>X</b>
ODS 17. <b>Alianzas para lograr objetivos.</b>		<b>X</b>		

A continuación, se muestra una reflexión personal acerca de la relación del TFG/TFM con los ODS y con el/los ODS más relacionados, acompañados de las metas que cumplen de dicho objetivo:

- **ODS 3. Salud y bienestar:** Pienso que este objetivo tiene una gran relación con el proyecto debido a que, con el estudio del tráfico de una urbe, se puede reducir el número de muertes y lesiones causadas por accidentes de tráfico. Por otro lado, este trabajo está orientado al análisis de la contaminación en determinadas zonas de una ciudad, por lo que su investigación permitirá reducir el número de muertes y enfermedades producidas por la contaminación de aire (Metas 3.6 y 3.9).
- **ODS 7. Energía asequible y no contaminante:** Este objetivo tiene una cierta relación con el proyecto teniendo en cuenta que consiste en una investigación relacionada con una tecnología relativa a la eficiencia energética y de reducción del uso de combustibles fósiles. Además, que la idea presentada trata de promover el uso de infraestructuras de tecnologías limpias (Meta 7A).

- **ODS 8. Trabajo decente y crecimiento económico:** El proyecto pretende reducir la producción y el consumo eficiente para evitar una degradación del medio ambiente con lo que existe una leve relación con el objetivo (Meta 8.4).
- **ODS 9. Industria, innovación e infraestructuras:** En este trabajo, se desarrolla una infraestructura sostenible la cual apoya el bienestar humano mediante el estudio de la contaminación generada. Al mismo tiempo, hay que indicar que dicho sistema puede ser desarrollado e implementado en países menos desarrollados (Metas 9.1 y 9.A).
- **ODS 11. Ciudades y comunidades sostenibles:** Pienso que este ODS tiene una gran relación con el trabajo, considerando que la idea expuesta pretende reducir el impacto ambiental generado por las ciudades. También, el análisis de los datos generados por el sistema, pueden servir para aumentar las políticas para la mitigación del cambio climático y la prevención de desastres medio ambientales (Metas 11.6 y 11.B).
- **ODS 12. Producción y consumo responsables:** A lo largo del proyecto, elaboramos un instrumento que permite vigilar los efectos en el desarrollo sostenible, facilitando que cada vez más personas tengan los conocimientos necesarios sobre la situación del planeta y el desarrollo sostenible (Metas 12.2 , 12.8 y 12.B).
- **ODS 13. Acción por el clima:** Creo que este ODS está muy relacionado con el trabajo debido a que el sistema desarrollado puede ser considerado como una medida relativa a combatir el cambio climático. Por otro lado, permitirá mejorar la educación y la sensibilidad sobre el cambio climático, usando los datos generados para mitigar sus efectos (Metas 13.2 y 13.3).
- **ODS 17. Alianzas para lograr objetivos:** Creo que Dicho sistema y su implementación puede ser usada en otros países menos desarrollados para intentar paliar aún más los efectos del cambio climático. Como he comentado previamente, los datos obtenidos por el sistema pueden ser utilizados para realizar políticas coherentes para el desarrollo sostenible o como indicador para medir el progreso de éste (Metas 17.7, 17.14 y 17.19).