ISSN 1870-4069

Servicio para consumir datos de transporte público en tiempo real utilizando la plataforma FIWARE

Luis Reyes-González, Alicia Martínez-Rebollar,
Hugo Estrada-Esquivel, Yasmin Hernández-Pérez

[1]Tecnológico Nacional de México/CENIDET,
Departamento de Ciencias Computacionales,
México

{m21ce024, alicia.mr, hugo.ee, yasmin.hp}@ cenidet.tecnm.mx

Resumen. Hoy en día, las grandes ciudades producen una enorme cantidad de información proveniente de múltiples servicios públicos y privados. Una de las fuentes de datos que ha tenido especial interés para el desarrollo de Ciudades Inteligentes es la movilidad urbana. La movilidad ha sido uno de los factores que más ha favorecido el crecimiento económico, pero también ha generado múltiples problemas en las ciudades. Por esta razón, la movilidad también ha sido objeto de investigación por parte de muchos grupos a nivel internacional. Uno de los temas relevantes es el análisis de datos que se generan en estas grandes ciudades en tiempo real desde diferentes agentes y bases de datos. Sin embargo, estos datos se producen en formatos muy diferentes y usualmente sin considerar estándares, lo cual dificulta que los investigadores puedan usar los datos para desarrollar soluciones de movilidad inteligente. En este trabajo de investigación se presenta un servicio de software que permita consumir datos de transporte público, los cuales se encuentran dispersos en diferentes espacios de almacenamiento, estandarizarlos con modelos de datos y finalmente almacenarlos en la nube de FIWARE. De esta forma, desarrolladores y organizaciones interesadas por datos de movilidad podrán consultar el API de la nube de FIWARE y consumir los datos de movilidad ya estandarizados. Estos datos permitirán generar nuevo conocimiento o alguna aplicación que satisfaga necesidades concretas de movilidad.

Palabras clave: Modelo de datos, FIWARE, movilidad urbana, ciudades inteligentes, API.

Service for Consuming Public Transport Data in Real-Time Using the FIWARE Platform

Abstract. Currently, the big cities produce an enormous amount of information coming from several public and private services. Urban mobility is one of the

relevant data sources in the construction of Smart Cities. The mobility is one of the factors that better promote the economic growth, but it has also generated multiple issues in cities. For this reason, mobility has also been the subject of research by many groups internationally. One of the relevant issues is the analysis of data generated, in real-time, in these large cities from different agents and databases. However, these data are produced in very different formats and usually without considering standards. This makes it difficult for researchers in the field of mobility to use them for developing smart mobility solutions. In this research work, a software service is presented that allows consuming public transport data, which are located in different storage spaces, standardizing them with data models and finally storing them in the FIWARE cloud. In this way, developers and organizations interested in mobility data will be able to consult the API of the FIWARE cloud to consume standardized mobility data. These data will allow the generation of new knowledge or an application that satisfies specific mobility needs.

Keywords: Data model, FIWARE, urban mobility, smart cities, API.

1. Introducción

Actualmente, las grandes ciudades son consideradas como las principales fuentes económicas del país. Esta consideración se debe a que en las ciudades se pueden encontrar el mayor número de comercios, aglomeraciones empresariales y servicios, tales como, transporte, tecnología, comunicación vial, telecomunicaciones, etc. [1]. La movilidad en las ciudades es uno de los factores que más ha favorecido el crecimiento económico [2]. Sin embargo, esta misma movilidad se ha convertido en unos de los problemas principales para los habitantes de las ciudades, debido a los problemas que se generan al no contar con un buen sistema de predicción del tráfico, para prevenir aglomeraciones o hacer un mejor uso del transporte público [3].

Uno de los grandes problemas de los nuevos sistemas de movilidad urbana impulsados por enfoques como el Internet de las Cosas es que la información se encuentra aislada en múltiples espacios de almacenamiento, y bajo diferentes modelos de representación [4, 5, 5, 6]. Estos modelos de representación no toman en cuenta el uso de formatos estándares que permitan la reutilización de los datos almacenados.

Actualmente se cuenta con diferentes plataformas especializadas de Internet de las cosas (IoT, por sus siglas en inglés Internet of Things), que cuentan con estándares y protocolos para el manejo y procesamiento de información de IoT a gran escala, con el objetivo de crear y desplegar aplicaciones de IoT inteligentes y administrables [8][9] [10]. Una de estas plataformas de IoT es FIWARE[1].

Estas plataformas de IoT brindan la posibilidad de construir aplicaciones de movilidad inteligente que almacenen sus datos en la nube. Sin embargo, es necesario construir mecanismos que permitan el acceso a los datos, su estandarización y almacenamiento en una nube que permita la reutilización de los datos por

[1] https://fiware-training.readthedocs.io/es_MX/latest/

desarrolladores externos. Este artículo tiene como objetivo presentar un servicio software que permita consumir datos de transporte público en tiempo real y estandarizarlos en la plataforma de FIWARE [11] para su uso en aplicaciones que satisfagan alguna necesidad de movilidad urbana.

En la actualidad, los sistemas de transporte público de grandes ciudades (ej. Ciudad de México), con ayuda de los agentes que conforman la Internet de las Cosas proveen información de algunas de sus unidades de transporte [12]. En este artículo se utiliza los datos provistos por el Sistema de Corredores de Transporte Público de Pasajeros de la Ciudad de México, conocido como Metrobús, los cuales se transforman al modelo datos *"vehicle"* del estándar NGSI [12] y se envían a la nube de FIWARE para proporcionar información estandarizada que pueda ser utilizada para generar aplicaciones inteligentes que ayuden a la mejora del servicio de transporte público.

2. Trabajos relacionados

La movilidad urbana es un tema de gran interés y ha sido abordada desde diferentes perspectivas. En esta sección se presentan algunos de los trabajos más representativos en el área de sistemas de información para movilidad en ciudades. En [13] se analizó la movilidad como uno de los principales riesgos de enfermedades crónicas, al analizar el sedentarismo de las personas. En este trabajo sólo se analizan datos en tiempo real de movilidad para el estudio del sedentarismo, sin embargo, a diferencia del enfoque presentado en este artículo los datos no se estandarizan ni se almacenan un único repositorio.

Otros trabajos de investigación han identificado como problemática la movilidad urbana debido a que el tráfico que se genera en las zonas metropolitanas afecta directamente a la calidad de vida de la sociedad y tiene un alto impacto económico. En [15] los autores proponen el uso de un modelo de agrupación para visualizar la estructura y las características de la movilidad urbana en la ciudad de Sao Paulo, Brasil.

Este modelo, permite encontrar patrones estructurales en la movilidad urbana y revela nuevos conocimientos sobre los sistemas de transporte. Los autores afirman que estos datos de movilidad urbana pueden ayudar a respaldar las actividades de toma de decisiones para contrarrestar esta problemática. En esta propuesta se utilizan los datos obtenidos de la encuesta Origen-Destino de la Compañía de metro de la Ciudad Metropolitana de Sao Paulo. Utilizando el Framework CUBu.

En este artículo no se aborda la estandarización de los datos de movilidad, la cual es un aspecto relevante en nuestra propuesta. En [16] se analiza la movilidad desde el punto de vista del Sistema de Transporte Público de la ciudad de Singapur. El objetivo de esta propuesta es desarrollar una interfaz de análisis visual para presentar y explorar la movilidad de los pasajeros en un sistema de transporte público.

1) Proponen realizar una integración de una solución de tres módulos de visualización.
2) Vista de mapa asíncrona para información geográfica
3) Vista de mapa de flujo isotime para la comparación y manipulación de información temporal efectiva

4) Vista de viajes pares (Origen-Destino) para el análisis visual detallada de los factores de movilidad a lo largo de rutas pares.

La información de este estudio se obtuvo de las tarjetas personalizadas RFID de los pasajeros. El sistema lector de tarjetas registra cada acción de entrada y salida de terminales. La principal diferencia con nuestro enfoque es que utiliza datos históricos y no en tiempo real. Otra propuesta en el análisis de movilidad en el transporte público es la propuesta en [17]. En esta propuesta se propone una solución de software que involucra los datos de movilidad de transporte público del Ayuntamiento de Valencia.

Un aspecto importante de esta solución es la carga masiva de datos GTFS a la plataforma de datos de FIWARE. Como método de prueba, el autor utiliza fichero GeoJSON y con ayuda de la plataforma[2] , consiguió ver la representación gráfica de cada parada de autobuses de la Empresa Municipal de Transporte de Valencia. La diferencia con nuestra propuesta es que es el trabajo no se utilizan datos en tiempo real y solo utiliza modelos GFTS al no considerar datos provenientes de vehículos.

Otras propuestas se han dado a partir de la pandemia del SARS-COV-19 en donde la movilidad humana ha cobrado mayor impacto [18], así como la movilidad en el transcurso del tiempo a partir de esta pandemia [19]. En este caso las fuentes son diferentes que las del trabajo presentado en este artículo, ya que su fuente son datos de redes sociales y no los datos de tiempo real de vehículos o tráfico.

3. Vista general del servicio propuesto

El consumo de los datos del transporte público y su estandarización requiere del análisis de los datos con técnicas de Internet de las Cosas. En este trabajo de investigación se propone el uso de la plataforma FIWARE, la cual permite desarrollar aplicaciones de software relacionadas con Internet de las Cosas. Esta plataforma ofrece componentes abiertos para realizar análisis de grandes volúmenes de datos y la provisión de métricas en tiempo real, manipulación de información de contexto, análisis de eventos en tiempo real, recopilación de información de sensores y acción sobre actuadores [11].

FIWARE ha sido diseñado como una plataforma abierta y estándar, basada en código abierto, para fomentar la creación de los estándares necesarios para desarrollar servicios y aplicaciones inteligentes en diferentes dominios, como ciudades inteligentes, logística, energía, agricultura, industria inteligente, etc. [20]. El caso de estudio utilizado en esta investigación ha sido los metrobuses de la Ciudad de México, los cuales comparten datos acerca de la ubicación en tiempo real de sus unidades de transporte en formato General Transit Feed Specification in Real Time (GTFS-RT)[12].

El formato GTFS-RT es un formato exclusivo para representar servicios de transporte público, por ejemplo, posición de las unidades, información sobre el viaje realizado y alertas de los estatus del viaje o unidad [21]. El servicio propuesto (MeTRoBus IoT) hace uso de la tecnología FIWARE para estandarizar la información generada por las fuentes de datos, aplicando estándares como el NGSI que permite

[2] http://geojson.io/

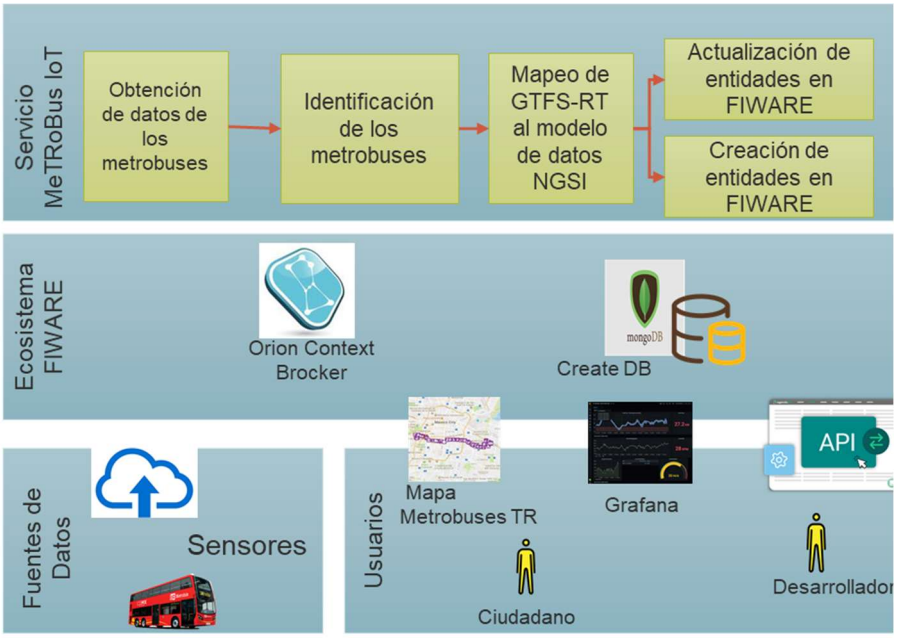


Fig. 1. Vista general de la solución propuesta.

representar objetos de la vida real en modelos de datos genéricos que puedan ser enviados al ecosistema de FIWARE. El componente Orion Context Brocker está preparado para almacenar los últimos valores obtenidos por los sensores de los metrobuses [22]. Por su parte, el componente Quantumleap se utiliza para almacenar el históricos de los datos que se han capturado a lo largo del tiempo [23].

Los datos podrán ser utilizados por desarrolladores que deseen utilizar la información generada por el servicio de transporte de la CDMX. La Figura 1 muestra la vista general de la solución propuesta en este artículo, mostrando las tareas de recolección de datos, identificación de metrobuses, mapeo de datos al estándar NGSI, el envío de datos a la nube de FIWARE y su correspondiente visualización.

4. Servicio MeTroBus IoT

El servicio MeTRoBus IoT para consumir datos de transporte público se encuentra organizado en 5 tareas principales:

– Obtención de los datos de los metrobuses de la Ciudad de México: se realizan peticiones *request* para la obtención de datos de los metrobuses en formato GTFS-RT del servicio provisto por la Secretaría de Movilidad de la Ciudad de México.

– Identificación de los metrobuses: la consulta de los datos de metrobuses genera una lista enorme con todos los datos de todos los metrobuses, por lo cual se

desarrolló un método para recorrer toda la lista de los datos y conservar los atributos necesarios para este proyecto de investigación.

– Mapeo de GTFS-RT a NGSI: con ayuda de los datos de la identificación de los metrobuses y sus respectivos atributos se construyen los JSON para la creación y la actualización de entidades.

– Envío de datos a la plataforma FIWARE: se generan las peticiones POST o PACH para crear o actualizar entidades en la plataforma FIWARE. Estas peticiones contienen el JSON con los datos GTFS-RT mapeados al modelo de datos "vehicle" del estándar NGSI.

– Visualización de datos: permite mostrar los datos en tableros de control o en mapas en los cuales se pueda analizar la movilidad de los metrobuses.

El algoritmo que implementa el servicio del MeTRoBus IoT se muestra en la Figura 2. En las siguientes secciones se brinda mayor detalle de cada uno de estos componentes que conforman el servicio MeTRoBus IoT.

4.1 Obtención de los datos de los metrobuses

Los datos de movilidad utilizados en esta investigación corresponden con los datos de localización de los metrobuses de la CDMX, los cuales se encuentran en el formato Transit Feed Specification in Real Time (GTFS-RT).

La función *consumeGTFS()* nos permite consumir estos datos es la cual se encarga de realizar peticiones *request* (GET) al sitio donde se encuentran alojados los datos, y como respuesta de esta petición se obtiene los datos de ubicación de todos los metrobuses de la Ciudad de México. Esta función se presenta en la Figura 3.

El resultado de esta función *consumeGTFS()* es la lista de cada uno de los metrobuses de la CDMX (más de 750 metrobuses en total).

La figura 4 muestra un ejemplo de la información de un solo metrobús de la lista retornada por la función. Sin embargo, es importante comentar que la lista retornada por la función contiene la información de los más de 750 metrobuses.

4.2 Identificación de los metrobuses

La identificación de la información de cada uno de los metrobuses se lleva recorriendo la lista retornada por la función consume GTFS. La información obtenida de cada metrobús es almacenada en variables para poder mapearse al modelo de datos NGSI. La información GTFS-RT obtenida es la siguiente:

– Position-latitude: brinda la latitud de la ubicación del metrobús.

– Position-longitude: brinda la longitud de la ubicación del metrobús.

– Position-bearing: brinda la dirección hacia dónde va el metrobús (el valor en grados).

```
Algoritmo ServicioMeTroBusIoT
   Mientras TRUE Hacer
         //OBTENCIÓN DE DATOS DE LOS METROBUSES
         //Esta función se encarga de realizar consultar request al sitio de
        //datos delos metrobuses y retorna la lista de los datos de los metrobuses
   feed<-consumeGTFS()
         //IDENTIFICACIÓN DE METROBUSES
         //Se genera el método para que recorra toda la lista de datos de los
        //metrobuses y seleccione los atributos necesarios
      Para i<-0 Hasta tamañoDelFeed Con Paso 1 Hacer
            //Selección de atributos
         datosMetrobusNGSI<-feed.entidad[i]
            //MAPEO DE GTFS-RT AL MODELO DE DATOS NGSI "Creación de entidad"
            //Se mapea los datos GTFS-RT a un JSON de la entidad "vehicle" del
           //estándar NGSI para la creación de entidades
         payloadCreacion<-llenaJSONcreacionEntidad(datosMetrobusNGSI)
            //CREACIÓN DE ENTIDADES
            //Se realiza la petición a los servicios de FIWARE para la creación
           //de la entidad
         estatusResquestPost <- crearEntidad(payloadCreacion)
            Si estatusResquestPost=existe Entonces
            //MAPEO DE GTFS-RT AL MODELO DE DATOS NGSI "Actualización de entidad"
            //Se mapea los datos GTFS-RT a un JSON de la entidad "vehicle" del
           //estándar NGSI para la actualización de entidades
               payloadActualizacion<-llenaJSONactualizacion(datosMetrobusNGSI)
                   //ACTUALIZACIÓN DE ENTIDADES
                   //Se realiza la petición a los servicios de FIWARE para la
                   //actualización de la entidad
               estatusRequestPach<-actualizacionEntidad(payloadActualizacion)
            Fin Si
      Fin Para
         //Espera 30 segundos para repetir el ciclo
      Esperar 30 Segundos
   Fin Mientras
Fin Algoritmo
```

Fig. 2. Algoritmo del Servicio MeTRoBus IoT propuesto.

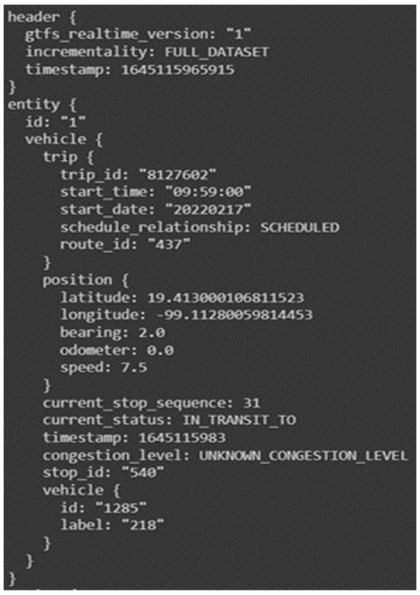
– Position-odometer: brinda el valor de odómetro del metrobús.

– Position-speed: brinda la velocidad del metrobús.

– Current_Status: brinda el estatus del metrobús.

– Timestamp: brinda la hora y fecha de cuando se realiza la consulta.

– Vehicle-label: brinda el número económico de la unidad.



```
header {
  gtfs_realtime_version: "1"
  incrementality: FULL_DATASET
  timestamp: 1645115965915
}
entity {
  id: "1"
  vehicle {
    trip {
      trip_id: "8127602"
      start_time: "09:59:00"
      start_date: "20220217"
      schedule_relationship: SCHEDULED
      route_id: "437"
    }
    position {
      latitude: 19.413000106811523
      longitude: -99.11280059814453
      bearing: 2.0
      odometer: 0.0
      speed: 7.5
    }
    current_stop_sequence: 31
    current_status: IN_TRANSIT_TO
    timestamp: 1645115983
    congestion_level: UNKNOWN_CONGESTION_LEVEL
    stop_id: "540"
    vehicle {
      id: "1285"
      label: "218"
    }
  }
}
```


Fig. 4. Ejemplo de un segmento de la lista de datos de los metrobuses.

El mapeo de los datos obtenidos de alguna fuente de datos debe ser estandarizada para poder ser consumida por cualquier aplicación o desarrollador que desee utilizarla. En el caso de estudio de esta investigación, la información de los metrobuses se encuentra en el formato GTFS-RT, por lo que se requiere mapearse al modelo de datos *"vehicle"* definido para la plataforma FIWARE. La Tabla 1 muestra como son mapeadas las variables del modelo GTFS-RT al modelo de datos de datos "*vehicle*" especificado en el estándar NGSI de FIWARE.

La plataforma FIWARE permite conservar el dato más actual obtenido y a partir de estos datos generar una base de datos histórica. Debido a este comportamiento, fue necesario generar una estructura JSON exclusiva para la actualización de entidades.

4.3 Creación y actualización de entidades en FIWARE

Los métodos que utiliza la plataforma FIWARE para poder consultar, guardar o actualizar entidades se basan en peticiones *request*. Los datos obtenidos del sistema de metrobuses y que han sido estandarizados utilizando un modelo de datos debe enviarse a la nube de FIWARE a través del componente Orion Context Broker (OCB). Una vez que los datos se encuentran en el OCB pueden ser consultados utilizando servicios REST. Los métodos para la creación y actualización de entidades son las siguientes:

– GET: consulta de información.

– POST: creación de entidades.

– PACH: actualización de ciertos atributos de la entidad.

Table 1. Variables del modelo GTFS-RT mapeadas a las variables del modelo NGSI de FIWARE.

Datos GTFS-RT		Entidad Vehicle (NGSI)	
Variable	Descripción	Variable	Descripción
speed	Representa la velocidad que tiene el vehículo representado en m/s.	speed	Representa la velocidad que tiene el vehículo. Esta información se requiere en km/h.
odometer	Indica la distancia total en metros recorrida por el vehículo desde su producción inicial.	mileageFromOdometer	Indica la distancia total en kilómetros recorrida por el vehículo desde su producción inicial.
bearing	Representa la orientación en grados del vehículo. Son valores en decimal donde 0 es norte y 90 es este.	heading	Representa la orientación de grados del vehículo. Son valores decimales donde 0 es norte y gira en sentido de las agujas del reloj.
latitud longitude	Representa la ubicación del vehículo. Brinda dos valores decimales separados, un valor para latitud y otro para longitud.	location	Representa la ubicación del vehículo. Requiere los datos en un punto GeoJSON donde contiene los grados latitud y longitud.
current_status	Representa el estado del vehículo, desde el punto de vista ser servicio prestado. Posibles estados: INCOMING_AT: el vehículo está a punto de llegar a la parada. STOPPED_AT: el vehículo está detenido en la parada. IN_TRANSIT_TO: el vehículo salió de la parada anterior y está en tránsito.	servicesStatus	Representa el estado del vehículo. Estado que más se adapta: onRoute: El vehículo está realizando una misión. Se pueden agregar modificadores separados por comas para indicar qué misión está entregando actualmente el vehículo. Ejemplo: *"onRoute, STOPPED_AT"*.
timestamp	Indica el tiempo en que se adquiere la información en formato timestamp.	timestamp	Indica el tiempo en que se adquiere la información en formato ISO 8601.
label	Representa la identificación del vehículo en el contexto de una flota de vehículos	fleetVehicleId	Representa la identificación del vehículo en el contexto de una flota de vehículos

Una vez que se cuenta con la especificación JSON de la entidad *"vehicle"* del estándar NGSI con los datos de los metrobuses se procede a realizar la petición POST o PACH ya sea para crear la entidad o actualizarla. En la figura 5 se puede observar el método que se utiliza para generar estas peticiones. Para poder identificar en qué momento crear o actualizar una entidad, se toma como referencia la respuesta del *request*. Estos son algunas respuestas:

- Request timed out: el servidor al cual se le hace la petición no contesta.
- 400: no se encuentra la URL solicitada.
- 200: petición GET bien realizada.
- 201: petición POST, creación de entidad de forma correcta.
- 422: petición POST, entidad no creada, la entidad ya existe.
- 204: petición PACH, actualización de entidad de forma correcta.

La lógica de nuestro método para la creación o actualización de entidades es: "Crear la entidad en la liga[3] con los datos del metrobús en turno, si la respuesta del request es igual a 422 entonces no debe crearse sino actualizarse la entidad con la liga[4] con los mismos datos del metrobús en turno".

La creación de entidades permite que los valores obtenidos de los metrobuses se almacenen en el componente Orion Context Broker (OCB) de la plataforma FIWARE y puedan por tanto ser consultados por otros desarrolladores.

Para visualizar los datos en mapas se desarrolló de una aplicación web que consume datos del Orion Context Broker. Esta aplicación web es capaz de representar por lo menos un metrobús en un espacio geográfico de un mapa con sus datos tiempo real. Tomando en cuenta que los datos de los metrobuses se actualizan en el OCB cada 30 segundos, la aplicación web se automatiza para que pueda consumir datos nuevos en este intervalo de tiempo y representarlos en el mapa. Con la automatización de la aplicación web se podrá observar el recorrido que el metrobús realiza en tiempo real.

4.4 Visualización de los datos

En este proyecto de investigación se proponen dos formas para la visualizar los datos guardados en la plataforma FIWARE: tableros de control y mapas. En el caso de los tableros de control, la herramienta Quantumleap de FIWARE contempla el componente software Grafana para la visualización de datos históricos de forma gráfica. Para lograr la visualización se crea la conexión entre Grafana y CrateDB. Con ayuda de esta conexión se generan las consultas SQL necesarias para poder visualizar datos en un dashboard con diferentes gráficas o tablas.

[3] *http://localhost:1026/v2/entities/*

[4] *http://localhost:1026/v2/entities/vehicle:MetroBus:NUMERODEUNIDAD/attrs*

```
#URL para creación de entidades
urlCreacionEntidad = "http://"+ str(ipServer) +":1026/v2/entities/"
payloadCreacionEntidad=llenaJSONcreacionEntidad(idVehicle,
                                                labelVehicle,
                                                latitudeVehicle,
                                                longitudeVehicle,
                                                timeConsulta,
                                                statusVehicle,
                                                speedVehicle,
                                                bearingVehicle,
                                                odometerVehicle)

estatusResquestPOST = requests.post(url= urlCreacionEntidad,
                                    data=payloadCreacionEntidad,
                                    headers= headers_string)
#imprime el estado del POST
print("Estado POST: {}" .format(estatusResquestPOST.status_code))

if(estatusResquestPOST.status_code==422):
    #URL para actualización de entidades
    urlActualizacionEntidad = "http://"+ str(ipServer) +":1026/v2/entities/vehicle:MetroBus:"+str(labelVehicle)+"/attrs"
    payloadActualizacionEntidad=llenaJSONactualizacion(labelVehicle,
                                                       latitudeVehicle,
                                                       longitudeVehicle,
                                                       timeConsulta,
                                                       statusVehicle,
                                                       speedVehicle,
                                                       bearingVehicle,
                                                       odometerVehicle)
    estatusResquestPACH = requests.patch(url= urlActualizacionEntidad,
                                         data=payloadActualizacionEntidad,
                                         headers= headers_string)
    #imprime el estado del PACH
    print("Estado PACH: {}" .format(estatusResquestPACH.status_code))
```

Fig. 4. Funciones para llenar el JSON de creación o actualización de entidades.

Fig. 5. Visualización del recorrido del metrobús número 9407 en el mapa.

Fig. 6. Visualización del recorrido del metrobús número 1208 en el mapa.

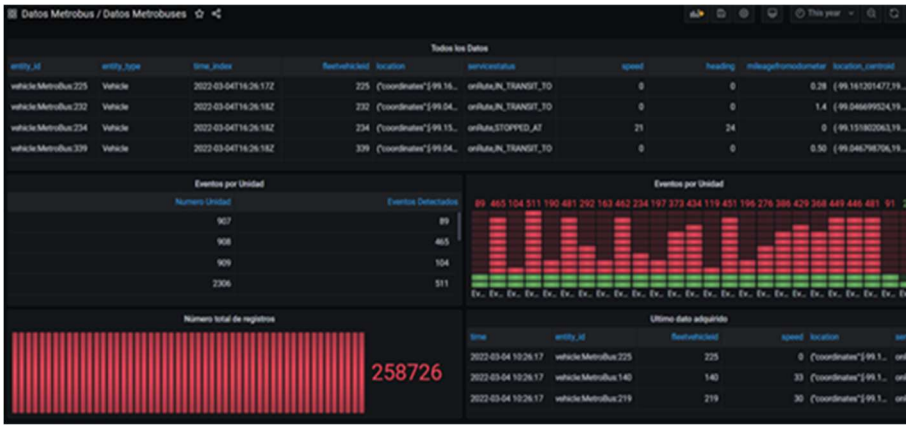


Fig. 7. Visualización de datos históricos en Grafana.

5. Pruebas al servicio MeTRoBus IoT

El servicio MeTRoBus IoT se encuentra automatizado para consumir, identificar, mapear y crear o actualizar entidades en FIWARE cada 30 segundos de los aproximadamente 750 metrobuses de la Ciudad de México que cuentan con monitoreo GPS. Como resultado del proceso, nuestra nube de FIWARE contiene datos en tiempo real y datos históricos de los metrobuses.

Esta información puede ser utilizada para el análisis de la movilidad en la ciudad. La plataforma FIWARE brinda APIs de acceso rápido a los datos, por lo que la mejor forma de validar el funcionamiento del servicio MeTRoBus IoT es visualizando de forma gráfica los datos alojados en la plataforma FIWARE. Por esa razón, se desarrolló una aplicación web que representa la posición de metrobuses en un mapa.

Esta aplicación consume los datos en tiempo real de los metrobuses de la Ciudad de México que están alojados en el Orion Context Broker de la plataforma FIWARE. En la figura 6 se muestra la trayectoria que recorrió el metrobús número 9407 en un tiempo de monitorio de 10 minutos. La figura 7 muestra el recorrido del metrobús número 1208, con un tiempo de monitoreo de 13 minutos.

Como ejemplo adicional de la ejecución del servicio propuesto, se genera una grabación que muestra el funcionamiento del servicio MeTRoBus IoT. En la siguiente liga[5] se puede apreciar que, mientras el servicio MeTRoBus IoT se encuentra en ejecución, la posición de los metrobuses cambia en el mapa. A diferencia de las primeras pruebas mostradas en el video, en esta ocasión no se muestra el recorrido, sino solo la posición los metrobuses en tiempo real.

Para la visualización del histórico de los datos de los metrobuses se utiliza la herramienta Grafana. En la Figura 8 se observa uno de los paneles de control con esta

[5] https://www.youtube.com/watch?v=x8WW5Z6pwYU

herramienta para el proyecto presentado en este artículo. Cada una de estas tablas o gráficas se generaron a partir de consultas tipo SQL.

6. Conclusiones y trabajo a futuro

El servicio MeTRoBus IoT contribuye a la estandarización de entidades de dispositivos IoT para datos de movilidad urbana. El servicio propuesto permite obtener datos del servicio de Metrobuses de la CDMX y estandarizarlos con un modelo de datos. Estos datos ya estandarizados son alojados en la plataforma FIWARE la cual brinda APIs de acceso rápido a datos para su posterior uso por parte de desarrolladores.

Las APIs generadas en este proyecto facilitan el acceso a los datos de movilidad con el objetivo de generar nuevo conocimiento que ayude a la planificación de la movilidad urbana o a la necesidad de generar aplicaciones que satisfagan alguna necesidad de movilidad urbana. La herramienta Grafana permite tener una visualización de los datos de una forma simple e intuitivo para el usuario. Esta visualización también facilita el análisis de los datos de movilidad. Como trabajos futuros se pretende consumir los datos de la plataforma HEREMaps.

Esta plataforma brinda información acerca de la congestión vehicular de la Ciudad de México en tiempo real. Estos datos también tendrán un proceso de identificación de segmentos de ciudades con sus respectivos atributos para mapear los datos al modelo de datos *"TrafficFlowObserved"* provisto por FIWARE. Los datos de tráfico capturados a partir de HEREMaps también deberán estar alojados en la plataforma FIWARE.

La información de los metrobuses CDMX y de HEREMaps permitirá contar con datos, en tiempo real y datos históricos, de la movilidad en la Ciudad de México en un formato estandarizado. Este espacio de datos de movilidad podrá ser de utilidad para la planificación de movilidad urbana en la ciudad.

Referencias

1. REDEUSLAC Economía Urbana. (2022)
2. Montezuma, R.: Ciudad y transporte: la movilidad urbana. (2003)
3. Colchado Flores, I. C. F.: La movilidad urbana en la Ciudad de México: un problema complejo. Centro de Ciencias de la Complejidad (2017)
4. Pública, A. D. D. I.: Portal de Datos Abiertos de la CDMX. Portal de Datos Abiertos de la CDMX (2021)
5. Google. Referencia de GTFS Real-time | Transporte en tiempo real |. Google Developers. (2021)
6. FIWARE. Modelos de datos - Aprende FIWARE en español. Aprendiendo FIWARE (2020)
7. HERE Maps. Documentation, Code Examples and API References. HERE Developer. (2022)
8. KaaIoT. Enterprise IoT Platform with Free Plan | Kaa. Kaa IoT Platform (2022)
9. Bustamante, A. L.: Thinger.io plataform. Thinger.Io (2020)
10. de Panfilis, G.: About FIWARE. (2022)

11. FIWARE. La plataforma FIWARE - Aprende FIWARE en español (2022)
12. Metrobuses CDMX. Datos Abiertos. metrobús (2022)
13. FIWARE. Vehicle - FIWARE DataModels. (2019)
14. Wang, Y., König, L. M., Reiterer, H.: A smartphone app to support sedentary behavior change by visualizing personal mobility patterns and action planning (SedVis): Development and pilot study. JMIR Formative Research, vol. 5, no. 1 (2021)
15. Martins, T. G., Lago, N., De Souza, H. A., Santana, E. F. Z., Telea, A., Kon, F.: Visualizing the structure of urban mobility with bundling: A case study of the city of São Paulo. Anais do Workshop de Computação Urbana. Sociedade Brasileira de Computação – SBC (2020)
16. Zeng, W., Fu, C.-W., Arisona, S. M., Erath, A., Qu, H.: Visualizing mobility of public transportation system. IEEE Transactions on Visualization and Computer Graphics, vol. 20, no. 12, pp. 1833–1842 (2014)
17. Alemany, J. L. P.: Creación de un repositorio de contenidos para transporte público urbano con Fiware. Universitat Politécnica de Valéncia Escuela Técnica Superior de Ingeniería Geodésica, Cartográfica y Topográfica, Valencia (2019)
18. Huang, X., Li, Z., Jiang, Y., Ye, X., Deng, C., Zhang, J., Li, X.: The characteristics of multi-source mobility datasets and how they reveal the luxury nature of social distancing in the U.S. during the COVID-19 pandemic (2020)
19. Jeffrey, B., Walters, C. E., Ainslie, K. E. C., Eales, O., Ciavarella, C., Bhatia, S., Riley, S.: Anonymised and aggregated crowd level mobility data from mobile phones suggests that initial compliance with COVID-19 social distancing interventions was high and geographically consistent across the UK. Wellcome Open Research, vol, 5, no. 170 (2020)
20. Telefónica. FIWARE, el estándar que necesita el IoT (2017)
21. Google. Descripción general de GTFS Realtime | Transporte en tiempo real. Google Developers (2022)
22. FIWARE. Orion Context Broker (OCB) - Aprende FIWARE en español (2020)
23. FIWARE. QuantumLeap – QuantumLeap (2020)