



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA,
CARTOGRÁFICA Y TOPOGRÁFICA

---

# Creación de un repositorio de contenidos para transporte público urbano con Fiware

---

**Autor:**
Jose Luis Pastor Alemany

**Tutor:**
Ángel Marqués Mateu

Grado en Ingeniería Geomática y Topografía

Valencia, Septiembre 2019

# Compromiso

"El presente documento ha sido realizado completamente por el firmante; no ha sido entregado como otro trabajo académico previo y todo el material tomado de otras fuentes ha sido convenientemente entrecomillado y citado su origen en el texto, así como referenciado en la bibliografía"

# Agradecimientos

A mí familia y amigos por apoyarme en todas mis decisiones. A ellos les debo lo que soy.

A mi tutor de TFG Ángel Marqués por su ayuda y su guía durante la elaboración del proyecto.

A María Perpiña por ayudarme a dar forma con LaTex a este proyecto.

A Rubén Garrigues, el mejor amigo que la carera me ha dado.

# Resumen

A lo largo de este trabajo se expondrá la importancia que tienen las Smart Cities en la vida cotidiana de las personas, sus diferentes definiciones y dimensiones, así como los retos a los que se enfrenta cada ciudad para poder ser considerada como "Smart City".

Se abordará también lo que supone el uso de las Tecnologías de la Información y las Comunicaciones, más conocida por sus siglas TIC. Estas tecnologías nos ayudarán en cierta manera a monitorizar el rendimiento de la ciudad y saber cómo poder tratar toda la información recibida para que la urbe sea más eficiente en todos los campos.

Un aspecto fundamental de este proyecto es la carga masiva de datos GTFS (*Google Transit Feed Specification*,actualmente conocido como General Transit Feed Specification a la plataforma de datos abiertos Fiware. Para lograrlo, se han requerido el uso de diversos entornos de computación como Python, MongoDB, Docker; los cuales han permito la creación de repositorios en los servidores Fiware para posteriormente, mediante una serie de códigos, cargar los datos obtenidos del portal de datos abiertos del Ayuntamiento de Valencia. Como resultado de todo el trabajo se han cargado en el repositorio `http://137.74.44.56:1026/v2/entities` datos relacionados con la EMT de Valencia como pueden ser:

- GtfsStops (paradas)
- GtfsRoutes (rutas de las distintas líneas)
- GtfsAgency (datos sobre la agencia)
- GtfTrips (define los viajes de cada ruta)
- GtfsStop_time (los horarios de una parada determinada)

Una vez cargados los datos en la plataforma, estos podrán ser usados por diferentes usuarios de Fiware u otros entornos para diversos fines. Uno de

esos fines podría ser la creación de una aplicación que permita ver al usuario final, cual de todas las paradas que tiene a su alrededor o en un radio cercano es la más cercana. Más adelante también se podría incluir en el paquete de datos qué ruta es la más rápida para llegar de un punto a otro o incluso la ruta más verde, es decir, la más ecológica en cuanto al desplazamiento.

# Índice general

# Índice de figuras

# Parte I

# Memoria

# Capítulo 1

# Smart Cities

Desde comienzos de la civilización, la humanidad se ha concentrado en asentamientos. Estos asentamientos poco a poco fueron creciendo en tamaño y en importancia con un área de influencia cada vez más grande hasta llegar a formar grandes imperios desde donde se organizaban las fuerzas militares, se concentraba el poder político y se llevaba a cabo la administración.

Las ciudades siempre han jugado un papel importante en la vida de las personas. El número de población que vive en las ciudades no ha dejado de crecer desde la primera revolución industrial en Europa y América del Norte y durante las últimas décadas en países como China e India. En España el 70 % de la población vive en un entorno urbano de más de 50.000 habitantes. (*Figura 1.1*)

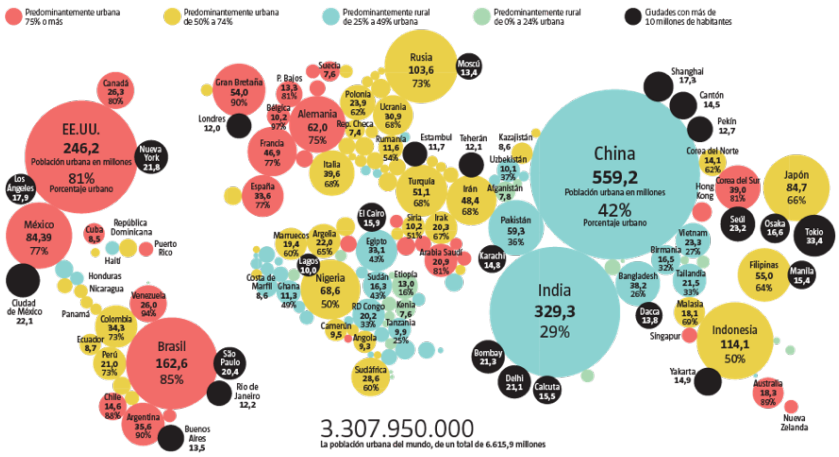


Figura 1.1: Porcentaje de la población viviendo en un entorno urbano. Fuente: La Vanguardia, 2017

Según los últimos informes de la ONU [12] para el año 2050 se llegará a concentrar en los entornos urbanos el 70 % de la población mundial, lo que supondrá que más de 6.000 millones de personas convivirán en ciudades. Este aumento de la población afectará de sobremanera al rendimiento ambiental de estas.

En este punto es donde entra en juego el papel de las llamadas Smart Cities. No debemos pensar que una Smart City es una ciudad llena de tecnología que nos soluciona la vida. Una Smart City es una ciudad que afronta los nuevos retos de las ciudades, como los enlaces de transporte entre diferentes partes de la ciudad, ciudades satélites, una alta calidad en los servicios urbanos o la correcta gestión de los residuos, por poner unos ejemplos.

## 1.1 Concepto de Smart City

Durante las últimas dos décadas los expertos no han sabido consensuar una única definición para explicar qué es y en qué consiste una Smart City. El término en sí abarca una amplia gama de subtérminos que van asociados a las propias dimensiones que puede tener una Smart City.

Harrison, C. [17], describió una Smart City como una ciudad instrumentada, interconectada e inteligente.

- Instrumentada, ya que tenía que ser capaz de obtener e integrar datos del mundo real constantemente con el uso de sensores.
- Interconectada, para poder integrar todo el volumen de datos obtenido en una plataforma que permitiera la comunicación con los distintos servicios urbanos de una ciudad.
- Inteligente, porque debe ser capaz de incluir análisis complejos, modelizaciones, optimizaciones y tener gran visualización para que los servicios urbanos tomen unas decisiones operacionales propicias.

Otra visión de lo que es una Smart City viene de la mano de Nam, T. y Pardo, T. A.[20] en la *Conference on Research in Digital Goverment*. La visión general sobre lo que es una Smart City queda muy clara con el mapa conceptual de la *Figura 1.2*.

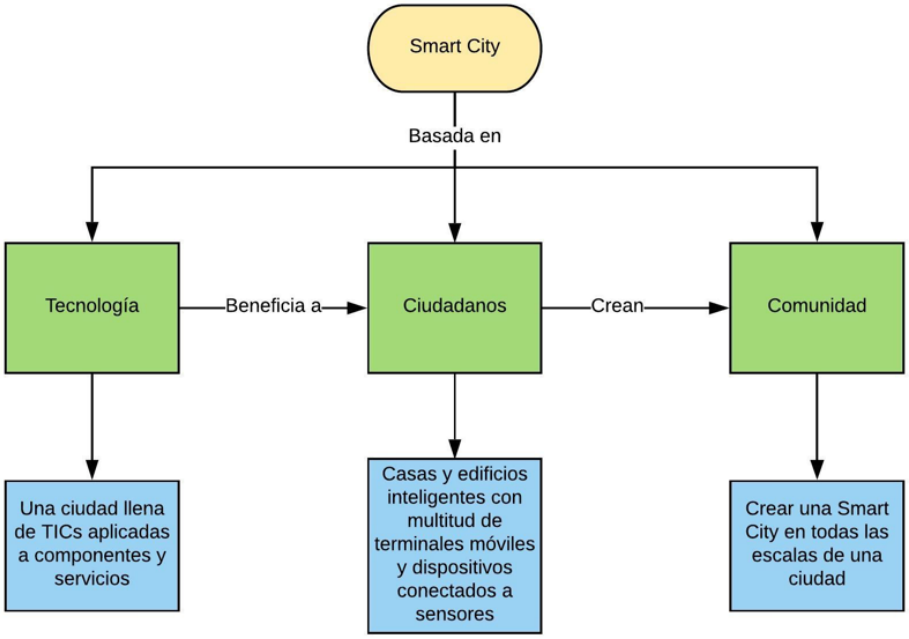


Figura 1.2: Mapa conceptual según la visión de Nam y Pardo de cómo se debe organizar una Smart City.

Aunque la tecnología no lo es todo en una Smart City, los componentes tecnológicos son la clave para los conceptos que se engloban en este proyecto. La idea principal no es que las personas adapten su vida a las exigencias tecnológicas, sino lo que se busca es cómo las personas pueden adaptar la tecnología a sus necesidades mejorando así su nivel de vida y el de la comunidad.

Hay otros términos referidos al concepto Smart City que son menos inclusivos con ciertos niveles de una ciudad [6]:

- Ciudad Digital: Ciudad interconectada que combina infraestructuras de comunicaciones para satisfacer las necesidades del gobierno local, ciudadanos y empresarios.

- Ciudad Inteligente: Usan la información tecnológica para transformar la vida y el trabajo en una urbe. Esto implica un aprendizaje, desarrollo tecnológico e innovación en las ciudades donde las personas no están incluidas.

- Ciudad Virtual: Concepto híbrido que consiste en una ciudad real y otra virtual creando así un ciberespacio.

Todas estas ideas de cómo debería ser una ciudad no tiene como protagonista al ciudadano de a pie como lo tiene el concepto Smart City.

## 1.2 Dimensiones de una Smart City

Como se ha visto en el apartado anterior, la idea de Smart City es muy amplia y conviene definir bien cuáles son sus dimensiones. Según *Libro blanco smart cities* [11], una Smart City consta de unos elementos fundamentales, que son:

- Espacios urbanos
- Sistemas de infraestructuras
- Complejo de redes y plataforma inteligentes
- Ciudadanía que ejerza como eje vertebrador

Todos ellos se deben de gestionar de la manera más eficiente en todas y cada una de las áreas de una ciudad. Estas buenas gestiones llevarán a la creación de nuevos aspectos de una Smart City relacionados con el día a día de una ciudad.

| Components of a smart city | Related aspect of urban life |
|---|---|
| smart economy | Industry |
| smart people | education |
| smart governance | e-democracy |
| smart mobility | logistics & infrastructures |
| smart environment | efficiency & sustainability |
| smart living | security & quality |

Figura 1.3: Conceptos de una Smart City y sus aspectos relacionados. Fuente: *Modelling the Smart City Performance* (Lombardi 2012)

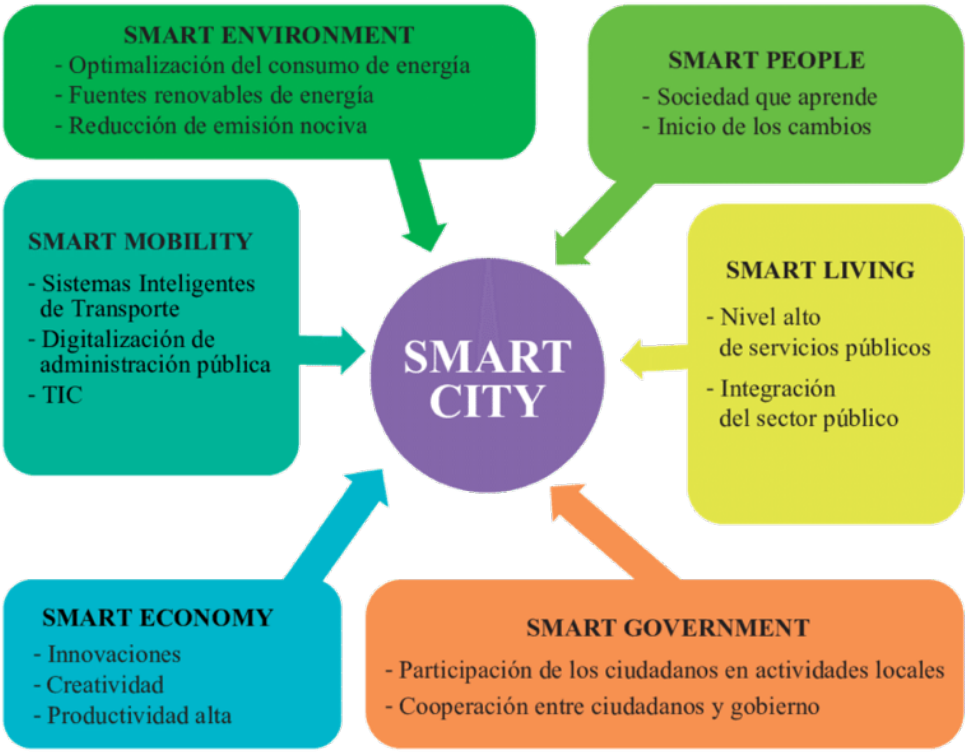


Figura 1.4: Conceptos de una Smart City. Fuente: Dorota Sikora Fernández

Los conceptos representados en las *Figuras 1.3 y 1.4* tendrán como objeto la mejora de la vida diaria de la ciudadanía en todos los ámbitos posibles. La política de los gobiernos locales para adaptar su ciudad al nuevo concepto tiene que ser clara con unos objetivos bien definidos a corto y largo plazo. No hay planes generales trazados que se puedan aplicar a todas las ciudades por igual. Por ello, cada ciudad tiene sus características y necesidades; y no se debe limitar el rango de actuación a las grandes agrupaciones metropolitanas, sino que en un entorno rural también permitirá el avance tecnológico y una mejor adaptación a las necesidades de los ciudadanos.

## 1.3 El uso de las TIC

Las Tecnologías de la información y las Comunicaciones o también conocidas por sus siglas como TIC, son herramientas tecnológicas que permite mejorar la calidad de las infraestructuras de una ciudad. Esta mejora se produce con la obtención de datos mediante la monitorización de las distintas

áreas de la urbe como pueda ser la recogida de residuos orgánicos, el flujo del transporte público o la actividad de los semáforos.

El uso de las TIC se extiende a los términos de la innovación, sostenibilidad y de la mejora de la calidad de vida. Es un recurso transversal donde su buena gestión e implantación llevará al crecimiento de la Smart City.

Según el *Libro blanco Smart Cities* [11] los ejes principales de las TIC son una administración electrónica que actue como nexo de unión entre el gobierno local y el ciudadano, la digitalización de la información obtenida mediante las TIC, la modernización administrativa para poder trabajar y procesar los datos proporcionados por las TIC y, por último, la interoperabilidad entre los distintos servicios digitales *Figura 1.5*.

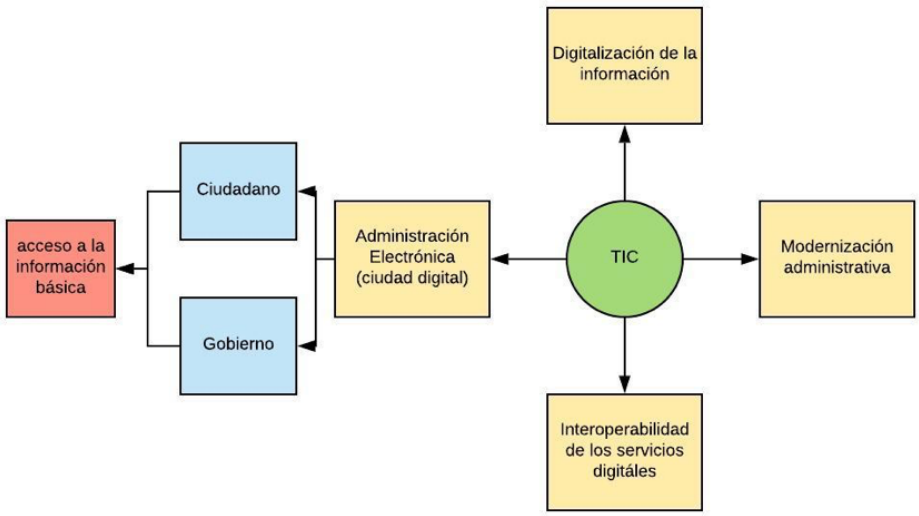


Figura 1.5: Mapa conceptual sobre los ejes principales de las TIC.

Las administraciones públicas tienen que ser la entidad dinamizadora de las TIC de cara al ciudadano de a pie al mismo tiempo que ser su principal usuaria. Las Smart Cities comienzan desde el capital humano y los gobiernos no deben caer en el pensamiento que las TIC pueden crear automáticamente todos los aspectos que una Smart City conlleva. Para ello, sería conveniente la creación de unas herramientas como pueden ser:

- Portales multiacceso.
- Smart Cards, donde se tenga el acceso a diferentes servicios de la ciudad.

- Servicios telefónicos.
- Puntos municipales inalámbricos de conexión wifi.
- Sensores que nos permitan tener monitorizada aquellas áreas de interés de una ciudad.
- Información en tiempo real de lo que está sucediendo.

En el artículo *Networked society city index*[18] se expuso las TIC como un elemento fundamental en las últimas innovaciones así como el mayor contribuidor a la globalización. Es por eso que se debe promulgar su buena gobernanza y saber usarlas de la manera apropiada para que no solo repercuta en el modelo socioeconómico de la ciudad, sino también en el medioambiental.

El objeto de este Trabajo de Fin de Grado es aportar una mejora en el ámbito de la smart mobility con la carga masiva de datos a la plataforma Fiware para que otro usuarios con un acceso fácil basado en protocolos estándar. La aportación se llevará a cabo a través de programas y códigos que conecten el PC que se use con el servidor web de Fiware, en este caso, el url `http://137.74.44.56:1026/v2/entities`. También se procederá a la creación de un índice espacial para una mejor optimización en los tiempos de computación para la búsqueda de paradas cercanas.

# Capítulo 2

# *Softwares* utilizados

## 2.1 Docker

Docker es una herramienta open source que ayuda a crear, desplegar y ejecutar aplicaciones en forma de contenedor. Estos contienen las aplicaciones que el desarrollador crea y las aísla del sistema donde se ejecuta. Sus elementos se detallan a continuación.

### 2.1.1 Contenedores

Estos contenedores son directorios que se pueden compartir y ejecutar en diferentes plataformas y máquinas virtuales. El contenido de estos contenedores es:

- Librerías de los sistemas operativos donde se ejecute.
- Herramientas del sistema.
- Programas ejecutables, que nos permitirá ejecutar la aplicación dentro del contenedor.

¿Cuáles son las ventajas de este método de contenedores?

- Solo es necesario la programación de la aplicación una sola vez debido a que la app se ejecuta en contenedores con cualquier sistema operativo que tenga instalado Doker.

- Mayor consistencia entre los entornos de prueba y los entornos de producción. Los entornos de prueba que usan los programadores son idénticos al entorno donde se ejecutará el software.

- Mayor modularidad. Esta basado en microservicios, es decir, sin llegar a cambiar la arquitectura de la aplicación, los distintos elementos de la app se pueden ejecutar en diferentes contenedores y así reduciendo la complejidad.

### 2.1.2 Imágenes

Las imágenes contienen distintas capas del sistema como la distribución, diferentes softwares, librerías y la personalización. Estás imágenes sirven como plantillas para la crear contenedores y hacer los cambios necesarios en él, pero nunca en la imágen.

Se puede añadir una nueva capa, llamada capa fina en la parte de arriba de la imagen principal. Se conoce como capa contenedor y todos los cambios realizados durante la ejecución se esribirán en esta capa.*Figura 2.1*

Los contenedores creados a partir de la imagen se pueden convertir en una imagen mediante las utilidades de docker. Las imágenes creadas se pueden transferir a otros servidores y hacer las copias de seguridad necesarias.

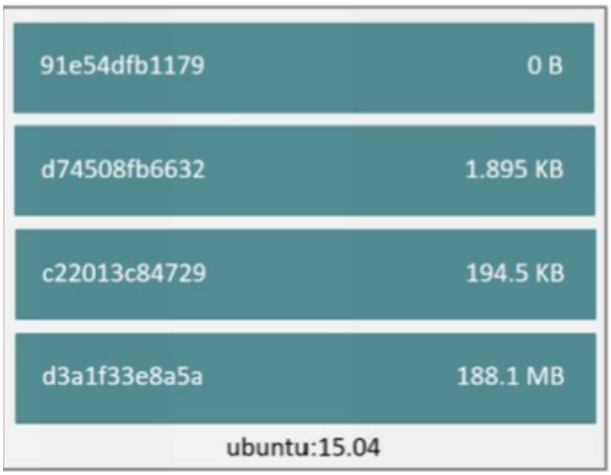


Figura 2.1: Imagen Docker con las diferentes capas de datos. Fuente: *The comlete guide to starting with Docker.*

## 2.2 Fiware

La comunidad Fiware es una comunidad independiente cuyos miembros materializan su misión, que es: *Crear un ecosistema sostenible y abierto alrededor de una plataforma pública, libre y un software impulsado por la implementación que facilitará el desarrollo de aplicaciones Smart en múltiples sectores.*

Los principios fundamentales de esta plataforma son la independencia en la toma de decisiones, la apertura, la transparencia y la meritocracia. La estructura de Fiware fomenta todo tipo de contribución y proporciona las garantías contra la pérdida de equilibrio entre los distintos miembros de la comunidad.

Está basado en tecnologías abiertas como Docker, MongoDB y HTTP. Tanto Docker como MongoDB sirven para la administración de los datos. El primero, para una gestión de contenedores que almacenan los datos complejos y el segundo para la persistencia de los datos. Por último, el protocolo HTTP nos sirve para la gestión, actualización y eliminación de los datos en Internet. Para el paso anterior se usa la herramienta del sistema operativo Linux llamada cURL. Esta herramienta permite la transferencia de archivos desde un servidor a otro, desde la línea de órdenes del sistema operativo. cURL es una buena opción para dar los primeros pasos en el entorno Fiware, sin embargo, para operaciones masivas de carga o actualización es recomendable usar un entorno de programación con mayores prestaciones en la automatización de tareas. Para la realización de este proyecto ese entorno de programación es el software Python 3.7.2

Para lograr una posición a nivel global, Fiware necesita adaptarse a escala global. Se está siguiendo un enfoque impulsado por el mercado respaldado por sus miembros y basándose en la madurez de las diferentes regiones del mercado a medida que surgen nuevas oportunidades. Los diferentes continentes están en constante colaboración con compañías como Orange, Telefónica y Platinum[4].

## 2.3 Python 3.7.2

Python es un lenguaje de programación que surgió a finales de los años 80 y que está administrado por Python Software Foundation. Su licencia es de código abierto por lo que resulta compatible con muchos programas informáticos que necesiten una interfaz programable.

Python tiene un tipo de programación multiparadigma, es decir, permite varios estilos como pueda ser a objetos, imperativa y funcional. El lenguaje proporciona dos tipos de datos que son especialmente adecuados para el desarrollo del presente proyecto: las listas y los diccionarios.

### 2.3.1 Listas

*Las listas son un tipo de colección ordenada. Pueden contener cualquier tipo de dato: números, cadenas, booleanos y listas* [10]

Ejemplo:

```
L1= [10,10,Dr. Lluch - Mare de Déu del Sufragi,C DOCTOR LLUCH
99 (DAVANT) - VALÈNCIA,39.4672336950355,-0.328305576746874,,,0,]xt
```

Dentro de la lista que se ha creado anteriormente se han añadido varios elementos. Cada elemento está indexado con una posición en la lista, siendo el primer elemento la posición L1[0] y el último la tercera.

### 2.3.2 Diccionarios

*Son colecciones que relacionan una clave y un valor. Como clave se puede utilizar números, cadenas, booleanos, tuplas,... pero no listas o diccionarios.* González Duque,R.(2008). "Colecciones" en Python para todos, editores Titivillus.

Ejemplo:


```
{
  "id": "urn:ngsi-ld:GtfsAgency:EMT",
  "type": "GtfsAgency",
  "name": {
    "value": "EMT Valencia"
  },
  "page": {
    "value": "http://www.emtvalencia.es"
  },
  "timezone": {
    "value": "Europe/Madrid"
  }
}
```


La diferencia entre las listas y los diccionarios es la manera diferente de almacenar los elementos. En un diccionario accederemos a la información por

la clave ya que no se guarda ordenadamente como las listas, que siguen un índice.

# Capítulo 3

# Datos

## 3.1 Formato GTFS

Los datos con los que se han realizado en el presente trabajo de fin de grado es un fichero GTFS (*General Transit Feed Specification*) extraído el 08/01/2019 del portal de datos abiertos del Ajuntament de València. Este tipo de datos es una carpeta comprimida con diversos ficheros en formato txt donde cada uno representa diferentes aspectos de la compañía de transportes, en este caso la EMT de Valencia.(*Figura3.1*)

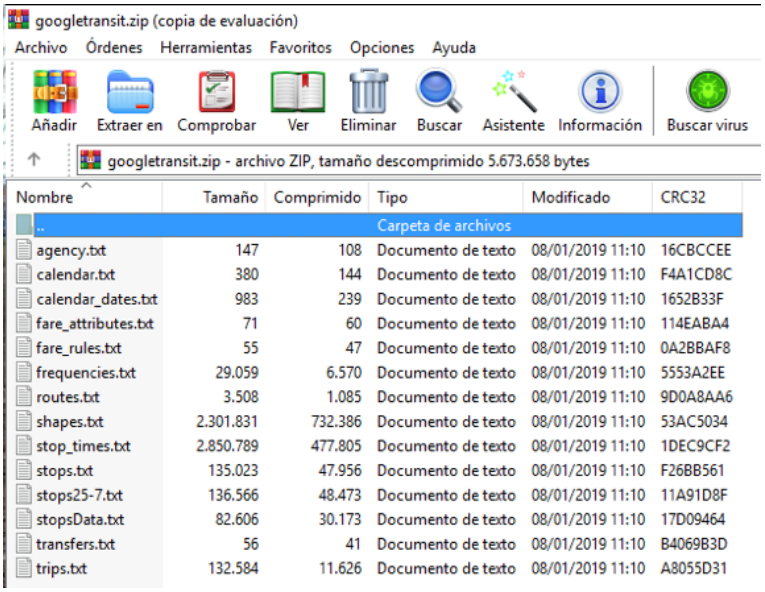


Figura 3.1: Imagen de la carpeta con los datos GTFS.

Durante el proyecto se ha trabajado con diferentes ficheros de texto donde cada uno contenía una información diferente del servicio de autobuses. Muchos de los datos de la especificación son opcionales, lo cual provoca una falta de datos no importantes en todos los ficheros GTFS. A continuación se describen los tipos de datos de la especificación GTFS con el identificador de tipo de dato utilizado posteriormente en el modelo de datos UrbanMobility de Fiware. Cada fichero tiene datos diferentes al resto pero entre ellos hay una cierta relación, por ejemplo, con las rutas y las paradas que en ella están o los tiempos de llegada, la ruta a la que pertenecen y las paradas donde tienen que efectuar parada los autobuses. Por ello, hay cierto atributos dentro de cada fichero que comparten con otros.

## 3.2 Modelo de datos Fiware

### 3.2.1 GTFSStop

El fichero de datos stops.txt contiene información sobre las paradas de autobús en los siguientes campos.(*Figura3.2*)

- Stop_id: identificador único para cada parada de autobús ya que es posible que varias rutas usen la misma parada.
- Stop_code: puede incluir un texto corto o un número que identifica de forma exclusiva la parada de los pasajeros. En el caso de EMT valencia el stop_code coincide con el stop_id ya que está orientado al pasajero.
- Stop_name: indica el nombre de la parada. Por lógica y para que resulte más fácil para las personas locales y turistas, el nombre de la parada pertenece a la calle donde se encuentra.
- Stop_desc: es un campo opcional donde se incluye una descripción sobre la parada como el número de portal para saber a la altura y lado que está de la avenida o calle.
- Stop_lat: incluye la latitud de la parada. Las latitudes están en WGS84.
- Stop_lon: incluye la longitud de la parada. Las longitudes están en WGS84.
- Zone_id: es un campo opcional define la zona de la tarifa para un ID de parada.

- Stop_url: es un campo opcional donde se incluye la URL de la parada en particular.
- Location_type: es un campo opcional indica si el identificador de la parada representa parada, estación o entrada de estación. En caso de que el campo este vacío o con un 0 se considera como parada, 1 para estación y 2 para entrada de estación.
- Parent_station: es un capo opcional y identifica las estaciones asociadas a las paradas que se encuentran dentro de estas.


```
{
    "id": "urn:ngsi-ld:GtfsStop:Malaga_101",
    "type": "GtfsStop",
    "code": {
        "value": "101"
    },
    "operatedBy": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsAgency:Malaga_EMT"
    },
    "location": {
        "type": "geo:json",
        "value": {
            "type": "Point",
            "coordinates": [-4.424393, 36.716872]
        }
    },
    "name": {
        "value": "Alameda Principal Sur"
    }
}
```


Figura 3.2: Modelo de datos de GTFStop de Autobús de la EMT Malaga. Fuente: Fiware GTFSStop data model

Existen otros campos con los que poder ampliar la tabla de información como stop_timezone, wheelchair_boarding y platform_code pero no se encontraban dentro del fichero 'stops.txt'. Estos datos están disponible en el portal de datos abiertos del Ajuntament de València.

### 3.2.2 GtfsAgency

El fichero de datos agency.txt contiene información sobre la agencia.(*Figura3.3*)

- Id: identificador único para la agencia de transporte.
- Type: el tipo de la entidad. En este caso GtfsAgency.

- DataProvider: especifica el URL con información sobre el proveedor de esta información.
- DataCreated: marca de tiempo de creación de la entidad
- DataModified: última actualización de los datos.
- Source: secuencia de caracteres que proporciona la fuente original de los datos de la entidad como URL.
- Zone_id: es un campo opcional define la zona de la tarifa para un ID de parada.
- Name: nombre de la agencia.
- Page: página oficial de la agencia.
- TimeZone: zona horaria en la que opera.
- Phone: número de telefono.
- Language: Idioma en la que opera la agencia.
- Adress: dirección de las oficinas centrales.


```
{
    "id": "urn:ngsi-ld:GtfsAgency:Malaga_EMT",
    "type": "GtfsAgency",
    "name": {
        "value": "Empresa Malague\u00f1a de Transportes"
    },
    "language": {
        "value": "ES"
    },
    "page": {
        "value": "http://www.emtmalaga.es/"
    },
    "source": {
        "value": "http://datosabiertos.malaga.eu/dataset/lineas-y-horarios-bus-google-
    },
    "timezone": {
        "value": "Europe/Madrid"
    }
}
```


Figura 3.3: Modelo de datos de GTFSAgency de Autobús de la EMT Malaga. Fuente: Fiware GTFSAgency data model

### 3.2.3 GtfsRoutes

El fichero de datos routes.txt contiene información sobre las diferentes rutas.(*Figura3.4*)

- Id: identificador único para cada ruta.
- Type: el tipo de la entidad. En este caso GtfsRoute.
- DataProvider: especifica el URL con información sobre el proveedor de esta información.
- DataCreated: marca de tiempo de creación de la entidad
- DataModified: última actualización de los datos.
- Source: secuencia de caracteres que proporciona la fuente original de los datos de la entidad como URL.
- ShortName: número que recibe la ruta.
- Name: nombre de la ruta.
- Description: Descripción sobre la ruta.
- RouteType: tipo de ruta.
- Page: URL de la ruta.
- RouteColor.
- RoutueTextColor.
- OperatedBy: Agencia que opera en la ruta

```
{
    "id": "urn:ngsi-ld:GtfsRoute:Spain:Malaga:1",
    "type": "GtfsRoute",
    "name": {
        "value": "Parque del Sur _ Alameda Principal _ San Andr\u00e9s"
    },
    "shortName": {
        "value": "1"
    },
    "page": {
        "value": "http://www.emtmalaga.es/emt-mobile/informacionLinea.html"
    },
    "routeType": {
        "value": "3"
    },
    "operatedBy": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsAgency:Malaga_EMT"
    }
}
```


Figura 3.4: Modelo de datos de GTFSRoute de Autobús de la EMT Malaga. Fuente: Fiware GTFSRoute data model

### 3.2.4 GtfsTrip

El fichero de datos trip.txt contiene información sobre los diferentes viajes.(*Figura3.5*)

- Id: identificador único para cada viaje.
- Type: el tipo de la entidad. En este caso GtfsTrip.
- DataProvider: especifica el URL con información sobre el proveedor de esta información.
- DataCreated: marca de tiempo de creación de la entidad
- DataModified: última actualización de los datos.
- Source: secuencia de caracteres que proporciona la fuente original de los datos de la entidad como URL.
- HeadSign.
- ShortName.
- Direction.
- Block.
- HasService

- HasShape.
- HasRoute.
- WeelChairAccesible: accesible para sillas de ruedas o no.
- BikesAllowed: bicicletas admitidas o no.

```
{
    "id": "urn:ngsi-ld:GtfsTrip:Spain:Malaga:1",
    "type": "GtfsTrip",
    "direction": {
        "value": 0
    },
    "headSign": {
        "value": "San Andr\u00e9s"
    },
    "hasRoute": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsRoute:Spain:Malaga:1"
    },
    "hasService": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsService:Malaga_LAB"
    },
    "hasShape": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsShape:Shape01"
    }
}
```


Figura 3.5: Modelo de datos de GTFSTrip de Autobús de la EMT Malaga. Fuente: Fiware GTFSTrip data model

### 3.2.5 GtfsStopTime

El fichero de datos stop_time.txt contiene información sobre los horarios de llegada a las pardas.(*Figura3.6*)

- Id: identificador único para cada parada.
- Type: el tipo de la entidad. En este caso GtfsStopTime.
- DataProvider: especifica el URL con información sobre el proveedor de esta información.
- DataCreated: marca de tiempo de creación de la entidad
- DataModified: última actualización de los datos.
- Source: secuencia de caracteres que proporciona la fuente original de los datos de la entidad como URL.

- HasTrip: identificador del viaje.
- HasStop: identificador de las paradas de autobús.
- ArrivalTime: hora de llegada a la parada.
- DepartureTime: hora de salida de la parada.
- StopSequence
- StopHeadsign.
- Pickuptype: tipo de recogida.
- DropOffType.
- TimePoint.


```
{
    "id": "urn:ngsi-ld:GtfsStopTime:Spain:Madrid:EMT:FE0010011_737",
    "type": "GtfsStopTime",
    "departureTime": {
        "value": "07:04:24"
    },
    "hasTrip": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsTrip:Madrid:EMT:FE0010011"
    },
    "stopSequence": {
        "value": 4
    },
    "distanceTravelled": {
        "value": 759
    },
    "arrivalTime": {
        "value": "07:04:24"
    },
    "hasStop": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsStop:Madrid:EMT:737"
    }
}
```


Figura 3.6: Modelo de datos de GTFStopTime de Autobús de la EMT Malaga. Fuente: Fiware GTFSStopTime data model

## 3.3 Limitaciones de GTFS

Los ficheros GTFS al ser solo ficheros con formato .txt carecen de relación de datos entre ellos. Por eso, una línea del trabajo a desarrollar sería la creación de dichas relaciones mediante una base de datos como la que se puede observar en la *Figura 3.7*.

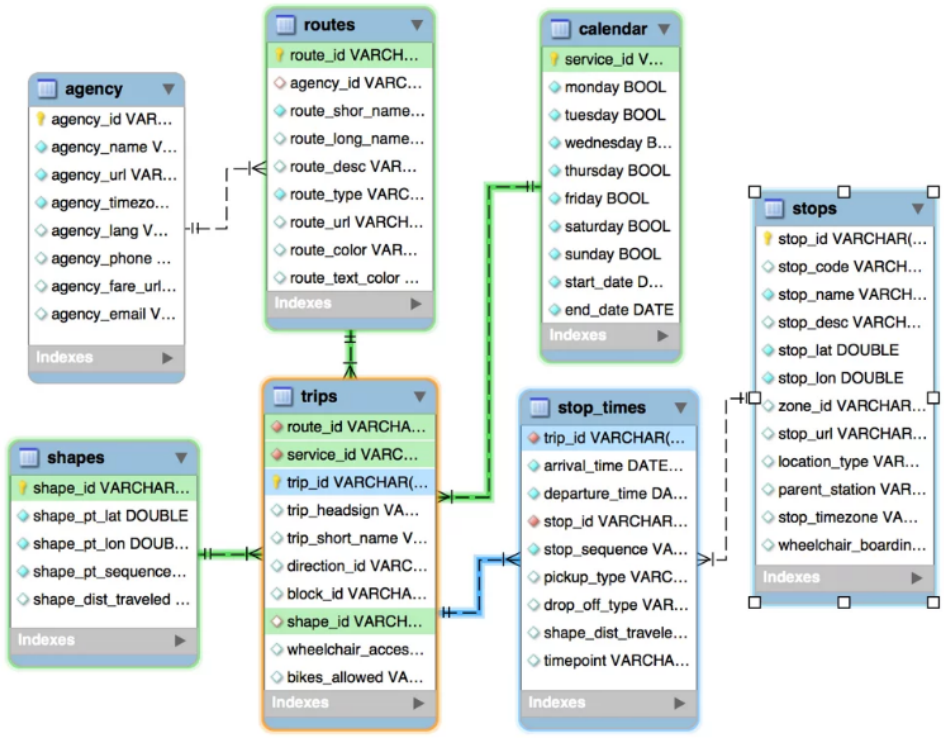


Figura 3.7: Relación entre ficheros GTFS. Fuente: Altantbh (2016)

Estas relaciones junto con la creación de un índice espacial ayudarían a la reducción del tiempo de búsqueda para realizar un viaje en autobús combinando el posicionamiento GPS del móvil (sabiendo así en qué celda del índice se encuentra el dispositivo) y la relación entre las diferentes tablas para llegar de un punto A a un punto B.

## 3.4 Formato GeoJSON

Es un formato para codificar una variedad de estructuras de datos geográficos usando JavaScript Object Notation (JSON). Un objeto GeoJSON representa una región del espacio, un atributo o una lista de atributos. Se pueden encontrar varios tipos de geometría [15]:

- Punto y multi-punto.
- Línea o multi-línea.

- Polígono o multi-polígono
- Colección de Geometría.

El valor del objeto geométrico debe ser al menos uno de los siete tipos geométricos. Todos los datos en formato GeoJSON deben tener un campo que se llame coordenadas.


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -0.4099999999999966,
              39.44999999999999
            ],
            [
              -0.3999999999999966,
              39.44999999999999
            ],
            [
              -0.3999999999999966,
              39.45999999999999
            ],
            [
              -0.4099999999999966,
              39.45999999999999
            ],
            [
              -0.4099999999999966,
              39.44999999999999
            ]
          ]
        ]
      },
      "type": "Feature",
      "properties": {
        "GID": "(17959, 12945)"
      }
    }...
    ]
}
```

Para especificar un polígono, por ejemplo, se debería tener al menos cuatro o más posiciones con coordenadas, que la primera y la última posición sean idénticas, por último, la creación de ese polígono puede ser o el límite del mismo o el límite de un vacío dentro de la superficie del polígono. Una colección de geometría puede ser una composición heterogénea de pequeños objetos espaciales. El sistema de referencia que usan todos los ficheros GeoJSON es el WGS84 con longitudes y latitudes en grados decimales. Esto es equivalente al sistema de referencia de coordenadas identificado por Consorcio Geoespacial Abierto (CGA). Un elemento opcional de tercera posición debe ser la altura en metros por encima o por debajo del elipsoide de referencia WGS 84. En ausencia de valores de elevación, las aplicaciones sensibles a la altura o la profundidad deberían interpretar las posiciones como un terreno local o nivel del mar [15].

En los últimos años las aplicaciones que se basan en la ubicación están cogiendo importancia. Una gran cantidad de datos contienen información geográfica que antes su codificación era accesible a un los profesionales y que últimamente hay una gran comunidad importante de mapeo de código abierto [19]. Es por eso y por su fácil aplicación y usos que los formatos GeoJSON son cada vez más utilizados en los ámbitos de codificación geográfica.

El Identificador Uniforme de Recursos (URI) es una secuencia compacta de caracteres que identifica un recurso abstracto o físico. El esquema 'geo' URI identifica ubicaciones geográficas en un sistema de referencia de coordenadas, que es, por defecto, el Sistema Geodésico Mundial 1984 (WGS84). El esquema proporciona la representación textual de las coordenadas espaciales de la ubicación en dos o tres dimensiones (latitud, longitud y, opcionalmente, altitud para el WGS-84). A continuación se muestra un ejemplo de dicho 'geo' URI:

```
geo: 13.4125,103.8667
```

Tales URI son independientes de un protocolo específico, aplicación o formato de datos, y se pueden usar en cualquier otro protocolo o formato de datos que admita la inclusión de URI arbitrarios.

Los 'geo' URI identifican ubicaciones geográficas y se pueden asignar ubicaciones precisas a objetos de geometría GeoJSON. Un 'geo' URI con dos coordenadas y una geometría de punto GeoJSON pueden mapearse entre sí. Un punto GeoJSON siempre se convierte en un 'geo' URI que no tiene ningún parámetro de incertidumbre.

# Capítulo 4

# Metodología

Primero que todo se debe instalar una máquina virtual que nos permita la ejecución del Sistema Operativo de Linux en caso de no contar con el Windows Pro o el Windows Enterprise para la instalación de Docker. Para la instalación de Docker se necesitan permisos del administrador. Por eso, cada vez que se empiece a usar la terminal del Sistema Operativo y durante todas las comandas que se realicen, se deberá escribir la palabra 'sudo' al inicio de esta permitiendo actuar como administrador.

Una vez completada la instalación del software se procede a crear contenedores y una red de conexiones entre el PC del usuario y Fiware. Para ello se necesita la instalación de MongoDB y de Orion. Lo que se creará posterior de su instalación es una red virtual para la gestión de los contenedores y el arranque de todos los componentes para conectarlos a la red.

Durante la instalación, se deberá asignar a Docker la conexión a un puerto del servidor web, en este caso el 1026. Este puerto es uno cedido por Fiware para la carga masiva de datos. Todos los comandos que se requieran para realizar una consulta o modificación de datos en el servidor cedido por Fiware podría realizar con el comando cURL.

La herramienta cURL es muy útil para gestión de datos cortos que no necesiten muchas líneas de comando. Por ello, cuando se necesita hacer una gestión de datos grandes como los GTFS, esta herramienta pierde utilidad y la mejor manera de gestionar los datos es mediante un entorno de programación Python.

Antes de empezar con la escritura del código con Python, es necesario la instalación de paquetes que permitan conectar el PC con el servidor donde cargar los datos. Se necesita la instalación de la librería para las peticiones

HTTP llamada "requests" y también la librería para la representación de los puntos en un un mapa, llamado "folium". A la hora de usar estas dos librerías en el código de los programas, se les llamará escribiendo al principio del programa 'import requests' o en el caso del folium, 'import folium', muy habitual en el entorno Python.

Sabiendo los datos que se van a utilizar durante la ejecución del proyecto se procede a la creación de fiheros GeoJSON. Este formato permite intercambiar datos de una manera sencilla, legible y fácil de comprender. A su vez, son formatos muy útiles para trabajar con aplicaciones WebMapping ya que el GeoJSON contiene objetos *features* y este a su vez diferentes entidades divididas en *geometry* y *properties*.

En este tipo de ficheros podemos encontrar varios tipos de geometría que se podrán usar para crear el fichero GeoJSON:

- Punto y multi-punto.
- Línea y polilínea.
- Polígono y multi-polígono.
- Colección de Geometría.

Un ejemplo donde se representa un polígono con sus coordenadas y propiedades en un GeoJSON es el siguiente:

```
{
      "geometry": {
        "type": "Point",
        "coordinates": [
          -0.402776519236904,
          39.458628918607
        ]
      },
      "type": "Feature",
      "properties": {
        "GID": "(17959, 12945)",
        "PID": "1175"
      }
    },
    {
      "geometry": {
        "type": "Point",
        "coordinates": [
```

```
            -0.404016751851126,
            39.4563032183764
          ]
        },
        "type": "Feature",
        "properties": {
          "GID": "(17959, 12945)",
          "PID": "1225"
        }
      },...
      ]
}
```


Una vez claro el concepto de un fichero GeoJSON se procede a programar con Python el archivo con las paradas de la EMT Valencia. Al ejecutar el programa *Apéndice C.1*, este lee el fichero de paradas con la ruta que tiene definida, lo transforma en un fichero GeoJSON con sus coordenadas y sus propiedades para posteriormente validarlo en la página web http://geojson.io/. Este sitio web abrirá un mapa con una representación de las paradas en forma de puntos donde al hacer click aparecerá una tabla editable con las propiedades de la parada así como la información, que en este caso son sus coordenadas(*Figura 4.1*).

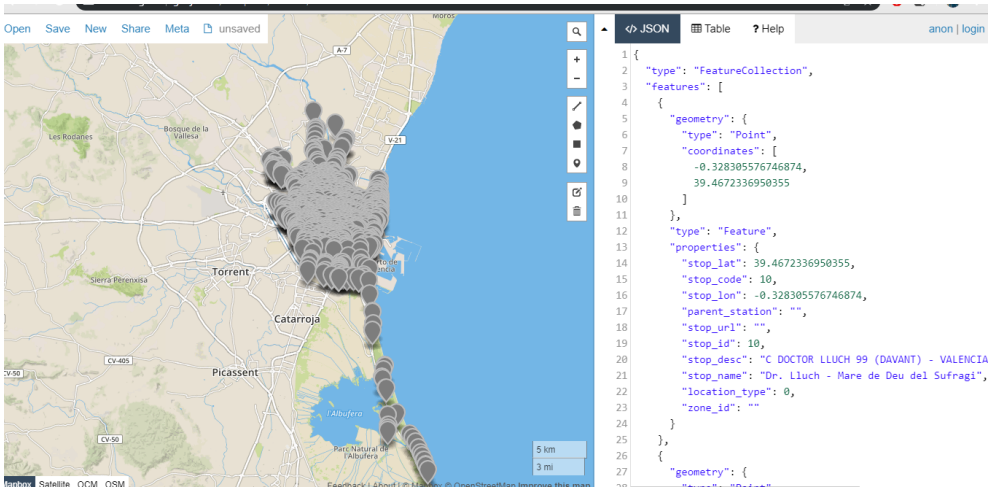


Figura 4.1: Paquete de datos con todas las paradas de la EMT Valencia

La creación de los ficheros GeoJSON de las paradas de la EMT, permitirá que se cree un índice en forma de cuadrícula donde cada uno de los polígonos contendrá al menos una parada de autobús, reduciendo así el tiempo de computación en la búsqueda de la parada más cercana.

El código (*Apéndice C.2*) genera un índice espacial a nivel global, es decir, de 90°N a 90°S y 180°E a 180°O con una resolución de 0.01°×0.01° por celda. Las celdas que se guardan en un fichero que las contiene con su 'Id' como clave y su valor es la lista de estaciones en cada celda.

Una vez creado el índice espacial, se visualiza en la web mediante la librería de Python Folium. Esta visualización se guarda en el ordenador como un fichero HTML (Hyper Text Markup Lenguage). El índice espacial creado es la representación de todas las celdas donde al menos hay una parada de autobús. Puede ser muy útil para el usuario final agilizar la búsqueda de paradas cercanas a su posición y así calcular la ruta más rápida al destino deseado. (*Figura 4.2*)

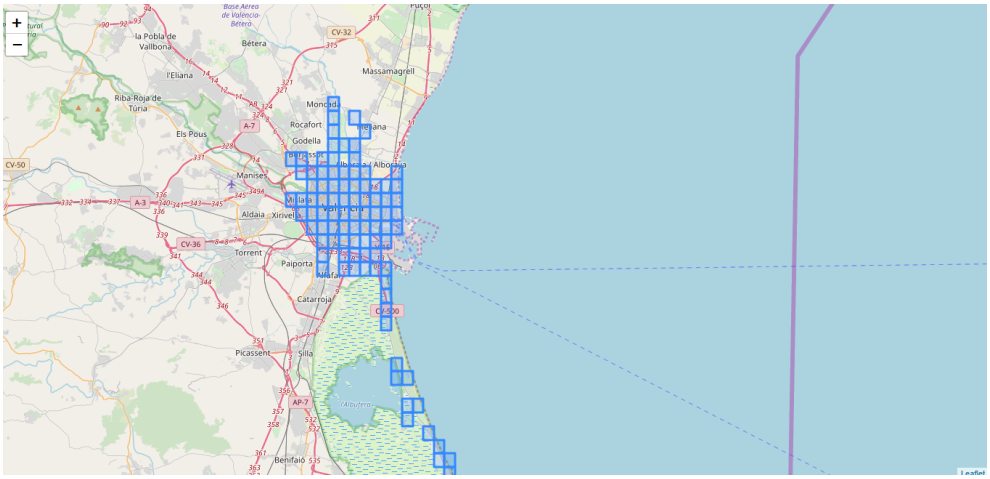


Figura 4.2: Índice espacial para la ciudad de Valencia.

El *HyperText* está basado en el HTTP o también conocido como Hyper Text Transfer Protocolo.'*El Protocolo de transferencia de hipertexto (HTTP) es un protocolo de nivel de aplicación para hipermedia distribuido, colaborativo entre sistemas de información. HTTP ha sido utilizado por la iniciativa de información global de la World-Wide Web desde 1990.*'[13]. Con HTTP suele enviarse diferentes tipos de archivos así como *scripts* y su funcionamiento es de la siguiente manera. Cuando un usuario hace una consulta en su navegador, este se conecta mediante HTTP con el Puerto 80, encargado de aceptar las peticiones web. Una vez la información ha sido entregada al usuario el servidor web cierra la conexión.

El siguiente paso a seguir en este proyecto es la carga masiva de datos a la plataforma Fiware con todo lo relacionado con los modelos de datos GTFS de la EMT de Valencia. El código al ejecutarse empieza a leer una

por una todas las líneas de los distintos ficheros que se han utilizado, en este caso, los modelos de datos GtfsAgency, GtfsStops, GtfsRoute, GtfsTrip, GtfsStopTime. La ejecución del código generará por cada línea del fichero un diccionario con clave-valor con los datos que hay en ella. Este diccionario contendrá todos los datos que estén relacionados con el modelo de datos de Fiware y sigan su formato, en caso contrario la carga de datos al servidor no se podría realizar. Por ejemplo, en la *Figura 4.3* se muestra el modelo de datos para las paradas de autobús.


```
{
    "id": "urn:ngsi-ld:GtfsStop:Malaga_101",
    "type": "GtfsStop",
    "code": {
        "value": "101"
    },
    "operatedBy": {
        "type": "Relationship",
        "value": "urn:ngsi-ld:GtfsAgency:Malaga_EMT"
    },
    "location": {
        "type": "geo:json",
        "value": {
            "type": "Point",
            "coordinates": [-4.424393, 36.716872]
        }
    },
    "name": {
        "value": "Alameda Principal Sur"
    }
}
```


Figura 4.3: Modelo de datos de GTFStop de Autobús de la EMT Malaga. Fuente:Fiware GTFSStop data model

Para una mejor representación geográfica de los datos cargados, la librería Folium permite una visualización en forma de mapa de los datos. Por eso, se lleva a cabo la escritura del código que nos permita visualizar la distribución de las paradas a lo largo de la ciudad. Esta librería de Python es una herramienta de visualización muy buena para representar en un mapa el proyecto que se está realizando. Folium guarda sus mapas en un fichero HTML.

Otra rama que podría salir de este proyecto que va relacionada con la creación de la base de datos representada en la *Figura 3.7*, podría ser la búsquedade de las rutas que el usuario quiere hacer para desplazarse desde su posición. (*Figura 4.4*)

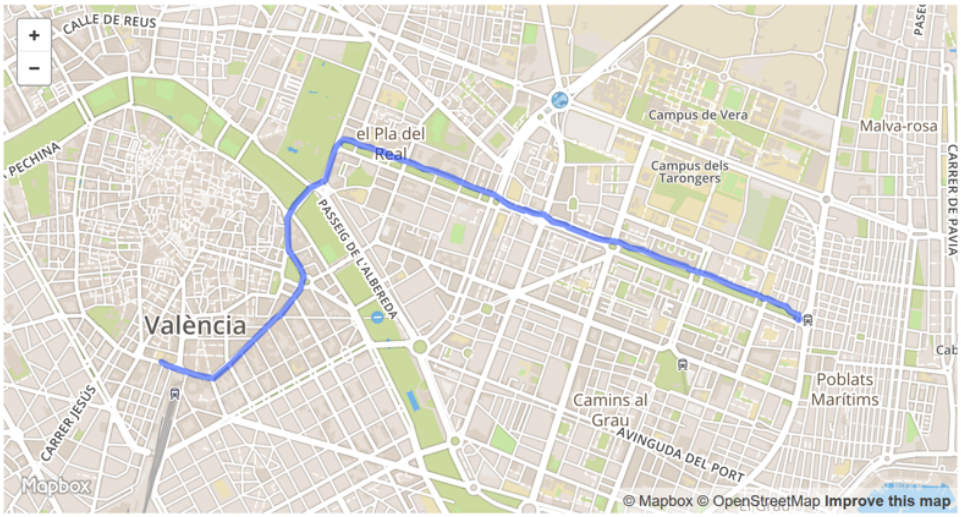


Figura 4.4: Línea 81 de la EMT Valencia. Fuente: Ajuntament de València

Para ello los pasos que se deberían seguir son los siguientes:

1. Definir origen: posición GPS del móvil del usuario.
2. Definir destino: podría ser que el usuario lo defina sobre el mapa o mediante dirección postal del destino. Esto sería una posición cercana a la parada de destino o parada 2 (P2).
3. Buscar parada más cercana (P1) a la posición GPS del móvil del usuario.
4. Buscar rutas que pasan por dicha parada P1.
5. Obtener paradas de las rutas que pasan por P1.
6. Obtener la parada más próxima al destino P2.
7. Conociendo la parada origen P1, la parada destino P2 y la ruta que une dichas paradas ya tenemos el trayecto y se podría dibujar.

Para hacer todo esto hay que utilizar varios ficheros GTFS y crear las relaciones pertinentes para que el código que se programe pueda funcionar correctamente.

# Capítulo 5

# Resultados

Tras la realización del trabajo realizado durante el proyecto, se abordará y se explicarán los resultados obtenidos durante su desarrollo. El resultado final y principal es la carga masiva de datos a la plataforma de Fiware para que estos puedan ser utilizados por diferentes usuarios tanto a nivel de desarrollo de aplicaciones como para su uso diario.

## 5.1 GeoJSON estaciones EMT Valencia

Como primer resultado tenemos la creación del fichero GeoJSON con toda la información de las paradas de la EMT Valencia.


```
{
    "type": "FeatureCollection",
    "features": [
        {
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -0.328305576746874,
                    39.4672336950355
                ]
            },
            "type": "Feature",
            "properties": {
                "stop_lat": "39.4672336950355",
                "stop_code": "10",
                "stop_lon": "-0.328305576746874",
                "parent_station": "",
```

```
            "stop_url": "",
            "stop_id": "10",
            "stop_desc": "C DOCTOR LLUCH 99 (DAVANT) - VALENCIA",
            "stop_name": "Dr. Lluch - Mare de Deu del Sufragi",
            "location_type": "0",
            "zone_id": ""
        }
    },
    {
        "geometry": {
            "type": "Point",
            "coordinates": [
                -0.345434552993697,
                39.4510148911155
            ]
        },
        "type": "Feature",
        "properties": {
            "stop_lat": "39.4510148911155",
            "stop_code": "1007",
            "stop_lon": "-0.345434552993697",
            "parent_station": "",
            "stop_url": "",
            "stop_id": "1007",
            "stop_desc": "AV JESUS MORANTE BORRAS 26 - VALENCIA",
            "stop_name": "Jesus Morante Borras (I)",
            "location_type": "0",
            "zone_id": ""
        }
    },
    {
        "geometry": {
            "type": "Point",
            "coordinates": [
                -0.344468097469677,
                39.4494826719026
            ]
```


Lo mostrado es el resultado de la ejecución del primer código (*Apéncice A.1*) donde se muestra toda la información de las paradas en el fichero con formato txt. En este ejemplo se observa las propiedades de dos paradas y sus coordenadas en formato de punto. Para la validadción de su código se recurre a la página web `http://geojson.io/`. Una vez validado su código, la representación de las paradas será un mapa donde se encuentren todas ellas y al hacer zoom se aprecia su posición.

La carga de datos en la plataforma Fiware se sube a un servidor prestado por la misma plataforma. En el caso del trabajo, este servidor es uno alquilado

por la UPV `tehttp://137.74.44.56:1026/v2/entitiesxt` y es aquí donde todos los datos se suben en formato GeoJSON. (*Figura 5.1*)


```
▼ 0:
    id:          "urn:ngsi-ld:GtfsRoute:28"
    type:        "GtfsRoute"
  ▼ name:
      type:      "Text"
      value:     "Ciutat Artista Faller  Mercat Central"
      metadata:  {}
  ▼ operatedBy:
      type:      "Relationship"
      value:     "urn:ngsi-ld:GtfsAgency:urn:ngsi-ld:GtfsAgency:EMT"
      metadata:  {}
  ▼ routeType:
      type:      "Text"
      value:     "3"
      metadata:  {}
  ▼ shortName:
      type:      "Text"
      value:     "28"
      metadata:  {}
▼ 1:
    id:          "urn:ngsi-ld:GtfsRoute:95"
    type:        "GtfsRoute"
  ▼ name:
      type:      "Text"
      value:     "Parc de Capçalera  Ciutat Arts i Ciencies"
      metadata:  {}
  ▼ operatedBy:
      type:      "Relationship"
      value:     "urn:ngsi-ld:GtfsAgency:urn:ngsi-ld:GtfsAgency:EMT"
      metadata:  {}
  ▼ routeType:
      type:      "Text"
```


Figura 5.1: Datos cargados en el servidor cedido por la plataforma Fiware.

El ejemplo de la *Figura 5.1* son los datos cargados del fichero de las diferentes rutas de la agencia. Se puede apreciar que cumple con el modelo de datos planteado por Fiware. De otra manera, no hubiera sido posible la subida de datos al servidor. Para una mejor visualización del trabajo realizado, se representa la carga de datos con la librería Folium de Pyhton. Este mapa cargará todas las paradas subidas en la plataforma Fiware con un mapa base de Open Street Maps. De esta manera tendremos una mejor percepción espacial de la localización de las paradas como se muestra en la (*Figura 5.2*).

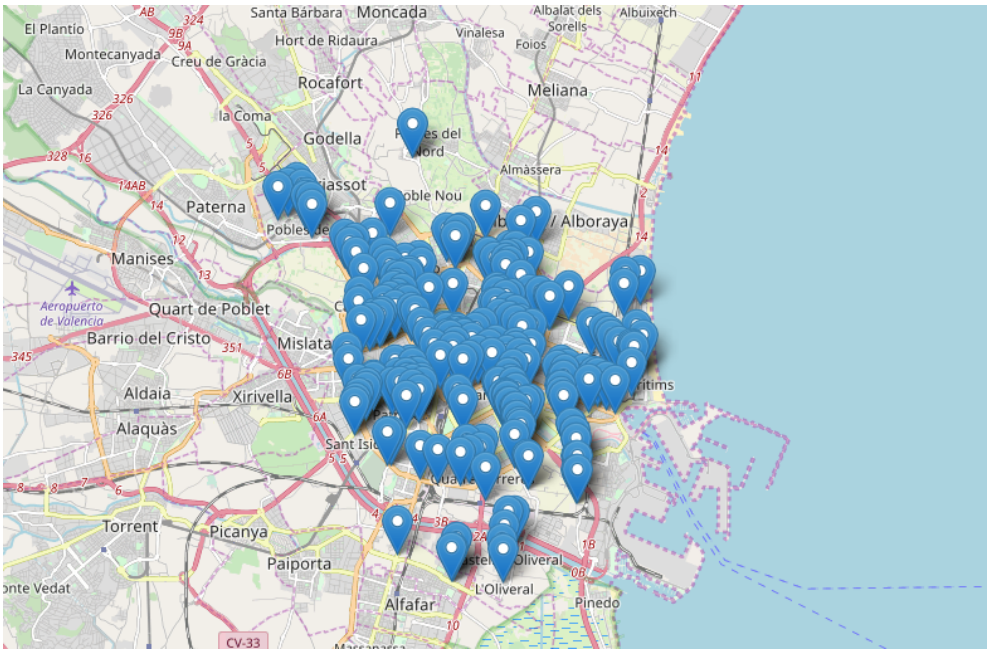


Figura 5.2: Paradas de autobús después de la ejecución del Folium.

## 5.2 Índice espacial.

El fichero que se genera es un formato .html donde se representa la ciudad de Valencia con la división de 91 celdas. Cada celda cuenta con una resolución espacial de 0.1°x0.1°.

La *Figura 4.2* muestra una malla o índice espacial donde al menos cada una de las celdas contiene una parada de autobús. Con el uso de este índice espacial se reduciría el tiempo de búsqueda para una ruta o encontrar la parada más cercana. Sin él, los tiempo de búsqueda aumentarían sobre manera haciendo poco útil la aplicación de los datos por si solos.

Dependiendo donde quiera ir el usuario final desde su ubicación, se le asignará la parada más cercana que se ajuste a su búsqueda.

# Capítulo 6

# Conclusiones

Vivimos en un mundo que se va haciendo más urbano a pasos agigantados. Desde la primera revolución industrial hasta nuestros tiempos, el flujo de población de las áreas rurales a las urbanas ha provocado un aumento descomunal de la población en las ciudades.

Estos crecimientos poblacionales descontrolados de la población provocan en la persona que vive en las grandes urbes un constante estrés, una mala respiración debido al aire contaminado por los vehículos, así como una mala calidad tanto en el agua como en la vida en general.

Es por ello por lo que las Smart Cities tienen un papel fundamental en el día a día de las ciudades. El lector debe recordar que el papel de las Smart Cities es hacer al ciudadano partícipe en todos los campos de la ciudad, desde el buen uso y gestión del agua hasta la transparencia del gobierno. Una ciudad sana significa una ciudad con aires limpios, una ciudad con zonas verdes, una ciudad con buenas comunicaciones tanto para las personas de a pie como para los diferentes transportes públicos. En general, una ciudad por y para los ciudadanos.

¿Cómo empezar a construir una Smart City? Los comienzos siempre son difíciles, pero una buena gestión del transporte público con buenas conexiones y respetuosos con el medio ambiente siempre ayudarán a que los ciudadanos de las ciudades y de las ciudades adyacentes usen y fomenten este servicio público. Este uso masivo del transporte público generará una bajada significativa del $CO_2$ así como de otros elementos contaminantes en el aire que todos respiran como puede ser el NO, $NO_2$, $NO_X$, etc.

Con la carga de datos GTFS a la plataforma Fiware no solo de la ciudad de Valencia si no de todas las ciudades con un servicio público, garantizará

que se puedan producir diversas aplicaciones para poder concienciar a la ciudadanía de la necesidad de este servicio y de esta manera limpiar el aire de nuestras ciudades.

La carga de datos Fiware permite al programador poner datos a disposición de otros usuarios con un acceso (API) fácil basado en protocolos estándard. Para que los usuarios utilizaran estos datos se debería crear relaciones entre los ficheros de formato GTFS como en la *Figura 3.7* ya que el propio formato no las tiene.

A modo de resumen, para que todo esto suceda y podamos reconocer a una ciudad como Smart City, las administraciones públicas deben hacer campañas de divulgación y transparencia de datos para que todos los ciudadanos de una ciudad conozcan su existencia ya que este es un aspecto fundamental de las ciudades inteligentes como se ha dicho en la introducción.

# Bibliografía


[1] Geojson. https://geojson.org/.

[2] Guía de salarios IT. https://recursos.openwebinars.net/wp-content/uploads/2019/04/Guia-de-Salario-IT.pdf.

[3] ¿Qué es HTTP? https://www.pickaweb.es/ayuda/que-es-http/.

[4] What is Fiware? https://www.fiware.org/about-us/.

[5] Los beneficios de usar Docker y contenedores a la hora de programar, 2018.

[6] Albino, V., Berardi, U., and Dangelico, R. M. Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of urban technology 22*, 1 (2015), 3–21.

[7] Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., and Schmidt, C. The geojson format specification. *Rapport technique 67* (2008).

[8] Dameri, R. P., and Rosenthal-Sabroux, C. *Smart city: How to create public and economic value with high technology in urban space*. Springer, 2014.

[9] Developers, G. Descripción general de la especificación gtfs estática. https://developers.google.com/transit/gtfs/.

[10] Duque, R. G. Python para todos.

[11] Enerlis, E., and Young, F. Madrid network (2012): Libro blanco de smart cities. *Madrid. Imprintia. Recuperado de: https://goo. gl/qxrmmN*.

[12] Environment, U. Perspectivas del medio ambiente mundial GEO6.

[13] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext transfer protocol–http/1.1, 1999.

[14] Fiware. Fiware data models. `https://www.fiware.org/developers/data-models/`.

[15] Gillies, S., Butler, H., Daly, M., Doyle, A., and Schaub, T. The geojson format. *coordinates 102* (2016), 0–5.

[16] Goasguen, S. *Docker Cookbook: Solutions and Examples for Building Distributed Applications*. ."O'Reilly Media, Inc.", 2015.

[17] Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., and Williams, P. Foundations for smarter cities. *IBM Journal of research and development 54*, 4 (2010), 1–16.

[18] Ltd, E. The networked society city index. *Developed by Ericsson with Sweco* (2014), 30pp.

[19] Mayrhofer, A., and Spanring, C. A uniform resource identifier for geographic locations ('geo'uri), 2010.

[20] Nam, T., and Pardo, T. A. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times* (2011), ACM, pp. 282–291.

[21] Serrano, F. P. Json y geojson en el mundo gis. `https://mappinggis.com/2018/03/json-y-geojson-en-el-mundo-gis/`.

# Parte II

# Apéndices

# Apéndice A

# Presupuestos

Para la elaboración de los presupuestos se ha tenido en cuenta el salario que cobraría un perfil junior (0-2 años de experiencia) en el campo de *Python Developer* [2]. El salario anual de este tipo de trabajadores en Valencia es alrededor de los 31.000 € anuales.

| Razón | Coste |
|---|---|
| Estudio y busqueda de información | 200€ |
| Horas dedicas a la elaboración del proyecto | 4842.00€ |
| *Un total de 300 horas a 16.14€/hora* | |
| Conocimientos técnicos | 500€ |
| Mantenimiento de los equipos | 50€ |
| Honorarios | 150€ |
| Total( %IVA) | 6947.82€ |

Cuadro A.1: Presupuesto del proyecto

# Apéndice B

# Intalación Docker

La instalación de Docker se realizará de la siguiente manera:

1. `$ sudo apt-get update`.

   Este comando nos permite actualizar los repositorios con los que el SO está trabajando.

2. `$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common`.

   Para la instalación de paquetes de apoyo.

3. `$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -` .

   Se obtiene la clave de instalación de Docker GNU Privacy Guard, agregándola a la base de datos de Ubuntu.

4. `$ sudo add-apt-repository \deb [arch=amd64]  https://download.docker.com/linux/ub $(lsb_release -cs) stable"`.

   Se añade los repositorios de Docker en Ubuntu.

5. `te$ sudo apt-get install docker-cext`.

   Finalmente se instala el Docker en el Ubuntu.

Una vez completada la instalación del software se procede a crear contenedores y una red de conexiones entre el PC del usuario y Fiware. Para ello se necesita la instalación de MongoDB y de Orion. Lo que se creará a continuación de su instalación es una red virtual para la gestión de los contenedores y el arranque de todos los componentes para conectarlos a la red.

1. `$ sudo docker pull mongo:3.6`.

   Instalación del MongoDB.

2. `$ sudo docker pull fiware/orion`.

   Instalación del Orion.

3. `$ sudo docker network create fiware_default`.

   Creamos la red virtual para la gestión de contenedores.

4. `$sudo docker run --detach --restart=always --name=mongo-db --network=fiware --expose=27017 mongo:3.6 --bind_ip all --smallfiles`.

5. `$sudo docker run --detach --restart=always --name=fiware-orion --hostname=orion --network=fiware_default --publish=1026:1026 fiware/orion -dbhost mongo-db`

   Se arrancan todos los componentes y se conectan a la red.

De esta manera ya tendriamos Docker listo para empezar a trabajar con él.

# Apéndice C

# Códigos Fuente

## C.1 Transformación a GeoJSON

Transformación de las coordenadas de las paradas de la EMT Valencia a GeoJSON.

```
%Lista de elementos (features).
feat = []
for r1 in t1[1:]:
    r2 = r1.strip().split(',')
    try:
        lng = float(r2[5]) %Coordenadas.
        ltd = float(r2[4])
    except:
        print r2
        continue
    propd = dict(zip(prop, r2)) %Propiedades.
    %La expresion dict(zip(lista1, lista2)) convierte dos listas en un diccionario).
    f = {}
    f['type'] = 'Feature'
    f['geometry'] = {'type': 'Point', 'coordinates': [lng,ltd]}
    f['properties'] = propd
    feat.append(f)
%GeoJSON en formato diccionario Python.
geojsond = {'type': 'FeatureCollection', 'features': feat}
```

En este código se hace una lectura al fichero de paradas para coger la información necesaria y crear el fichero GeoJSON

{

```
  "type": "FeatureCollection",
  "features": [
    {
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -0.4099999999999966,
              39.44999999999999
            ],
            [
              -0.3999999999999966,
              39.44999999999999
            ],
            [
              -0.3999999999999966,
              39.45999999999999
            ],
            [
              -0.4099999999999966,
              39.45999999999999
            ],
            [
              -0.4099999999999966,
              39.44999999999999
            ]
          ]
        ]
      },
      "type": "Feature",
      "properties": {
        "GID": "(17959, 12945)"
      }
    }...
    ]
}
```


## C.2 Índice espacial

Creación del Índice espacial de resolución 0.1°x0.1°.

```
%Bucle para crear el indice espacial a nivel global.
celdas={}
celdasll = {}
```

```
for r1 in t1[1:]:
    r2 = r1.strip().split(',') %Separamos las filas del fichero stop

    %celdas de las coordenadas e identificador
    idx= int(divmod(float(r2[5])-Xmin,dx)[0])
    idy= int(divmod(float(r2[4])-Ymin,dy)[0])

    try:
        celdas[(idx,idy)].append(r2[0])
        celdasll[(idx,idy)].append([r2[0], float(r2[5]), float(r2[4])])
    except:
        celdas[(idx,idy)]=[r2[0]]
        celdasll[(idx,idy)]=[[r2[0], float(r2[5]), float(r2[4])]]
```

El código generará unas celdas por toda la cudad de Valencia donde se encuentre una parada de autobús y creará ficheros con las coordenadas de las celdas así como las coordenadas de las paradas que contenga.


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -0.4199999999999875,
              39.49000000000001
            ],
            [
              -0.4099999999999875,
              39.49000000000001
            ],
            [
              -0.4099999999999875,
              39.50000000000001
            ],
            [
              -0.4199999999999875,
              39.50000000000001
            ],
            [
              -0.4199999999999875,
              39.49000000000001
            ]
          ]
```

```
        ]
      },
      "type": "Feature",
      "properties": {
        "GID": "(17958, 12949)"
      }
    },
    {
      "geometry": {
        "type": "Point",
        "coordinates": [
          -0.417785120056735,
          39.4990924668261
        ]
      }, ...

    ]
}
```

## C.3 Carga de datos a Fiware

Carga masiva de datos a la plataforma Fiware. En el fragmento de código que se muestra a continuación corresponde a la carga de datos sobre la agencia que lleva el servicio del transporte público.

```
% -*- coding: utf-8  -*-

import json
import requests

url = 'http://137.74.44.56:1026/v2/entities'
head = {'Content-Type': 'application/json'}

%CSV file agency.
fm = 'C:/Users/jolu9/Documents/TFG/googletransit/agency.txt'
f2 = open(fm, 'rt', encoding='utf8')
t2 = f2.readlines()
f2.close()

%Data agency dictionary.
agency = {}

%Add entities
for r3 in t2[1:]:
    r4=r3.strip().split(',')
```

```
    agency = { \
        'id':'urn:ngsi-ld:GtfsAgency:' + r4[0], \
        'type':'GtfsAgency', \
        'name': { \
            'value': r4[1]
            }, \
        'page':{ \
            'value': r4[2]
            }, \
        'timezone':{ \
            'value': r4[3]
            }
        }

    print(json.dumps(agency, indent=2))
    print(type(json.dumps(agency, indent=2)))

    r = requests.post(url,headers=head,data=json.dumps(agency))
    print (r)
    print(r.content)

% CSV file stops.
fn = 'C:/Users/jolu9/Documents/TFG/TRABAJO/stops.txt'
f1 = open(fn, 'rt', encoding='utf8')
t1 = f1.readlines()
f1.close()
% Data stop dictionary...
datad = {}
% Add entities...
for r1 in t1[1:]:
    r2 = r1.strip().split(',')
    if len(r2)>1:
        % Payload - Simple attributes...
        datad['id']='urn:ngsi-ld:GtfsStop:' + r2[0]
        datad['type']='GtfsStop'
        datad['code']= { \
            'value': r2[0]
            }
        datad['operatedBy']= { \
            'type':'Relationship', \
            'value': agency['id']
            }
        % Payload - Complex attributes...
```

Como se ha dicho durante el proyecto, se cargan los datos en un puerto del servidor de Fiware. La carga de los datos mostrados en el código anterior se visualizarían en la plataforma de la siguiente manera:

## C.4 Mapa Folium

Código para crear el mapa de Folium.

```
% Retrieve access points...
url = 'http://137.74.44.56:1026'
end = '/v2/entities'
pars = {'type': 'GtfsStop'}
% Layer of access points...
pnt = folium.FeatureGroup(name='Bus_stops')
```

Seleccionamos de la plataforma Fiware los datos que se desean utilizar para el mapa.

```
            for g in rj:
             %print (g)
             gid=g['code']['value']
             lng,ltd = g['location']['value']['coordinates']
             pnt.add_child(folium.Marker(location=[ltd,lng], \
                                         popup=(folium.Popup(gid))))
% Create map...
m = folium.Map()
m.add_child(pnt)  % Point layer...
m.fit_bounds(pnt.get_bounds())  % Zoom...
% Create HTML map file...
mfn = './folium.html'
m.save(mfn)
% Show map in default browser...
webbrowser.open(os.path.abspath(mfn))
```

```
        datad['location'] = { \
            'type':'geo:json', \
            'value': { \
            'type':'Point', \
            'coordinates':[float(r2[5]),float(r2[4])]
            }
            }
```

Con esos datos extraídos de la plataforma les damos la forma adecuada para que el programa de Folium pueda ejecutarse y crear el mapa que se desea.

# 53627535:Creación_de_un_repositorio_de_contenidos_para_tra.

INFORME DE ORIGINALIDAD

| 6% | 5% | 2% | 4% |
|---|---|---|---|
| INDICE DE SIMILITUD | FUENTES DE INTERNET | PUBLICACIONES | TRABAJOS DEL ESTUDIANTE |

FUENTES PRIMARIAS

| | | |
|---|---|---|
| 1 | Submitted to Universidad Carlos III de Madrid<br>Trabajo del estudiante | <1% |
| 2 | fferrer.dsic.upv.es<br>Fuente de Internet | <1% |
| 3 | aeca.es<br>Fuente de Internet | <1% |
| 4 | computernetworking11.blogspot.com<br>Fuente de Internet | <1% |
| 5 | ikee.lib.auth.gr<br>Fuente de Internet | <1% |
| 6 | export.arxiv.org<br>Fuente de Internet | <1% |
| 7 | universo-digital.net<br>Fuente de Internet | <1% |
| 8 | Submitted to Universidad Americana<br>Trabajo del estudiante | <1% |
| 9 | Submitted to University of Edinburgh<br>Trabajo del estudiante | <1% |

| | | |
|---|---|---|
| 10 | di002.edv.uniovi.es<br>Fuente de Internet | <1% |
| 11 | Submitted to University of Hertfordshire<br>Trabajo del estudiante | <1% |
| 12 | Submitted to Universidad de Sevilla<br>Trabajo del estudiante | <1% |
| 13 | www.fgg.uni-lj.si<br>Fuente de Internet | <1% |
| 14 | André Lins Gonzalez, Diego Izidoro, Roberto Willrich, Celso A. S. Santos. "Chapter 7 OurMap: Representing Crowdsourced Annotations on Geospatial Coordinates as Linked Open Data", Springer Science and Business Media LLC, 2013<br>Publicación | <1% |
| 15 | reunionesdeestudiosregionales.org<br>Fuente de Internet | <1% |
| 16 | es.patents.com<br>Fuente de Internet | <1% |
| 17 | Submitted to Universidad Anahuac México Sur<br>Trabajo del estudiante | <1% |
| 18 | Submitted to Universitat Politècnica de València<br>Trabajo del estudiante | <1% |
| 19 | library.wmo.int | |

| | | |
|---|---|---|
| | Fuente de Internet | <1% |
| 20 | Submitted to Universidad de Salamanca<br>Trabajo del estudiante | <1% |
| 21 | www.geodesignhub.com<br>Fuente de Internet | <1% |
| 22 | articles.chicagotribune.com<br>Fuente de Internet | <1% |
| 23 | speroni.web.cs.unibo.it<br>Fuente de Internet | <1% |
| 24 | Submitted to Universidad Pedagogica y Tecnologica de Colombia<br>Trabajo del estudiante | <1% |
| 25 | www.electroindustria.cl<br>Fuente de Internet | <1% |
| 26 | www.jiide.org<br>Fuente de Internet | <1% |
| 27 | Submitted to MDCC - Interamerican Campus<br>Trabajo del estudiante | <1% |
| 28 | synchronicity-iot.eu<br>Fuente de Internet | <1% |
| 29 | bienvenidovasquez.blogspot.com<br>Fuente de Internet | <1% |
| | www.yasader.org | |

| | | |
|---|---|---|
| 30 | Fuente de Internet | <1% |
| 31 | M. Alberto Ruiz, J. Francisco Rodriguez, D. Villanueva, H. Estrada, J. Carlos Tellez. "Environmental Monitoring Based on FIWARE: A Medical Case Study", 2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), 2018<br>Publicación | <1% |
| 32 | Submitted to Escuela Superior Politécnica del Litoral<br>Trabajo del estudiante | <1% |
| 33 | Submitted to Escuela Politecnica Nacional<br>Trabajo del estudiante | <1% |
| 34 | www.streetfighteralpha.dvdenlared.com<br>Fuente de Internet | <1% |
| 35 | www.archipielagonoticias.com<br>Fuente de Internet | <1% |
| 36 | www.lexureditorial.com<br>Fuente de Internet | <1% |
| 37 | Submitted to Facultad Latinoamericana de Ciencias Sociales (FLACSO) - Sede Ecuador<br>Trabajo del estudiante | <1% |
| 38 | bdigital.unal.edu.co<br>Fuente de Internet | <1% |

| | | |
|---|---|---|
| 39 | informatica-juridica.com<br>Fuente de Internet | <1% |
| 40 | internal.dstm.com.ar<br>Fuente de Internet | <1% |
| 41 | riunet.upv.es<br>Fuente de Internet | <1% |
| 42 | Submitted to Universidad de Granada<br>Trabajo del estudiante | <1% |
| 43 | www.ruidos.org<br>Fuente de Internet | <1% |
| 44 | Submitted to Universidad Autónoma de Madrid<br>Trabajo del estudiante | <1% |

| | | | |
|---|---|---|---|
| Excluir citas | Activo | Excluir coincidencias | < 5 words |
| Excluir bibliografía | Activo | | |