UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Máster en Ingeniería de Computadores y Redes

Trabajo Fin de Máster

Visualización de datos para el IoT

Autor: *XU WEI*

Director(es): *< Dr. Pietro Manzoni >*

<Diciembre,2020>

Resumen

La visualización de los datos en IoT es un aspecto crucial. La necesidad de integración de varias fuentes de información ya asociación de meta-información como la geolocalización y el instante temporal de obtención.

En este artículo se quiere estudiar las herramientas a disposición para la visualización y para la integración de datos tanto de sensores básicos como de información en la web (http://gobiernoabierto.valencia.es/en/info-api/). Se utilizarán APIs como MQTT y REST para la obtención de datos y librerías como https://leafletjs.com/ para simplificar las operaciones de visualización en mapas y para la aplicación de filtros de agregación.

Palabras claves: Internet de las cosas (IOT), interfaz de programación de aplicaciones (API), Leaflet.

Abstract

The visualization of data in IoT is a crucial aspect as it requires the integration of several sources of information and the need to associate meta-information such as geolocation and the time of acquisition. In this Article we want to study the tools available for the visualization and integration of data from both basic sensors and information on the web (http://gobiernoabierto.valencia.es/en/info-api/) APIs will be used such as MQTT and REST to obtain data and libraries such as https://leafletjs.com/ to simplify the operations of visualization in maps and for the application of aggregation filters.

Keywords: Internet of Things (IOT), Application Programming Interface (API), Leaflet.

Índice general

LISTA DE FIGURAS

1. Capítulo 1

Introducción

Desde finales del siglo XX, la economía mundial ha continuado en recesión y existe una necesidad urgente de que surja una tecnología emergente para promover la transformación de la productividad y convertirse en un nuevo motor de crecimiento económico. Mediante la integración de la ciencia y la tecnología originales, las personas han descubierto un nuevo modelo tecnológico.

En la década de 1980, Estados Unidos y Europa tenían el concepto de edificios inteligentes y hogares inteligentes en ciernes silenciosamente; en 1995, el libro de Bill Gates "The Road to the Future" hizo la aplicación de la tecnología de Internet de las cosas en escenarios domésticos. La elaboración detallada les dio a las personas una comprensión preliminar de la aplicación de Internet de las cosas; en 1999, el Laboratorio de identificación automática del Instituto de Tecnología de Massachusetts presentó un conjunto de soluciones para construir la Internet de las cosas basadas en el estándar de código electrónico del producto y la Internet de las cosas. La viabilidad de la tecnología es concreta; sin embargo, cuando se trata de la fuente del término "Internet de las cosas", fue solo después de que la Unión Internacional de Telecomunicaciones (ITU) publicara el informe "Internet de las cosas" en 2005; y más tarde, En 2009, la estrategia de Internet de las cosas de tres pasos de IBM generó respuestas generalizadas en la industria y el mundo académico a nivel mundial.

El desarrollo continuo de la tecnología IOT también ha traído consigo innovaciones tecnológicas relacionadas, entre las que se encuentra la visualización de datos IOT.

1.1 Motivación

Con el rápido desarrollo de la era de la Internet de las cosas, los datos generados por la Internet de las cosas también han crecido a una velocidad geométrica. La cuantificación de datos se ha convertido en una tendencia de desarrollo inevitable. En este caso, con la ayuda de medios gráficos, la información se puede comunicar de manera clara y efectiva.

Es por este motivo, se lleva a cabo el desarrollo de este proyecto. En este proyecto como primer punto, se han obtenido los datos abiertos disponibles en la página del Ayuntamiento de Valencia. Además, con ayuda de los APIs se obtienen datos como estado del tráfico en tiempo real, contaminación y salud etc. Luego esos datos se visualizan en un mapa interactivo.

El mapa, y la obtención de datos en este trabajo se han creado con la ayuda de la librería leaflet de JavaScript.

1.2 Objetivos

Este Trabajo de Fin de Máster tiene como objetivos:

- Desarrollar un Mapa interactivo con los datos abiertos de la ciudad de Valencia, utilizando la librería Leaflet.
- Realizar pruebas de envió de mensajes de la página de Ayuntamiento de Valencia con diferentes APIs .
- Analizar los datos obtenidos, y Visualizar estos datos en el mapa.

1.3 Metodología

Para el desarrollo de este proyecto, como primer punto se ha utilizado Georeferenced Data API, la cual permite obtener lo datos del portal de datos abiertos del ayuntamiento de Valencia.

En segundo lugar, en el mapa es posible obtener, la visualización de los datos. Los cuales fueron agrupados por temáticas como: transporte público, Fallas y monumentos.

1.4 Estructura

El presente Trabajo de Fin de Máster consta de cuatro capítulos, En el primer capítulo, encontramos la idea preliminar del problema que queremos resolver y el objetivo general del trabajo. En el segundo capítulo, presentamos los antecedentes del diseño en detalle, las herramientas de transferencia de datos (API) y las librerías utilizadas, también se expone una descripción general del formato de archivo Geojson y la información básica del compilador de código.

El Capítulo 3 describe con más detalle los complementos utilizados por la biblioteca de folletos, los pasos para crear un mapa y la visualización de datos en el mapa.

En el Capítulo 4, presentaremos las conclusiones extraídas del desarrollo del trabajo y nos referiremos al trabajo futuro.

2. Capítulo 2

Antecedentes

El auge del Internet de las cosas tiene una relación inseparable con el desarrollo de la tecnología de la información. Puede permitir que los seres humanos y las cosas interactúen, y dar un salto cualitativo en el desarrollo de la sociedad humana. Con el rápido desarrollo de la ciencia y la tecnología de la información. Los datos son el portador de información y la información es la manifestación de datos. Si desea que los datos produzcan valor, debe extraer su información útil. Esto es lo que hace la visualización de datos.

2.1 Leaflet

Leaflet [1], es la librería JavaScript de código abierto líder para mapas interactivos compatibles con dispositivos móviles. Con un peso de aproximadamente 39 KB de JS, tiene todas las funciones de mapeo que la mayoría de los desarrolladores necesitan.

El Leaflet está diseñado teniendo en cuenta la simplicidad, el rendimiento y la facilidad de uso. Funciona de manera eficiente en todas las principales plataformas móviles y de escritorio, se puede ampliar con muchos complementos, tiene una API hermosa, fácil de usar y bien documentada, y un código fuente simple y legible al que es un placer contribuir.

Figura 1 : ejemplo de Leaflet.

Aquí creamos un mapa en el ' map' div, agregamos mosaicos de nuestra elección y luego agregamos un marcador con algo de texto en una ventana emergente:

```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://{s}. tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
}).addTo(map);
```

```
L.marker([51.5, -0.09]).addTo(map)
    .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
        .openPopup();
```

2.2 API

API [2], (Application Programming Interface, interfaz de programación de aplicaciones) son algunas funciones predefinidas o se refiere al acuerdo entre diferentes componentes del sistema de software. Se utiliza para proporcionar un conjunto de rutinas a las que las aplicaciones y los desarrolladores pueden acceder en función de cierto software o hardware sin tener que acceder al código fuente o comprender los detalles del mecanismo de trabajo interno.

2.2.1 Introducción de API

El sistema operativo (OS) es la interfaz entre el usuario y el sistema de hardware de la computadora. Con la ayuda del sistema operativo, el usuario puede manipular de manera rápida, efectiva, segura y confiable varios recursos en el sistema informático para procesar sus propios programas. Para permitir que los usuarios utilicen el sistema operativo de manera conveniente, el sistema operativo proporciona a los usuarios los siguientes dos tipos de interfaces:

(1) Interfaz de usuario: el sistema operativo proporciona específicamente a los usuarios la "interfaz entre el usuario y el sistema operativo", generalmente denominada interfaz de usuario. Esta interfaz admite la interacción entre el usuario y el sistema operativo, es decir, el usuario solicita un servicio específico del sistema operativo y el sistema devuelve el resultado del servicio al usuario.

(2) Interfaz de programa: el sistema operativo proporciona a los programadores una "interfaz de programa y sistema operativo", denominada interfaz de programa, también conocida como interfaz de programación de aplicaciones (API). Esta interfaz es utilizada por los programadores durante la programación. A través de esta interfaz, los sistemas y aplicaciones pueden acceder a los recursos del sistema y obtener servicios del SO durante la ejecución. También es la única forma de que los programas obtengan servicios del sistema operativo. La interfaz del programa de la mayoría de los sistemas operativos se compone de un conjunto de llamadas al sistema, y cada llamada al sistema es una subrutina que puede completar una función específica.

La interfaz de programación de aplicaciones, también conocida como interfaz de programación de aplicaciones, es una colección de definiciones, procedimientos y protocolos, y se comunica entre el software de computadora a través de interfaces API. Una de las funciones principales de la API es proporcionar un conjunto común de funciones. La API también es una especie de middleware que permite compartir datos para varias plataformas.

En la práctica de la programación, el diseño de la interfaz de programación debe primero dividir razonablemente las responsabilidades del sistema de software. Un buen diseño de interfaz puede reducir la interdependencia de varias partes del sistema, mejorar la cohesión de las unidades constituyentes y reducir el grado de acoplamiento entre las unidades constituyentes, mejorando así la capacidad de mantenimiento y escalabilidad del sistema.

2.2.2 Clasificación

API de Windows

Las funciones de API están contenidas en el archivo de librería de vínculos dinámicos en el directorio del sistema de Windows. La API de Windows es un conjunto de funciones predefinidas de Windows que se utilizan para controlar la apariencia y el comportamiento de varios componentes de Windows. Cada acción del usuario hará que se ejecuten una o varias funciones para decirle a Windows lo que sucedió. Esto es similar al código natural de Windows en cierto modo. Otros lenguajes solo proporcionan una forma de acceder a la API de forma automática y sencilla. Cuando hace clic en un botón del formulario, Windows enviará un mensaje al formulario, VB recibe la llamada y genera un evento específico después del análisis.

Es más fácil de entender: además de coordinar la ejecución de aplicaciones, la asignación de memoria y la administración de recursos del sistema, el sistema Windows también es un gran centro de servicios. Llamar a varios servicios de este centro de servicio (cada tipo de servicio es una función) puede ayudar a la aplicación a lograr el propósito de abrir ventanas, dibujar gráficos y usar dispositivos periféricos. Dado que los objetos servidos por estas funciones son aplicaciones, se denominan Programación de aplicaciones. Interfaz, denominada función API. La API WIN32 es también la interfaz de programación de aplicaciones de la plataforma Microsoft Windows de 32 bits.

Todas las aplicaciones ejecutadas en el entorno de trabajo de Windows pueden llamar a la API de Windows.

API de Linux

En Linux, la API de la interfaz de programación de usuario sigue el estándar POSIX, el estándar de interfaz de programación de aplicaciones más popular en UNIX. El estándar POSIX es un sistema estándar desarrollado conjuntamente por IEEE e ISO / IEC. Según la práctica y la experiencia de UNIX existente en ese momento, el estándar describe la API de la interfaz de programación de llamadas del sistema del sistema operativo para garantizar que los programas de aplicación se puedan trasplantar y ejecutar en múltiples sistemas operativos en el nivel del programa fuente. Estas interfaces de programación de llamadas al sistema se implementan principalmente a través de la librería C (CSL).

2.2.3 Función de programa

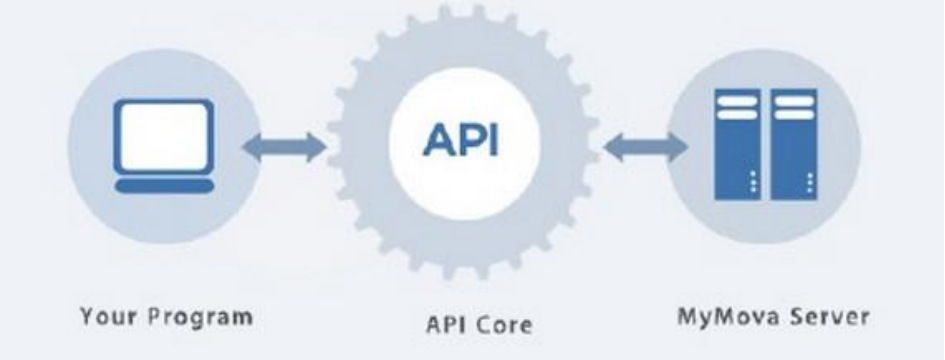

Figura 2: Función de API [3].

Llamada a procedimiento remoto (RPC): La comunicación entre programas se realiza a través del proceso (o tarea) que actúa sobre el búfer de datos compartidos.

Lenguaje de consulta estándar (SQL): es un lenguaje de consulta estándar para acceder a datos, que permite compartir datos entre aplicaciones a través de una base de datos general.

Transferencia de archivos: la transferencia de archivos permite compartir datos entre aplicaciones mediante el envío de archivos formateados.

Entrega de información: se refiere a información en formato pequeño entre aplicaciones poco acopladas o estrechamente acopladas, y el intercambio de datos se logra mediante la comunicación directa entre programas.

Los estándares que se aplican actualmente a la API incluyen la API SQL estándar ANSI. También hay algunos estándares para otros tipos de estándares aún en desarrollo. La API se puede aplicar a todas las plataformas informáticas y sistemas operativos. Estas API conectan datos en diferentes formatos. Cada formato de datos requiere diferentes comandos y parámetros de datos para lograr una correcta comunicación de datos, pero al mismo tiempo se generarán diferentes tipos de errores. Por lo tanto, además del conocimiento requerido para realizar tareas de intercambio de datos, este tipo de API también debe resolver muchos problemas de parámetros de red y posibles condiciones de error, es decir, cada aplicación debe saber si tiene un desempeño sólido para soportar la comunicación entre programas. Por el contrario, debido a que esta API solo trata con un formato de información, la API de entrega de información en esta situación solo proporciona un pequeño subconjunto de comandos, parámetros de red y condiciones de error. Debido a esto, el método de API de entrega reduce en gran medida la complejidad del sistema, por lo que cuando la aplicación necesita realizar el intercambio de datos a través de múltiples plataformas, el tipo de API de entrega de información es una opción ideal.

2.3 Georeferenced Data API

2.3.1 Acceso de la API

Cualquier acceso a la API de http://mapas.valencia.es/lanzadera

necesita autenticación básica. Para ello, habrá que solicitar la creación de un usuario y una contraseña, en el portal de datos abiertos del Ayuntamiento de Valencia, que se empleará en todas las llamadas a la API. Una llamada por HTTP deberá incluir la cabecera de autenticación básica con el usuario y la contraseña que se le proporcione codificada en Base64.

Figura 3: Solicitud de uso de la API [4].

Si la solicitud tiene éxito, recibirá un correo electrónico con su nombre de usuario y contraseña:

Re: Nueva Solicitud de uso de la API

AD

Ayuntamiento de Valencia <appvalencia@valencia.es>
收件人 wx13350358567@gmail.com

respecto a su solicitud de autorización de API le facilitamos

Usuario: wx1335
Password: 7wcjpn4a

un saludo

El 19/11/2020 a las 13:00, webmaster@valencia.es escribió:
>
> Datos recibidos
> autorización API
>
>
>
> Nombre: XU
>
> Apellidos: WEI WEI
>
> Empresa: Universidad Politécnica de Valencia
>
> Motivo: trabajo fin de master
>
> Teléfono: 635485334
>
> E-mail: wx13350358567@gmail.com
>
> Fecha solicitud: 19/11/2020 13:00
>

Figura 4:Licencia API solicitada.

2.3.2 Uso de la API

Petición Tipo [5]: GET

URL: http://mapas.valencia.es/lanzadera/gps/aparcamientos/{lat}/{lon}

Parámetros: • {lat}: latitud • {lon}: longitud La latitud y longitud se deben multiplicar por diez elevado a seis (106) y no llevarán decimales Respuesta Si se ha producido un error inesperado devolverá un código de estado HTTP 400. En cualquier otro caso devolverá HTTP 200 y una respuesta en formato json. Siempre devolverá un array de los elementos ordenados por distancia

• Si ha habido un error en la consulta, el primer elemento tendrá: o título: “ERROR”

• Si no se han encontrado resultados, el primer elemento tendrá: o título: “SIN_RESULTADOS”

• Si el servicio no está disponible temporalmente, el primer elemento tendrá: o título: “NO_DISPONIBLE”

• Si ha encontrado algo: o lon+Destino, lat+Destino: la posición del aparcamiento. La coordenada hay que dividirla por 106 para obtener la real o distancia: en metros o titulo: “APARCAMIENTOS”, aunque puede variar o mensaje: nombre del aparcamiento y el número de plazas libres. Incluye un separador de retorno de línea (\n) entre ambos campos.

2.4 GEOJSON

GeoJSON es un formato para codificar varias estructuras de datos geográficos. Los objetos GeoJSON pueden representar geometría, características o colecciones de características. GeoJSON admite los siguientes tipos de geometría: punto, línea, área, multipunto, multilínea, multisuperficie y colección geométrica. La característica en GeoJSON contiene un objeto geométrico y otros atributos, y el conjunto de características representa una serie de características.

2.4.1 Objeto de geometría

Geometry [6] es un objeto GeoJSON y el valor del miembro de tipo es una de las siguientes cadenas: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon" o "GeometryCollection".

Cualquier tipo de objeto de geometría GeoJSON excepto "GeometryCollection" debe tener un miembro llamado "coordenadas". El valor del miembro de coordenadas es siempre una matriz. La estructura de los elementos de esta matriz está determinada por el tipo de geometría.

Los diferentes objetos GEOJSON se describen a continuación:

• posición：La posición es la estructura geométrica básica. El miembro de "coordenadas" de un objeto geométrico consta de una posición (aquí es un punto geométrico), una matriz de posición (línea o multipunto geométrico), una matriz de matrices de posición (cara, polilínea) o una matriz multidimensional de posiciones.

La posición está representada por una matriz de números. Debe haber al menos dos elementos, puede haber más elementos. El orden de los elementos debe seguir el orden x, y, z (este, norte, altura de las coordenadas en el sistema de referencia de coordenadas proyectadas o la longitud de coordenadas, latitud, altura del sistema de referencia de coordenadas geográficas).

• **Point**：Para el tipo Point, el miembro de coordenadas debe ser una sola posición.

• **MultiPoint**：Para el tipo MultiPoint, el miembro de coordenadas debe ser una matriz de posición.

• **LineString**：Para el tipo LineString, el miembro de coordenadas debe ser una matriz de MultPoint.

LinearRing (anillo lineal) es una línea cerrada con 4 o más posiciones. La primera y la última posición son iguales (representan el mismo punto). Aunque LinearRing no se usa claramente como el tipo de geometría GeoJSON, se menciona en la definición del tipo de geometría poligonal (Polígono).

• **MultiLineString**：Para el tipo MultiLineString, el miembro de coordenadas debe ser una matriz de LineString.

• **Polygon**：Para el tipo Polygon, el miembro de coordenadas debe ser una matriz de LinearRing. Para caras con varios LinearRings, el primer anillo debe ser un anillo exterior y los otros deben ser anillos interiores o agujeros.

• **MultiPlygon**：Para el tipo MultiPlygon, el miembro de coordenadas debe ser una matriz Polygon.

• **GeometryCollection**：Un objeto GeoJSON de tipo GeometryCollection es un objeto de colección, que representa una colección de objetos geométricos.

La colección de geometrías debe tener un miembro denominado geometrías. El valor correspondiente a las geometrías es una matriz. Cada elemento de esta matriz es un objeto geométrico GeoJSON.

2.4.2 Feature

Un objeto GeoJSON de tipo " Feature " es un objeto de característica.

• El objeto de entidad debe tener un miembro llamado "geometría", y el valor de este miembro de geometría es el objeto geométrico definido anteriormente o el valor nulo de JSON.

• El juego de características debe tener un miembro llamado "properties", y el valor de este miembro de propiedad es un objeto (cualquier objeto JSON o valor null JSON).

• Si la característica es un identificador de uso común, entonces este identificador debe contener un miembro de objeto de característica llamado "id".

2.4.3 FeatureCollection

El objeto GeoJSON de tipo "FeatureCollection" es un objeto de colección de características.

Un objeto de tipo "FeatureCollection" debe tener un miembro llamado "features". El valor correspondiente a "características" es una matriz. Cada elemento de esta matriz es el objeto de característica definido anteriormente.

por ejemplo:

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [102.0, 0.5]
            },
            "properties": {
                "prop0": "value 0"
            }
        },
        {
            "type": "Feature",
            "geometry": {
                "type": "LineString",
                "coordinates": [
                    [102.0, 0.0],
                    [103.0, 1.0],
                    [104.0, 0.0],
                    [105.0, 1.0]
                ]
            },
            "properties": {
                "prop0": "value 0",
                "prop1": 0.0
            }
        },
        {
            "type": "Feature",
            "geometry": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [100.0, 0.0],
                        [101.0, 0.0],
                        [101.0, 1.0],
                        [100.0, 1.0],
```

```
                        [100.0, 0.0]
                    ]
                ]
            },
            "properties": {
                "prop0": "value 0",
                "prop1": {
                    "este": "esto"
                }
            }
        }
    ]
}
```

2.4.4 CRS

El sistema de referencia de coordenadas (CRS) de un objeto GeoJSON está determinado por su miembro "crs" (refiriéndose al objeto CRS a continuación). Si el objeto no tiene ningún miembro crs, entonces el miembro crs de su objeto padre o abuelo puede adquirirse como su objeto crs. Si el miembro crs no se obtiene de esta manera, el CRS predeterminado se aplicará al objeto GeoJSON.

- El CRS predeterminado es el sistema de referencia de coordenadas geográficas, que utiliza datos WGS84, y las unidades de longitud y altura son notaciones decimales.

- El valor del miembro llamado "crs" debe ser un objeto JSON (refiriéndose al objeto CRS a continuación) o JSON null. Si el valor de CRS es null, se supone que no hay CRS.

- El miembro crs debe ubicarse en la parte superior del objeto GeoJSON en la estructura jerárquica (en el orden de colección de características, característica y geometría), y no debe repetirse ni sobrescribirse en el objeto self o nieto.

- El valor del miembro de type debe ser una cadena, que indica el tipo de objeto CRS.

- Hay dos objetos obligatorios para los objetos CRS no vacíos: " type " y " properties ".

- El valor del miembro de atributo debe ser un objeto.

- CRS no puede cambiar el orden de las coordenadas.

2.4.4.1 Nombre CRS

Los objetos CRS pueden indicar el sistema de referencia de coordenadas por su nombre. En este caso, el valor de su miembro "tipo" debe ser la cadena "name". El valor de su miembro "properties" debe ser un objeto que contenga el miembro "name". El valor de este miembro "name" debe ser una cadena que identifique el sistema de referencia de coordenadas. Por ejemplo, el URN del OGC CRS de "urn: ogc: def: crs: OGC: 1.3: CRS84" debe usarse con preferencia al antiguo identificador como "EPSG: 4326":

```
"crs": {
    "type": "name",
    "properties": {
        "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
    }
```

Los objetos CRS también se pueden conectar a los parámetros CRS en Internet. En este caso, el valor de su miembro "type " debe ser la cadena "link", y el valor de su miembro " properties " debe ser un objeto de conexión.

El objeto de conexión consta de un miembro obligatorio: "href" y un miembro opcional: " type ".

El valor del miembro "href" requerida debe ser un URI dereferenced (Uniform Resource Identifier).

El valor del miembro "type " opcional debe ser una cadena, y esta cadena implica el formato utilizado para representar los parámetros de CRS en el URI proporcionado. Los valores recomendados son: "proj4", "ogcwkt", esriwkt ", pero se pueden utilizar otros valores:

```
"crs": {
    "type": "link",
    "properties": {
        "href": "http://example.com/crs/42",
        "type": "proj4"
    }
}
```

2.4 Compilador de Código

En este proyecto se han utilizado la siguiente herramienta: HBuilder

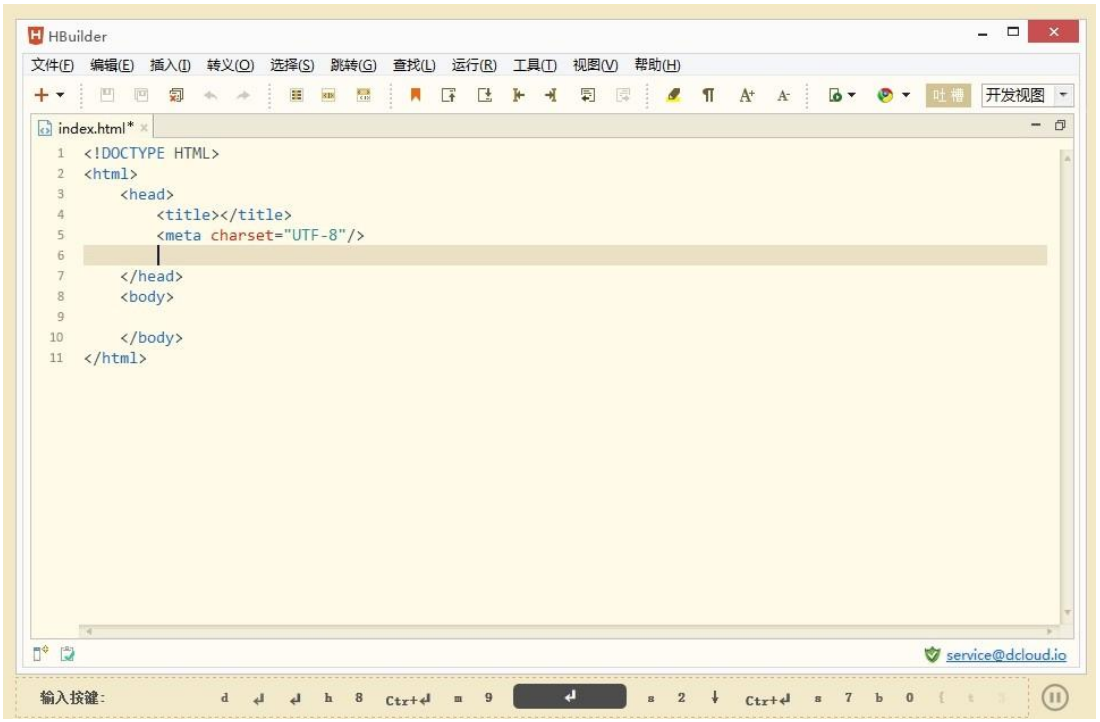

Figura 5:Interfaz de usuario de HBuilder [7].

HBuilder es un IDE de desarrollo web que admite HTML5 de DCloud (Digital Paradise). "La rapidez es la mayor ventaja de HBuilder. A través de indicaciones de sintaxis completas, métodos de entrada de código, bloques de código y muchas facilidades de soporte, HBuilder puede mejorar enormemente la eficiencia de desarrollo de HTML, js y css.

HBuilder, con "rápido" como núcleo, introduce el concepto de "sintaxis de atajos", que resuelve hábilmente el problema de demasiados atajos que afectan a muchos desarrolladores y no pueden recordar. Los desarrolladores solo necesitan recordar algunas gramáticas para realizar rápidamente saltos, escapes y otras operaciones. Por ejemplo, alt + [es saltar a paréntesis, alt + ’es saltar a comillas, alt + letra es saltar a elementos del menú, y alt + left es saltar a la última posición del cursor. Y Ctrl es varias operaciones, como ctrl + d es eliminar una línea. shift está escapando.

Además, el ecosistema de HBuilder puede ser el ecosistema de IDE web más rico porque es compatible con los complementos de Eclipse y Ruby Bundle. SVN, git, ftp, PHP, less y otras tecnologías tienen complementos de Eclipse.

HBuilder está escrito en Java, C, Web y Ruby. El cuerpo principal de HBuilder está escrito en Java, que se basa en Eclipse, por lo que es naturalmente compatible con los complementos de Eclipse. Pero debido a que Java es demasiado ineficiente, el lanzador se escribió en C. El diseño de la interfaz verde suave de HBuilder necesita ajustar dinámicamente el brillo de la

pantalla. También es compatible con la depuración conjunta de la máquina real del cable de datos del teléfono móvil, y todos están escritos en C.

Muchas interfaces de HBuilder, como la interfaz de información del usuario, se crean utilizando tecnología web, que es hermosa y rápida de desarrollar.

2.5 La base de datos

2.5.1 InfluxDB

Para obtener datos y facilitar la gestión, se ha utilizado una base de datos para almacenar la información. La base de datos utilizada es InfluxDB.

InfluxDB es una serie de datos de código abierto desarrollado por InfluxData. Está escrito por el lenguaje de programación Go y se centra en consultar y almacenar datos de series de tiempo con alto rendimiento. InfluxDB se usa ampliamente en escenarios como el monitoreo de datos de sistemas de almacenamiento y datos en tiempo real en la industria de IoT.

Para obtener los datos, de la pagina web y almacenarlos en la base de datos, se utilizado el código mostrado a continuación.

Utilizamos new para crear una entrada en la base de datos, escribimos la direccion, local de la base de datos, apuntamos a uno de los capos que deseamos obtener, y finalmente se envían con el comando send.

```
influxdb = new
Influxdb('http://127.0.0.1:8086/write?db=text&u=admin&p=admin',true);
influxdb.point("estado_trafico1",{key: '20201117'},{lat:latlng_1[0],ln
g:latlng_1[1],denominacion: denominacion_1,modified: modified_1}, time
stamp);
influxdb.send();
```

2.5.1 Uso de InfluxDB

•Crea una base de datos:

```
> crear database "text"
```

• Mostrar todas las bases de datos:

```
> show databases
name: databases
name
----
_internal
text
```

- Usar base de datos:

```
> use text
Using database text
```

- Mostrar todas las tablas en la base de datos:

```
> show measurements
name: measurements
name
----
estado_trafico
estado_traficol
>
```

- Consultar datos en la base de datos:

```
> select * from estado_traficol
name: estado_traficol
time          denominacion                                key      lat               lng               modified
----          ------------                                ---      ---               ---               --------
1605616229000 ACCESODESDEEDUARDOPRIMOYÚFERAACAMINOMORERAS 20201117 39.474484048938315 35.520678512231356 2020-11-16T15:39:07.239+01:00
1605616229000 ACCESODESDEEDUARDOPRIMOYÚFERAACAMINOMORERAS 20201117 39.47816210024783  35.516508647910996 2020-11-16T15:39:07.239+01:00
1605616229000 BARÓNDEC�?RCER(MERCADOCENTRAL)              20201117 39.46857646536295  35.49939782535507  2020-11-16T15:39:07.238+01:00
1605616229000 BARÓNDEC�?RCER(MERCADOCENTRAL)              20201117 39.465845430357625 35.506443458172214 2020-11-16T15:39:07.238+01:00
1605616229000 CAMINODELASMORERASHACIACARRERADELRIO        20201117 39.48835930802364  35.50737808293368  2020-11-16T15:39:07.239+01:00
1605616229000 CAMINODELASMORERASHACIACARRERADELRIO        20201117 39.48902792472701  35.50450231443688  2020-11-16T15:39:07.239+01:00
1605616229000 CAMINODELASMORERASHACIAPUENTEDEASTILLEROS   20201117 39.49220376555787  35.50040434432895  2020-11-16T15:39:07.239+01:00
1605616229000 CAMINODELASMORERASHACIAPUENTEDEASTILLEROS   20201117 39.49153517964073  35.505580727623176 2020-11-16T15:39:07.239+01:00
1605616229000 CARRERARIOHACIACAMINOMORERAS                20201117 39.482731527264804 35.52161313699281  2020-11-16T15:39:07.239+01:00
1605616229000 CARRERARIOHACIACAMINOMORERAS                20201117 39.484458914746874 35.51154794725402  2020-11-16T15:39:07.239+01:00
```

Como se mencionó anteriormente, ahora tenemos acceso a los datos y podemos administrar los datos en la base de datos. En el siguiente paso, podemos adjuntar los datos al archivo Geojson para usar.

3. Capítulo 3

Diseño y Metodología de Desarrollo

Los lenguajes de programación usados en la construcción del mapa, Se han utilizado HTML, CSS y JavaScript. Además de la librería de JavaScript se ha usado Leaflet.

3.1 Diseña un mapa básico

Antes de escribir cualquier código para el mapa, debe realizar los siguientes pasos de preparación en la página:

- Incluya el archivo CSS del Leaflet en la sección del encabezado de su documento:

```
<link rel="stylesheet" href="./leaflet.css"integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4PsZChSR7A==" crossorigin=""/>
```

- Incluya el archivo JavaScript del Leaflet después del CSS del Leaflet:

```
<script src="./leaflet.js"> </script>
```

- Coloque un elemento div con un ID específico en el mapa:

```
<div id='map'></div>
```

- Asegúrese de que el contenedor del mapa tenga una altura definida, configurándolo en CSS:

```
<style>
        html, body {
            height: 100%;
            margin: 0;
        }
        #map {

            height: 700px;
        }
    </style>
```

- A continuación, inicializaremos el mapa y configuraremos su vista en las coordenadas geográficas y el nivel de zoom de nuestra elección:

```
var valencia=[39.4810070700481, -0.34123508748639353];
var Map = L.map('mapid').setView(valencia, 15);
```

- Luego, agregaremos una capa de mosaico para agregar a nuestro mapa, en este caso es una capa de mosaico de Mapbox Streets.

```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
```

```
        }).addTo(Map);
```

- Podemos establecer un marcador para determinar la ubicación de la universidad.

```
L.marker(valencia).addTo(Map)
                .bindPopup("<b>Hello world!</b><br />Universitat
politecnica.").openPopup();
```

Asegúrese de que se llame a todo el código después de la inclusión de div y leaflet.js. Ahora tengo un mapa de Leaflets que funciona.

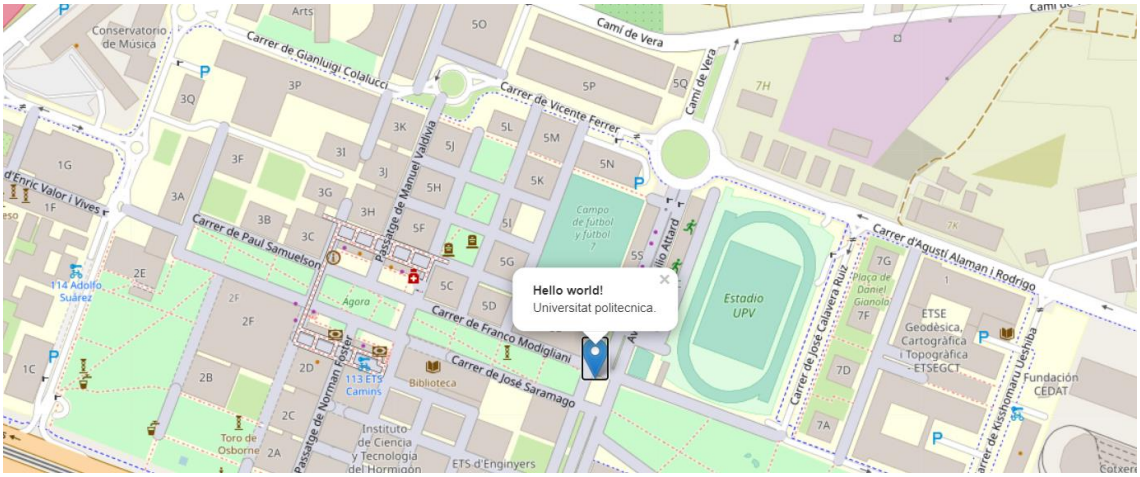

Figura 6:Diseña un mapa básico.

3.2 Usando GeoJSON con Leaflet

3.2.1 Obtener datos de GEOJSON

Como se mencionó anteriormente, podemos obtener datos a través de la API aplicada, y también podemos descargarlos directamente a través de

`http://gobiernoabierto.valencia.es/en/data/`

Contiene todos los datos e información disponibles públicamente en la ciudad de Valencia.

Figura 7:: Información de datos pública en el sitio web [8].

3.2.2 La capa GeoJSON

El objeto GeoJSON en Leaflet [9] se agrega al mapa a través de la capa GeoJSON. Podemos usar el siguiente código para crear una capa y agregarla al mapa:

```
L.geoJSON(geojsonFeature).addTo(map);
```

Si desea agregar varias capas al mapa para su visualización, se puede implementar el siguiente código:

```
var myLayer = L.geoJSON().addTo(map);
myLayer.addData(geojsonFeature);
```

3.2.2.1 onEachFeature

Simplemente agregar la capa al mapa obviamente no puede cumplir con los requisitos de visualización, por lo que necesitamos la función "onEachFeature" para llamar a la información en GeoJSON antes de agregarla a la capa GeoJSON, Puede adjuntar una ventana emergente a una función al hacer clic en la función:

```
function onEachFeature(feature, layer) {
    if (feature.properties && feature.properties.popupContent) {
        layer.bindPopup(feature.properties.popupContent);
    }
}

var geojsonFeature = {
    "type": "Feature",
    "properties": {
        "name": "tfm",
        "amenity": "universidad",
        "popupContent": "This is universitat politenica de valencia!"
    },
    "geometry": {
        "type": "Point",
        "coordinates": [39.4810070700481, -0.34123508748639353]
    }
};

L.geoJSON(geojsonFeature, {
    onEachFeature: onEachFeature
}).addTo(map);
```

3.2.2.1 Filter

Utilice la opción de Filter para controlar la visibilidad de las funciones de GeoJSON. Para ello, pasamos una función como opción de filtro. Esta función se llamará en cada entidad de la capa GeoJSON y pasará feature and layer. Luego, puede controlar la visibilidad utilizando el valor True o False en el atributo de característica.

En el siguiente ejemplo, " politécnica " no se mostrará en el mapa.

```
var algoFeatures = [{
    "type": "Feature",
    "properties": {
        "name": "universitat",
        "show_on_map": true
    },
    "geometry": {
        "type": "Point",
        "coordinates": [-105.99, 40.75]
    }
}, {
    "type": "Feature",
    "properties": {
        "name": "politécnica",
        "show_on_map": false
    },
    "geometry": {
        "type": "Point",
        "coordinates": [-104.98, 39.74]
    }
}];

L.geoJSON(algoFeatures, {
    filter: function(feature, layer) {
        return feature.properties.show_on_map;
    }
}).addTo(map);
```

3.3 Layer Groups y Layers Control

Cuando necesitamos agregar varias capas al mapa, el control y la conmutación de capas son esenciales. Podemos usar los dos pequeños controles en la librería de Leaflet para completar las operaciones anteriores: Layer Groups y Layers Control [10].

3.3.1 Layer Groups

Si necesita combinar un montón de capas en un grupo, se puede tratar como una capa en el código:

```
var arquitectura = L.marker([39.61, -105.02]).bindPopup('This is
arquitectura, CO.'),
    informática    = L.marker([39.74, -104.99]).bindPopup('This is
informática, CO.'),
    artes    = L.marker([39.73, -104.8]).bindPopup('This is artes,
CO.'),
    industria    = L.marker([39.77, -105.23]).bindPopup('This is
industria, CO.');
```

Puede usar la clase LayerGroup para realizar las siguientes operaciones en lugar de agregarlas directamente al mapa:

```
var universitat = L.layerGroup([arquitectura, informática, artes,
industria]);
```

Ahora hay una capa de universitat que combina marcadores universitarios en una capa, que se puede agregar o eliminar en el mapa.

3.3.2 Layers Control

Este control permite a los usuarios controlar las capas que ven en el mapa. Ahora, además de mostrar cómo usarlo, se mostrará otro uso conveniente de los grupos de capas.

Hay dos tipos de capas: (1) capas base mutuamente excluyentes (solo se puede ver una capa base en el mapa a la vez), como capas de mosaicos, y (2) capas de superposición, todas colocadas en Capa básica. Ahora queremos cambiar entre las dos capas base y la superposición para activar y desactivar: la marca universitaria que creamos anteriormente.

```
var universitat = L.layerGroup();

var trafico = L.layerGroup();


var map = L.map('map', {
        center: [39.4810070700481, -0.34123508748639353],
        zoom: 10,
        layers: [trafico,universitat]
```

A continuación, crea dos objetos. Uno contendrá la capa base y otro contendrá la capa superpuesta. Estos son solo objetos simples con pares clave / valor. La clave establece el texto de la capa en el control (como "Universitat") y el valor correspondiente es una referencia a la capa (como universitat).

```
    var baseLayers = {
        "tracifo": trafico,
        "Universitat":universitat
    };
    var overlays = {
        "map": map
```

Finalmente, todo lo que tiene que hacer es crear un control de capa y agregarlo al mapa. El primer parámetro que se pasa al crear el control de capas es el objeto de capas subyacentes. El segundo parámetro es el objeto de superposiciones. Ambos parámetros son opcionales: puede pasar solo el objeto de la capa base omitiendo el segundo parámetro, o pasar solo el objeto cubierto pasando null como primer parámetro.

```
    L.control.layers(baseMaps, overlayMaps).addTo(map);
```

3.4 Visualización de datos en el mapa

3.4.1 Visualización de Cámaras de tráfico

Cámaras de tráfico existentes en la ciudad de valencia.

- Tipo: Tipo de la cámara web de tráfico, superficie, paso inferior.
- Angulo: Ángulo de representación en el plano.
- Id cámara: identificador único de la cámara.
- Descripción: Descripción de la orientación de la cámara de tráfico.

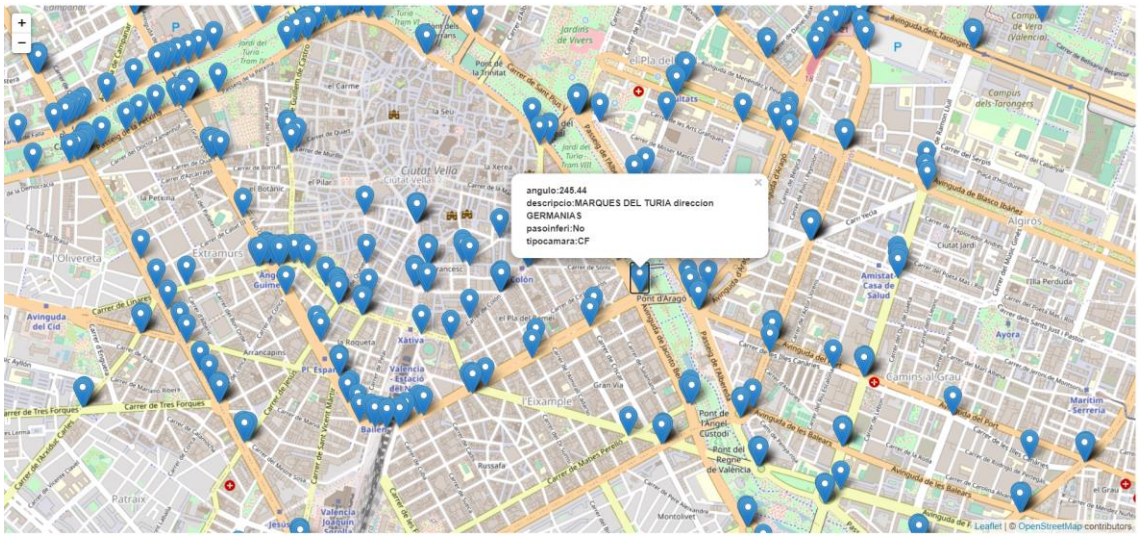

Figura 8: visualización de Cámaras de tráfico.

```
<link rel="stylesheet" href="./leaflet.css" />
<script src="./leaflet.js"> </script>>
<script src="./data.js"></script>
<style>
#mapDiv { height: 700px; }
</style>

<script>
        var valencia=[39.4810070700481, -0.34123508748639353];

        var url =
'https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}?access_token=pk.eyJ1
IjoiYmFrZXJ6aHVhbmciLCJhIjoiY2toNWlmdGN5MDhmcjJwb2Y2bmI0OWV1diJ9.Yb-
9KH21mJrpNHKRUiHY8g';

        var leafletMap = L.map('mapDiv').setView(valencia, 15);


L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(leafletMap);

       L.marker(valencia).addTo(leafletMap)
                .bindPopup("<b>Hello world!</b><br />Universitat
politecnica.").openPopup();
            //visualizar los datos//
                        for(var i=0;i<data.features.length;i++){
                L.marker([data.features[i].geometry.coordinates[1],
data.features[i].geometry.coordinates[0]]).addTo(leafletMap)
                        .bindPopup("<b>angulo:"+data.features[i].prope
rties.angulo+"</b><br />"+
                        "
<b>descripcio:"+data.features[i].properties.descripcio+"</b> <br />"+
                        "
<b>pasoinferi:"+data.features[i].properties.pasoinferi+"</b> <br />"+

"<b>tipocamara:"+data.features[i].properties.tipocamara+"</b>").openPo
pup()
        }
```

Código 1: El código de visualización de Cámaras de tráfico.

3.4.2 Visualización de Red de vigilancia de la contaminación atmosférica

Red de vigilancia de la contaminación atmosférica.

- Nombre: Nombre de la estación automática.
- Dirección: Dirección donde se encuentra la estación.
- Tipo zona: Tipo de zona, suburbana o urbana

• Tipo emisión: Fuente de emisión predominante: Fuente de emisión que mide. Fondo o Tráfico.

• Parámetros: Parámetros que mide

• Mediciones: Mediciones de todos los días del último mes.

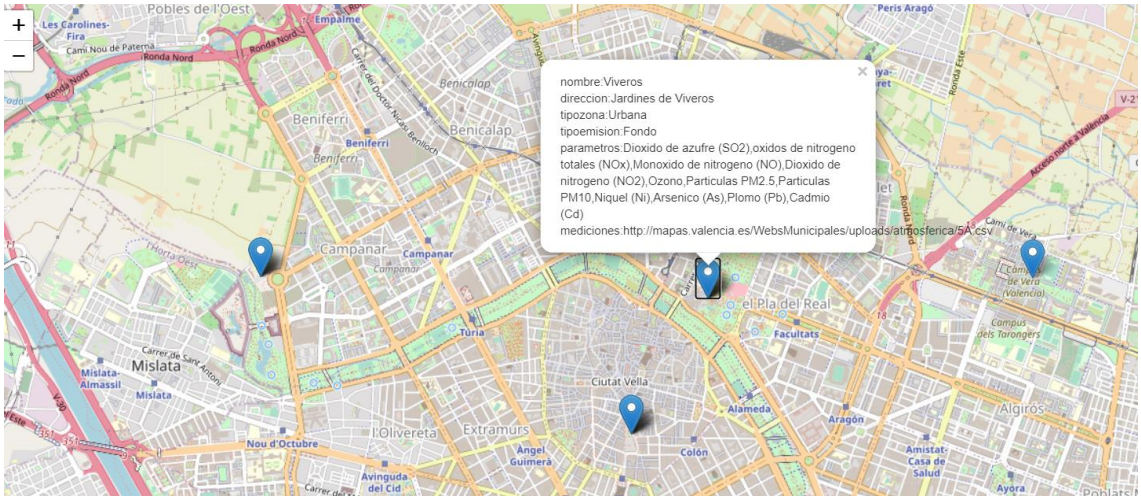

Figura 9:Visualización de Red de vigilancia de la contaminación atmosférica.

Entre los datos disponibles públicamente, se trata sólo de una contaminación. También podemos obtener una contaminación para una ubicación específica, como Datos de la estación de contaminación atmosférica de la Universidad Politécnica o Datos de la estación de contaminación atmosférica de Molí del Sol. Se utiliza para evaluar la calidad ambiental de diferentes ubicaciones.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="{CHARSET}">
        <title></title>
        <link rel="stylesheet" href="./leaflet.css"integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZM
Z19scR4PsZChSR7A==" crossorigin=""/>

   <script src="./leaflet.js"> </script>
   <script src="./leaflet.active-layers.min.js"></script>
   <script src="./leaflet.select-layers.min.js"></script>
    </head>
<style>
#mapid { height: 700px; }
</style>
    <body>
    <div id="mapid"></div>
    <script src="contaminacion.js" type="text/javascript"></script>
<script>
        var valencia=[39.4810070700481, -0.34123508748639353];
        var url =
'https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}?access_token=pk.eyJ1
IjoiYmFrZXJ6aHVhbmciLCJhIjoiY2toNWlmdGN5MDhmcjJwb2Y2bmI0OWV1diJ9.Yb-
9KH21mJrpNHKRUiHY8g';
        var leafletMap = L.map('mapid').setView(valencia, 15);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(leafletMap);
        L.marker(valencia).addTo(leafletMap)
                .bindPopup("<b>Hello world!</b><br />Universitat
politecnica.").openPopup();
                //añadir la capa al mapa//
            var Layer = L.geoJSON(geojsonFeature).addTo(leafletMap);
            Layer.addData(geojsonFeature);
        L.geoJSON(geojsonFeature).addTo(leafletMap);
     //visualizar los datos//
       L.geoJSON(geojsonFeature, {
 onEachFeature: onEachFeature
    }).addTo(leafletMap);

    function onEachFeature(feature, layer) {
        var popupContent = "";
        if (feature.properties && feature.properties.nombre) {
            popupContent +="nombre:"+feature.properties.nombre+"<br
/>direccion:"+feature.properties.direccion+"<br
/>tipozona:"+feature.properties.tipozona+"<br
/>tipoemision:"+feature.properties.tipoemision+"<br
/>parametros:"+feature.properties.parametros+"<br
/>mediciones:"+feature.properties.mediciones
        }
        layer.bindPopup(popupContent);
    }
</script>
    </body>
</html>
```

Código 2: El código de Visualización de Red de vigilancia de la contaminación atmosférica

3.4.3 Visualización de Estado tráfico tiempo real

Los datos de estado de tráfico se actualizan cada 3 minutos. En caso de que haya algún problema con la actualización de los datos aparecerá "Sin datos".

- Id tramo: Identificador único del tramo.
- Denominación: Denominación del tramo.
- Estado: Estado del tráfico en tiempo real del tramo. Los códigos de los estados del tráfico son los siguientes:
  - 0 Fluido
  - 1 Denso
  - 2 Congestionado
  - 3 Cortado

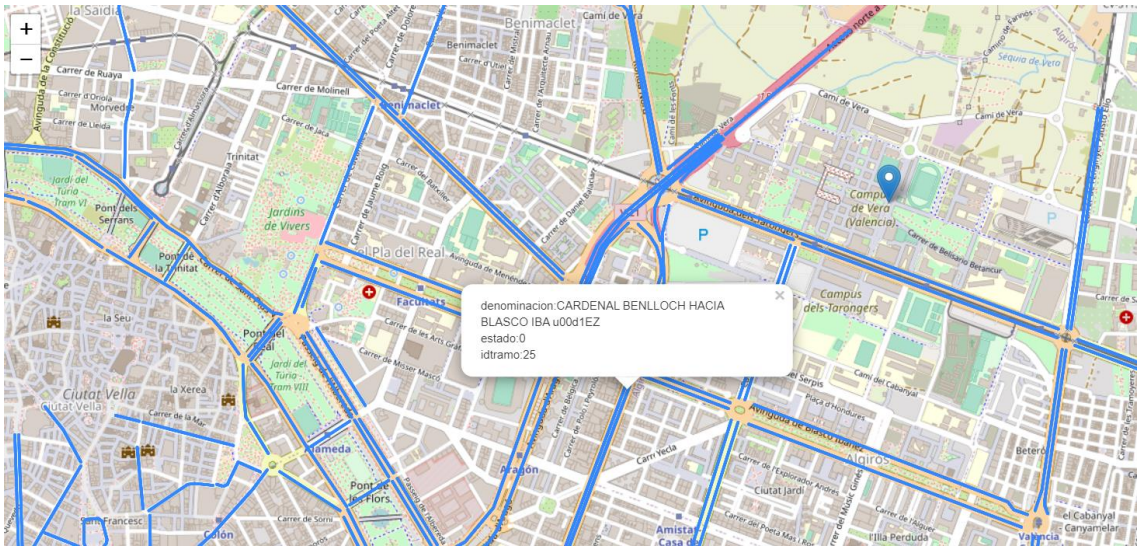

Figura 10:Visualización de Estado tráfico tiempo real.

Esta imagen se muestra los últimos datos de tráfico en tiempo real seleccionados. Para obtener visualización de datos durante un período de tiempo o más, podemos almacenar la información de datos requerida en la base de datos y llamarla cuando sea necesario.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="{CHARSET}">
        <title></title>
        <link rel="stylesheet" href="./leaflet.css"integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZM
Z19scR4PsZChSR7A==" crossorigin=""/>
   <script src="./leaflet.js"> </script>
   <script src="./leaflet.active-layers.min.js"></script>
   <script src="./leaflet.select-layers.min.js"></script>
    </head>
<style>
#mapid { height: 700px; }
</style>
    <body>
    <div id="mapid"></div>
    <script src="contaminacion.js" type="text/javascript"></script>
<script>
        var valencia=[39.4810070700481, -0.34123508748639353];
        var url =
'https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}?access_token=pk.eyJ1
IjoiYmFrZXJ6aHVhbmciLCJhIjoiY2toNWlmdGN5MDhmcjJwb2Y2bmI0OWV1diJ9.Yb-
9KH21mJrpNHKRUiHY8g';
        var leafletMap = L.map('mapid').setView(valencia, 15);


L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(leafletMap);
        L.marker(valencia).addTo(leafletMap)
                .bindPopup("<b>Hello world!</b><br />Universitat
politecnica.").openPopup();
              //añadir la capa al mapa//
           var myLayer = L.geoJSON(trafico).addTo(leafletMap);
           myLayer.addData(trafico);
        L.geoJSON(trafico).addTo(leafletMap);
            //visualizar los datos//
       L.geoJSON(geojsonFeature, {
 onEachFeature: estadotrafico
    }).addTo(leafletMap);

    function estadotrafico(feature, mylayer) {
        var trafico = "";

        if (feature.properties && feature.properties.denominacion) {

trafico+="denominacion:"+feature.properties.denominacion+"<br
/>estado:"+feature.properties.estado+"<br
/>idtramo:"+feature.properties.idtramo

        }

        mylayer.bindPopup(trafico);
</script>
    </body>
</html>
```

Código3: El código de Visualización de Estado tráfico tiempo real

3.4.4 Visualización de Quioscos de prensa distribuidos por la ciudad

Quioscos de prensa distribuidos por la ciudad

- Emplaza mie: Emplazamiento del Quiosco.
- Modelo: Modelo del Quiosco.

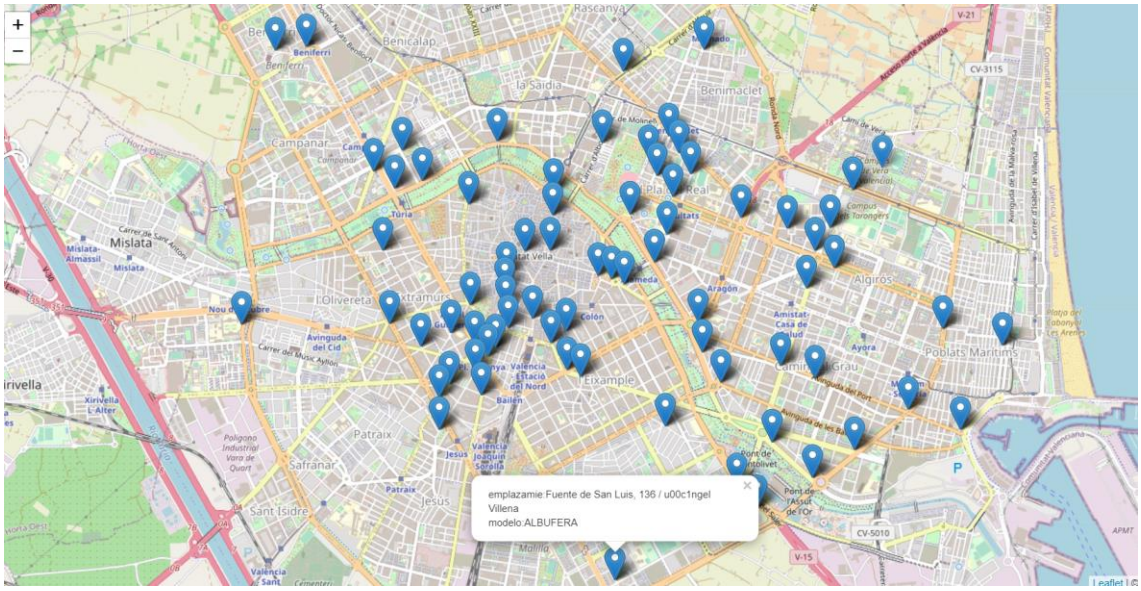

Figura 11:Visualización de Quioscos de prensa distribuidos por la ciudad.

Podemos encontrar a través de información visual que la mayoría de los Quioscos de prensa se encuentran en el centro de la ciudad. Quizás podamos sugerir la instalación de más quioscos en áreas remotas para atender a más personas.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="{CHARSET}">
        <title></title>
        <link rel="stylesheet" href="./leaflet.css"integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZM
Z19scR4PsZChSR7A==" crossorigin=""/>
   <script src="./leaflet.js"> </script>
   <script src="./leaflet.active-layers.min.js"></script>
   <script src="./leaflet.select-layers.min.js"></script>
    </head>
<style>
#mapid { height: 700px; }
</style>
    <body>
    <div id="mapid"></div>
    <script src="contaminacion.js" type="text/javascript"></script>
<script>
        var valencia=[39.4810070700481, -0.34123508748639353];
        var url =
'https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}?access_token=pk.eyJ1
IjoiYmFrZXJ6aHVhbmciLCJhIjoiY2toNWlmdGN5MDhmcjJwb2Y2bmI0OWV1diJ9.Yb-
9KH21mJrpNHKRUiHY8g';
        var leafletMap = L.map('mapid').setView(valencia, 15);


L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(leafletMap);
        L.marker(valencia).addTo(leafletMap)
                .bindPopup("<b>Hello world!</b><br />Universitat
politecnica.").openPopup();
                    //añadir la capa al mapa//
            var myLayer = L.geoJSON(quio).addTo(leafletMap);
            myLayer.addData(quio);
              //visualizar los datos//
        L.geoJSON(quio, {
 onEachFeature: quiosco
    }).addTo(leafletMap);
    function quiosco(feature, mylayer) {
        var quio = "";

        if (feature.properties && feature.properties.emplazamie) {
            quio+="emplazamie:"+feature.properties.emplazamie+"<br
/>modelo:"+feature.properties.modelo

        }

        mylayer.bindPopup(quio);
</script>
    </body>
</html>
```

Código 4: El código de Visualización de Quioscos de prensa distribuidos por la ciudad

. 3.4.5 Visualización de Mapa Polen Casuarina

Mapa del polen agrupado por varias especies arbóreas. (Casuarina)

- Densidad: densidad de polen por zona.

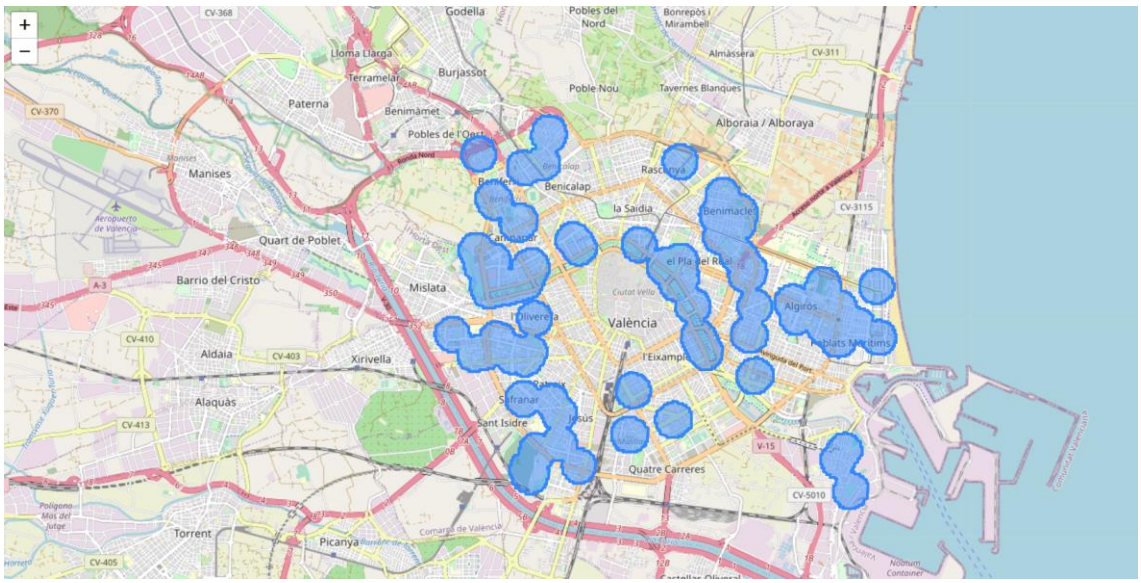

Figura 12: El código de Visualización de Mapa Polen Casuarina.

También podemos visualizar en el mapa en forma de cobertura. Como se muestra en la imagen de arriba, se ha elegido la densidad de una planta llamada Polen Casuarina en el casco urbano de Valencia. Tomando esto como ejemplo, también podemos mostrar claramente la distribución específica de varias otras plantas en el mapa, lo que puede mejorar la planificación de la plantación.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="{CHARSET}">
        <title></title>
        <link rel="stylesheet" href="./leaflet.css"integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZM
Z19scR4PsZChSR7A==" crossorigin=""/>
   <script src="./leaflet.js"> </script>
   <script src="./leaflet.active-layers.min.js"></script>
   <script src="./leaflet.select-layers.min.js"></script>
    </head>
<style>
#mapid { height: 700px; }
</style>
    <body>
    <div id="mapid"></div>
    <script src="polen.js" type="text/javascript"></script>
<script>
        var valencia=[39.4810070700481, -0.34123508748639353];
        var url =
'https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}?access_token=pk.eyJ1
IjoiYmFrZXJ6aHVhbmciLCJhIjoiY2toNWlmdGN5MDhmcjJwb2Y2bmI0OWV1diJ9.Yb-
9KH21mJrpNHKRUiHY8g';
        var leafletMap = L.map('mapid').setView(valencia, 15);


L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(leafletMap);
        L.marker(valencia).addTo(leafletMap)
                .bindPopup("<b>Hello world!</b><br />Universitat
politecnica.").openPopup();
             //añadir la capa al mapa//
           var polLayer = L.geoJSON(quio).addTo(leafletMap);
           myLayer.addData(po);
             //visualizar los datos//
        L.geoJSON(po, {
 onEachFeature: polen
    }).addTo(map);
        function polen(feature, pollayer) {
        var polenc = "";

        if (feature.properties && feature.properties.densidad) {
            polenc+="densidad:"+feature.properties.densidad

        }

        pollayer.bindPopup(polenc);
    }
</script>
    </body>
</html>
```

Código 5: El código de Visualización de Mapa Polen Casuarina.

. 3.4.6 Control de visualización de mapas

Una vez que hayamos creado todas las capas de datos necesarias, cómo integrar y controlar las capas es un punto extremadamente importante en la visualización de datos. En este paso, usamos los dos complementos de control de Layer Groups y Layers Control mencionados anteriormente para completar el trabajo que debe completarse. El efecto específico se muestra en las siguientes figuras:

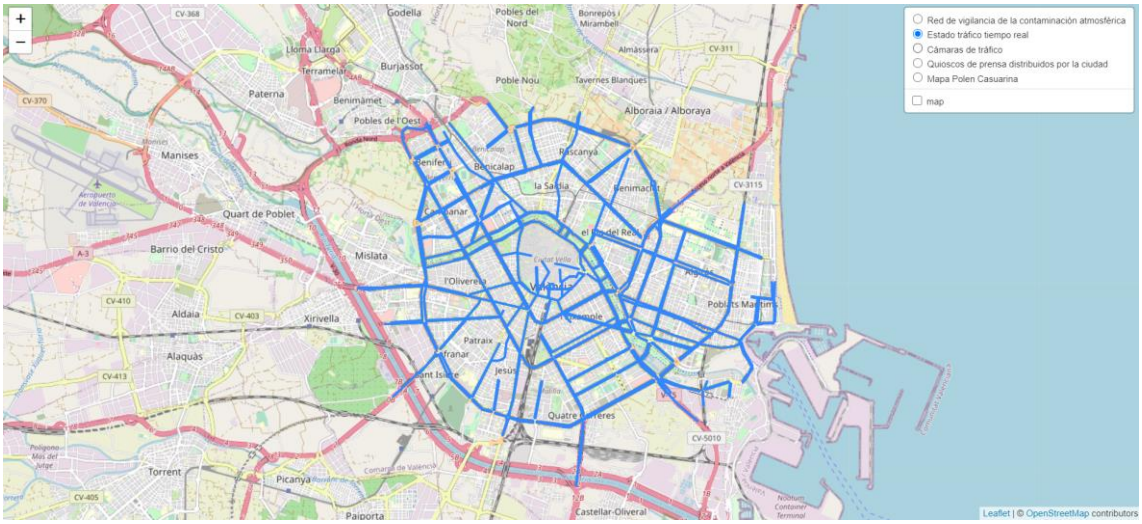

Figura 13: Capa de Estado tráfico en tiempo real.

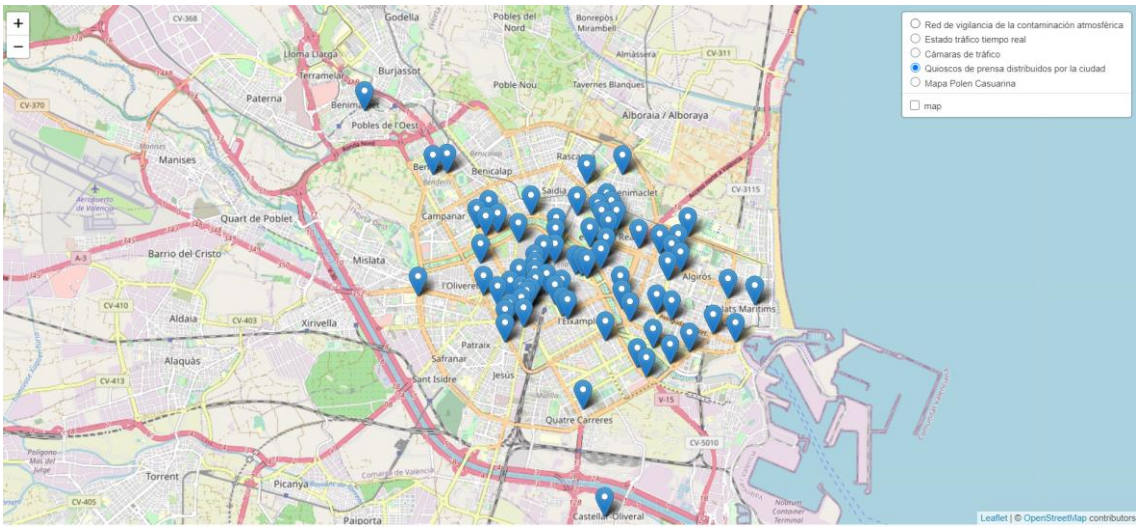

Figura 14: Capa de Quioscos de prensa distribuidos por la ciudad.

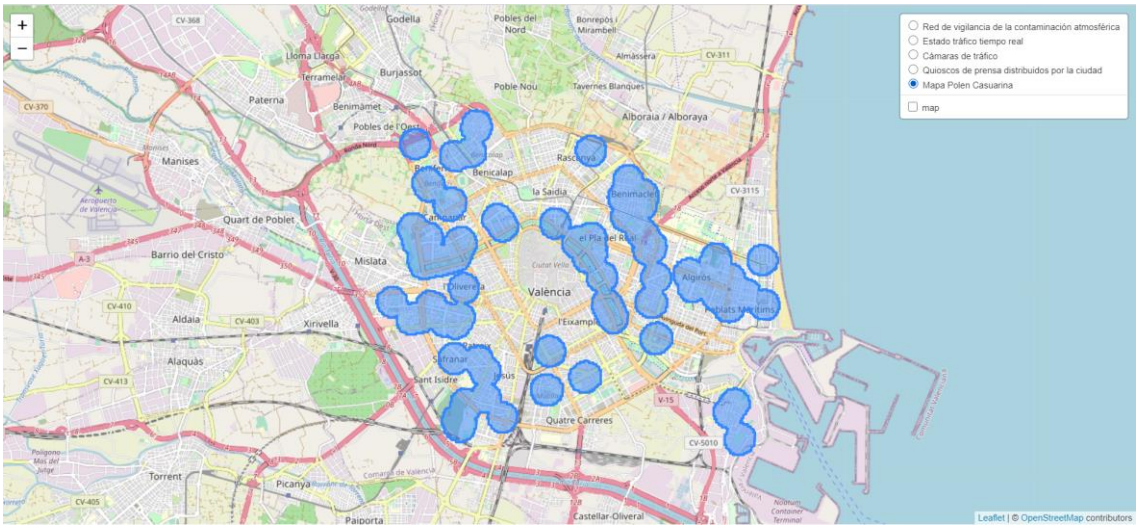

Figura 15: Capa de Mapa Polen Casuarina.

Además, se añadió una capa en blanco. Cuando se selecciona, la información de otras capas ya no se mostrará en el mapa y el usuario puede restaurar la apariencia original del mapa.

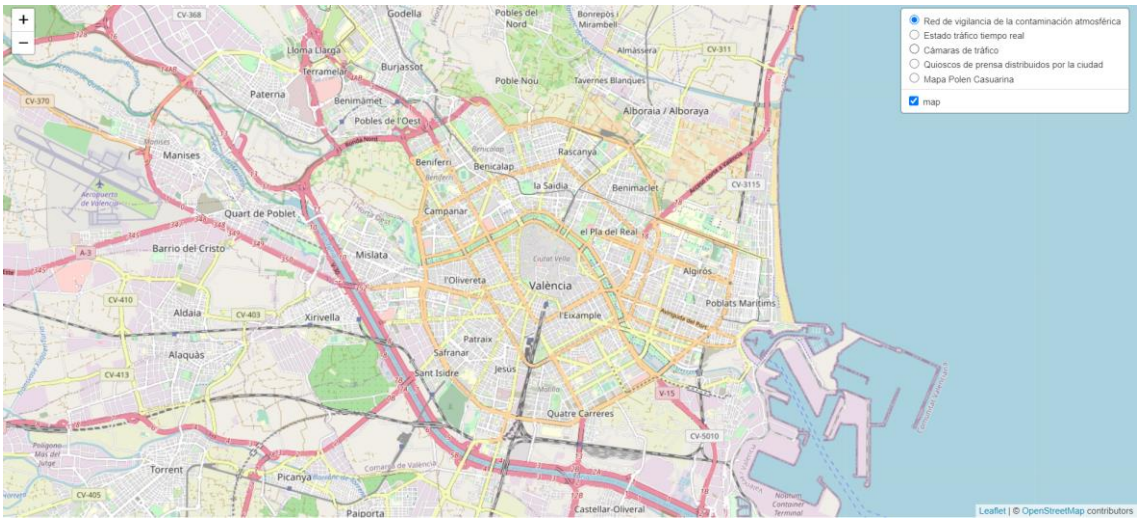

Figura 16: Restaurado el mapa.

```
<script>
    var cities = L.layerGroup();

    var trafico2 = L.layerGroup();

    var  camara = L.layerGroup();

    var qui = L.layerGroup();

    var pol = L.layerGroup();

   /* var pan = L.layerGroup();**/
var map = L.map('map', {
        center: [39.4810070700481, -0.34123508748639353],
        zoom: 10,
        layers: [trafico2,cities,camara,qui,pol]
    });
         //Crear grupo de capas//
        var myLayer = L.geoJSON(trafico).addTo(cities);
           myLayer.addData(trafico);

        var Layer = L.geoJSON(geojsonFeature).addTo(trafico2);
           Layer.addData(geojsonFeature);

        var quiLayer = L.geoJSON(quio).addTo(qui);
           quiLayer.addData(quio);

        var polLayer = L.geoJSON(po).addTo(pol);
           polLayer.addData(po);

        var camaraLayer = L.geoJSON(camadata).addTo(camara);
           camaraLayer.addData(camadata);
           //añadir las capas al mapa//
        L.geoJSON(geojsonFeature).addTo(trafico2);
        L.geoJSON(trafico).addTo(cities);
        L.geoJSON(quio).addTo(qui);
        L.geoJSON(po).addTo(pol);
        L.geoJSON(camadata).addTo(camara);

   L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(map);
          //Controla y cambia entre diferentes capas//
    var baseLayers = {
        "Red de vigilancia de la contaminación atmosférica": trafico2,
        "Estado tráfico tiempo real": cities,
        "Cámaras de tráfico":camara,
        "Quioscos de prensa distribuidos por la ciudad":qui,
        "Mapa Polen Casuarina":pol,

    };
    var overlays = {
        "map": map
    };
    L.control.layers(baseLayers,overlays).addTo(map);
</script>
```

Código 6: El código de Control de visualización de mapas

4. Capítulo 4

4.1 Conclusiones

En este proyecto, se ha estudiado las herramientas, para visualizar e integrar datos de sensores básicos e información en la Web, y utiliza API para obtener datos y librería (Leaflet) para simplificar la operación de visualización en el mapa y la aplicación de filtros de agregación. Para html, CSS, Javascript, estos lenguajes de programación relacionados con la web tienen una comprensión y un conocimiento relativamente profundos.

Además se ha utilizados el API, Georeferenced Data, la cual permite obtener las informaciones del portal de datos abiertos de la ciudad de Valencia, ya que forma parte de el API privada proporcionada por la ciudad de Valencia. Por lo tanto, puede seguir utilizando otras API disponibles para investigar y es una dirección para seguir aprendiendo. En cuanto a la librería de folletos, la mayoría de las funciones básicas que se utilizan en este artículo son sus funciones básicas, y hay muchos más complementos y funciones avanzadas que se pueden ampliar.

Se ha desarrollado, una plataforma de visualización de datos, entre los cuales se encuentran, estado del trafico de la ciudad, la contaminación, densidad de polen por zonas etc. Estos datos son obtenidos del la pagina del ayuntamiento, y almacenados en una base de datos local con ayuda de InfluxDB.

4.2 Trabajos futuros

Dado que ya existen muchas plataformas o sitios web que pueden realizar la transmisión de datos en línea y la visualización del Internet de las Cosas, este diseño se puede ampliar según las funciones de estas plataformas. Por ejemplo, se puede usar una base de datos para almacenar y administrar la información de datos obtenida y, como visualización de una interfaz web, se pueden desarrollar más opciones funcionales en la interfaz de usuario, etc.

5. Lista de Acrónimos

API application programming interface

CSL C standard library

GIS geographic information system

HTTP Hypertext Transfer Protocol

IBM International Business Machines Corporation

IOT Internet of things

OS Operating System

SQL Structured Query Language

ITU International Telecommunication Union

6. Bibliografía

[1] V. Agafonkin., «an open-source JavaScript library, » [En línea]. Available: https://leafletjs.com/. [Último acceso: 5 09 2020].

[2] S. Xiaochen, "Research on Copyright Protection and Restriction of Application Program Interface," [Online]. Available: https://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD201801&filename=1017284173.nh&uid=WEEvREcwSlJHSldRa1FhdXNXaEd2OXlvOUJCdkFKMmN5ZVhJMnpJeG9Kbz0=$9A4hF_YAuvQ5obgVAqNKPCYcEjKensW4IQMovwHtwkF4VYPoHbKxJw!!&v=MjIwMjhHdERMckpFYlBJUjhlWDFMdXhZU. [Accessed 12 09 2020].

[3] image.baidu, «image API,» [En línea]. Available: https://image.baidu.com/search/index?tn=baiduimage&ipn=r&ct=201326592&cl=2&lm=-1&st=-1&fm=index&fr=&hs=0&xthttps=000000&sf=1&fmq=&pv=&ic=0&nc=1&z=&se=1&showtab=0&fb=0&width=&height=&face=0&istype=2&ie=utf-8&word=api&oq=api&rsp=-1. [Último acceso: 13 09 2020].

[4] a. d. valencia, «Solicitud de uso de la API,» [En línea]. Available: http://www.valencia.es/ayuntamiento/DatosAbiertos.nsf/web_fApi?ReadForm&lang=1&nivel=3&seccion=1. [Último acceso: 13 09 2020].

[5] changhr2013, «GeoJSON format specification,» [En línea]. Available: https://www.jianshu.com/p/5c6c6e76d4df. [Último acceso: 15 09 2020].

[6] OSCHINA(OSChina.NET), «HBuilder HTML5 Web Development IDE, » [En línea]. Available: https://www.oschina.net/p/hbuilder?hmsr=aladdin1e1. [Último acceso: 15 09 2020].

[7] V. Agafonkin, «Using GeoJSON with Leaflet, » [En línea]. Available: https://leafletjs.com/examples/geojson/. [Último acceso: 20 09 2020].

[8] V. Agafonkin., «Layer Groups and Layers Control, » [En línea]. Available: https://leafletjs.com/examples/layers-control/. [Último acceso: 22 09 2020].

[9] a. d. valencia, «API AppCiudad Valencia,» [En línea]. Available: http://www.valencia.es/ayuntamiento/DatosAbiertos.nsf/0/2113BD9D1693D7EAC1257C6600449981/$FILE/API%20APPCIUDAD%20v3.pdf?OpenElement&lang=1. [Último acceso: 14 09 2020].

[10] a. d. valencia, «Vados,» [En línea]. Available: http://gobiernoabierto.valencia.es/en/dataset/?id=vados. [Último acceso: 18 09 2020]