Recolección en tiempo real de datos de telemetría y rastreo en el transporte público

Real-time Collection of Telemetry and Tracking Data in Public Transportation

Fabián Abarca Calderón
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica
fabian.abarca@ucr.ac.cr
0000-0001-8408-806X

Adrián Cordero Méndez
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica
adrian.corderomendez@ucr.ac.cr
0009-0001-6284-1134

Edson Joao Murillo Mamani
Escuela de Ingeniería
Universidad de Costa Rica
edson.murillo@ucr.ac.cr
0009-0000-0769-1156

Palabras clave: *API, ARC-IT, arquitecturas tecnológicas, GTFS, transporte público inteligente*

RESUMEN

Este artículo propone el diseño de un servidor web y la especificación de una interfaz de programación de aplicaciones (API) llamada *Databús* v1.0 para la recolección de datos de telemetría y rastreo vehicular en un sistema inteligente de transporte público, desarrollados a partir de la investigación de estándares, tecnologías y modelos de datos y de comunicación aptos para el transporte público y basados en principios de diseño orientados a la interoperabilidad y la neutralidad tecnológica.

Un subconjunto de la propuesta es implementado en el plan piloto de un sistema de información para el servicio de buses del campus de la Universidad de Costa Rica, con el objetivo de proveer datos GTFS *Realtime*, una especificación de datos en tiempo real para uso de aplicaciones de planificación de viajes intermodales y otros medios de difusión, como páginas web y pantallas.

Como complemento al sistema, diseñamos también la aplicación móvil *Databús* para uso de los conductores de los autobuses, como alternativa básica y de bajo costo de un equipo de telemetría y rastreo.

ABSTRACT

This article proposes the design of a web server and the specification of an Application Programming Interface (API) called Databús v1.0 for the collection of telemetry and vehicle tracking data in an intelligent public transportation system. These are developed based on the research of standards, technologies, and data and communication models suitable for public transportation, and based on design principles oriented towards interoperability and technological neutrality.

A subset of the proposal is implemented in the pilot plan of an information system for the bus service on the campus of the University of Costa Rica, with the aim of providing GTFS Realtime, a real-time data specification for use in intermodal trip planning applications and other distribution media, such as web pages and screens.

As a complement to the system, we also designed the Databús mobile application for use by bus drivers, as a basic and low-cost alternative to telemetry and tracking equipment.

I. INTRODUCCIÓN

Los sistemas inteligentes de transporte público utilizan tecnologías de información y comunicación para facilitar la recolección y uso de datos masivos provenientes de equipos y sensores en vehículos e infraestructura [1]. Estos datos son utilizados para mejorar la operación del sistema, su gestión administrativa, su planificación, su regulación y, sobre todo, para mejorar la experiencia de las personas usuarias del servicio [2].

Como sistema, el transporte público inteligente es una industria de alta tecnología [3] que opera sobre la base de sensores [4], redes de telecomunicaciones [5] y técnicas modernas de procesamiento y análisis de datos [6], entre otros componentes tecnológicos.

Este artículo presenta el diseño de un servicio de recolección de datos de telemetría y rastreo vehicular en tiempo real de autobuses, que es una de las diversas plataformas tecnológicas que habilitan los servicios de transporte público inteligente. Para esto implementamos el servidor web llamado `realtime`, diseñamos la especificación de la interfaz de programación de aplicaciones (API) llamada *Databús v1.0* y especificamos la aplicación móvil llamada *Databús* para celulares Android y iOS.

El diseño del servidor y su API abarca una gama amplia de categorías de datos en tiempo real para vehículos de transporte público, elegidos a partir de la arquitectura de referencia para transporte colaborativo e inteligente del Departamento de Transporte de Estados Unidos (ARC-IT, *Architecture Reference for Cooperative and Intelligent Transportation*) [7] que ofrece una generalidad suficiente para cualquier implementación típica.

Por su parte, la aplicación móvil *Databús* es una implementación limitada pero práctica y de bajo costo para suplir datos a un sistema de información para personas usuarias con la especificación GTFS *Realtime*. *Databús* puede ser utilizado en celulares Android y iOS de bajas prestaciones, y utilizado con mínimas modificaciones en los autobuses y con poca intervención de los conductores del vehículo.

La filosofía de diseño favorece el uso de tecnologías y estándares abiertos, una arquitectura orientada hacia la interoperabilidad, la posibilidad de la participación de múltiples proveedores tecnológicos y la posibilidad de una implementación gradual de creciente complejidad. Todas estas características, a nuestro criterio, son deseables en el contexto de Costa Rica, donde no ha sido posible hasta ahora hacer una inversión del tamaño requerido para crear un sistema "monolítico" de transporte público inteligente con beneficios tangibles, de amplio alcance y durables para las personas usuarias, a pesar de varios esfuerzos previos [8][9].

II. METODOLOGÍA

La metodología de este proyecto consiste en una investigación bibliográfica sobre estándares y arquitecturas de sistemas inteligentes de transporte público. También abarca la identificación de necesidades de los servicios conectados a este sistema y relevantes para el transporte público, la investigación y selección de modelos de datos de transporte público en tiempo real y sus detalles, y la investigación y selección de un modelo de comunicación entre el equipo del vehículo y el servidor en tiempo real. Además, se incluye el diseño del API (modelo de comunicación elegido) siguiendo una arquitectura REST conforme a la especificación OpenAPI v3.0, el diseño del servidor para implementar el diseño creado, el diseño funcional de la aplicación móvil para la implementación de bajo costo del equipo del vehículo, y finalmente la discusión sobre el diseño presentado y el trabajo futuro.

III. ARQUITECTURA DEL SERVIDOR EN TIEMPO REAL

El diagrama tecnológico del sistema del plan piloto completo está en la figura 1, con la implementación de este artículo en el cuadro resaltado.

Para diseñar la arquitectura de este subsistema de recolección de datos en tiempo real, es necesario seleccionar un *modelo de datos de transporte público en tiempo real*, es decir, aquellos datos que son relevantes para el servicio y que son procesados en tiempo real, y un *modelo de comunicación* entre el vehículo y el servidor, que sea apropiado para la naturaleza de los datos, los requerimientos del servicio y las posibilidades tecnológicas, como los equipos y redes de telecomunicaciones disponibles.

A. Modelos de información de datos de transporte público

La selección de los datos, según aparece en la especificación del API en la sección IV.A, viene de dos fuentes principales, que son referencias globales en el tema de sistemas de información para personas usuarias y transporte público inteligente: GTFS y ARC-IT.

A.1. GTFS

La especificación del suministro de datos abiertos de transporte público (GTFS, *General Transit Feed Specification*) es un estándar *de facto* para la distribución de información del servicio entre aplicaciones de planificación de viajes intermodales, como Google Maps o Moovit, y otros servicios como pantallas informativas o inclusive para labores de regulación del servicio e investigación académica. GTFS tiene dos grandes categorías de datos:

GTFS Schedule Presenta datos "estáticos", como los datos de la agencia operadora, las rutas, las paradas, las trayectorias geoespaciales, las tarifas y los horarios, entre otras informaciones. Esto constituye la información básica que una persona usuaria debería conocer para utilizar eficientemente el servicio.

GTFS Realtime Actualiza la información de la posición de los vehículos, la estimación de tiempos de llegada a las paradas y las alertas del servicio. A pesar de su nombre, debe ser considerado de tiempo *casi* real pues, estrictamente hablando, es actualizado a una frecuencia regular con un periodo del orden de decenas segundos en donde envía la información recopilada de todos los vehículos durante ese intervalo.

En particular, GTFS *Realtime* entrega los siguientes conjuntos de datos, llamados *entidades*:

`VehiclePositions` Entidad que especifica información como la posición geoespacial del vehículo, su nivel de ocupación de pasajeros y el nivel de congestión vial, entre otros. Estos datos son recopilados y suministrados por el *equipo electrónico* en los vehículos.

`TripUpdates` Entidad que actualiza los tiempos estimados de llegada a las paradas para cada viaje y avisa cancelaciones o cambios de ruta. Estos datos son calculados y suministrados por el *servidor* en tiempo real.

`Alerts` Entidad que informa sobre afectaciones al servicio, como cambios de paradas, interrupciones de la vía pública, eventos inesperados y otros. Estos datos son editados y suministrados por la *agencia operadora* del servicio, con intervención humana.

GTFS ofrece el detalle de todos los datos por enviar, incluyendo el nombre, el formato exacto de dato que admite, la lista de valores permitidos, etc.

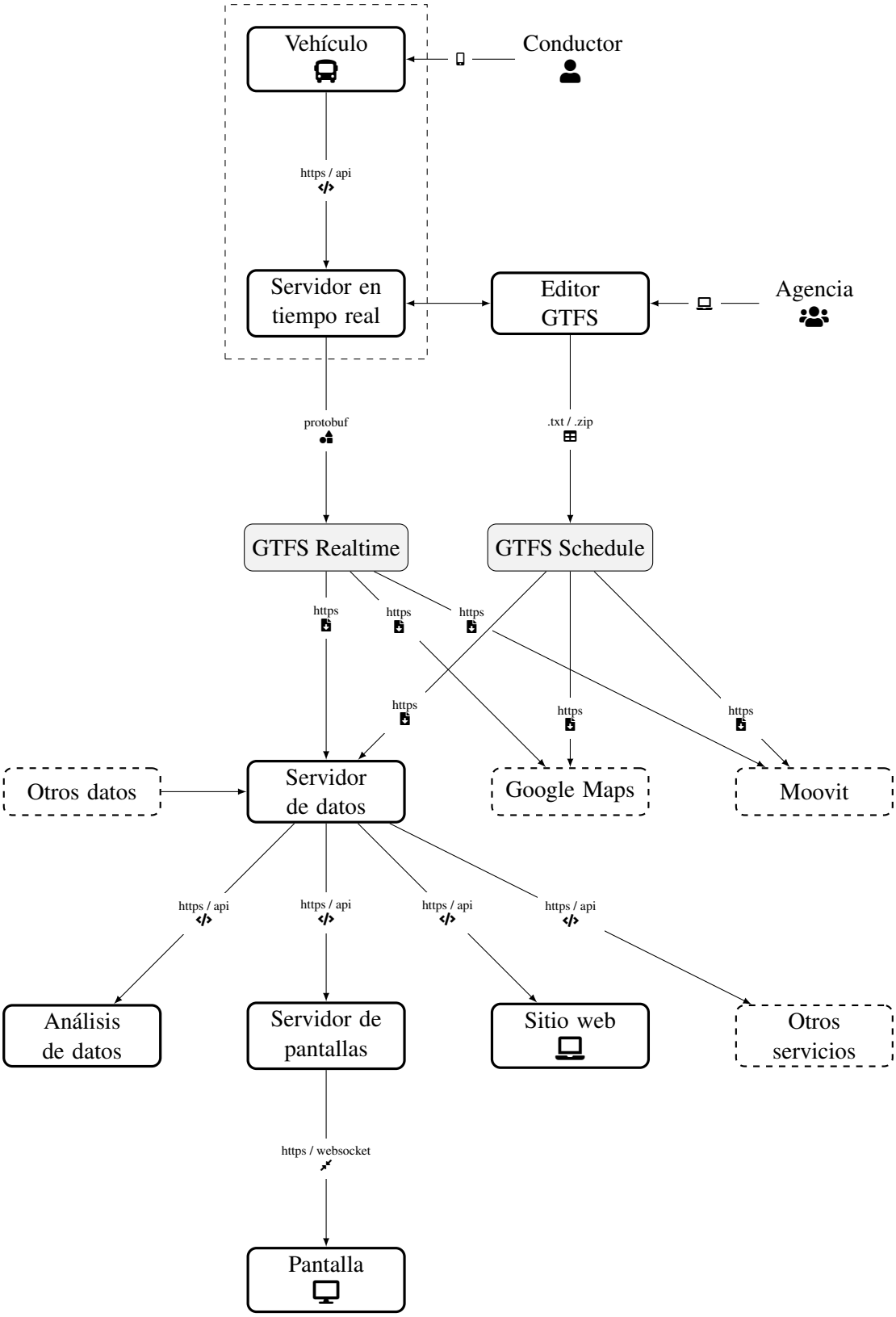

Fig. 1: Diagrama tecnológico de la implementación del sistema de información para ***b*UCR**

A.2. ARC-IT

La arquitectura de referencia de transporte colaborativo e inteligente del Departamento de Transporte de los Estados Unidos (ARC-IT, *Architecture Reference for Cooperative and Intelligent Transportation*) es un modelo amplísimo desde la perspectiva de la ingeniería de sistemas que describe las distintas perspectivas, motivaciones, componentes, actores, tecnologías, interfaces y muchos otros aspectos de un sistema inteligente de transporte. Según su descripción, es el producto de varias décadas de experiencia de administradores, usuarios, desarrolladores tecnológicos, comunidades y otras partes involucradas en el transporte.

En su núcleo están los paquetes de servicio (*service packages*), que son aplicaciones para proveer soluciones a problemas específicos. Por ejemplo, en el área de transporte público existen los paquetes de servicio **PT01** Rastreo de Vehículos de Transporte Público y **PT08** Sistemas de Información de Pasajeros, entre otros.

Para efectos de nuestro diseño en este trabajo, ARC-IT solamente llega hasta el nivel de detalle de una, así llamada, *tipleta* de comunicación, en la que especifica el origen del mensaje, el destinatario y el tipo del mensaje, pero sin dar ninguna especificación sobre el contenido. Por ejemplo, una tripleta es: equipo electrónico del vehículo (fuente) $\rightarrow$ datos de emisiones (mensaje) $\rightarrow$ centro de administración del transporte público (destino).

B. Modelos de comunicación vehículo / servidor

La forma de comunicación entre el vehículo y el servidor es una elección relevante. En general, será aceptado que la comunicación será sobre la red celular, que permite conexiones TCP/IP, sin embargo, es necesario definir el modelo de comunicación propiamente.

Solicitud/respuesta (*request/response*) En este paradigma un *cliente* hace una *solicitud* a un *servidor*, el cual envía de vuelta una *respuesta* [10]. HTTP es el ejemplo más conocido. Los web API operan sobre HTTP y son la opción más popular en la actualidad para conectar servicios. Es importante anotar que la comunicación siempre es iniciada por el cliente, lo que pone algunas limitaciones para las comunicaciones en tiempo real desde el servidor hacia el cliente, que estará condicionada a una estrategia de *polling* (sondeo) regular para recibir nuevas notificaciones.

Orientada a eventos (*event-driven*) En la arquitectura orientada a eventos (EDA, *Event-driven Architecture*) la generación de un evento puede desencadenar la invocación de uno o más servicios desacoplados [11]. Es utilizada en varios dominios como redes de sensores, control de sistemas en tiempo real, monitoreo de salud y otros [12].

Publicación/suscripción (*pub/sub*) En este paradigma un *publicador* genera *mensajes* sobre eventos para ser propagados a *suscriptores* que reciben la información por medio de *canales* de *tópicos* específicos distribuidos por un *intermediador de mensajes*. Es típicamente utilizado en sistemas de notificación de eventos [13] como MQTT (*Message Queuing Telemetry Transport*) o AMQP (*Advanced Message Queuing Protocol*).

El modelo de comunicación elegido para el sistema es un web API siguiendo la especificación OpenAPI v3.0.

IV. RESULTADOS Y ANÁLISIS

El servidor `realtime` está implementado con un sistema operativo **Linux Ubuntu** 22.04 LTS, la base de datos geoespacial **PostgreSQL** 14 con la extensión **PostGIS**, el lenguaje de programación **Python** 3.11, la plataforma de *backend* **Django** v5.0, el módulo Django **REST Framework** v3.15.1 para el API, el administrador y planificador de tareas **Celery** v5.3.6 y el intermediador de mensajes **Redis** v5.0.1.

Este es un proyecto de código abierto. El repositorio está disponible en GitHub en la dirección https://github.com/fabianabarca/realtime y su documentación está en https://fabianabarca.github.io/realtime. El proyecto está alojado en el servidor de prueba https://realtime.bus.ucr.ac.cr.

A. Especificación del API Databús v1.0

El API fue diseñado siguiendo las especificaciones de OpenAPI v3.0., el estándar *de facto* en la industria para diseñar API tipo REST. Los puntos finales o *endpoints* para intercambio de datos están descritos desde la tabla **??** hasta la **??**, en la sección **??**. La documentación completa del web API está en la dirección https://realtime.bucr.digital/api/docs/.

Patrones de actualización Los mensajes enviados al servidor por medio del API tienen una actualización que varía según el tipo de dato. Cuando son causados por un evento, el mensaje es enviado al momento de ocurrencia. Por ejemplo, el abordaje o descenso de un pasajero. Otros tipos de datos serán agrupados según su frecuencia de actualización regular, que puede variar entre unas cuantos segundos para la posición, unos cuantos minutos para las condiciones del vehículo como el nivel de combustible, o unas cuantas horas para la cuantificación de las emisiones de gases. En las tablas de descripción de los *endpoints* hay una indicación del tipo de actualización y la frecuencia de muestreo o bien el evento activador.

B. Funcionalidad de la aplicación móvil Databús

Para el plan piloto es necesaria una alternativa de bajo costo de funcionalidad limitada como equipo de rastreo vehicular para poder habilitar, al menos, la creación de GTFS *Realtime*. Una *app* puede ser usada con:

Interfaz de registro de equipo Conecta la *app* con el servidor, vinculado a un vehículo.

Interfaz de registro de operador Reporta el inicio de labores de un conductor en el vehículo.

Interfaz de inicio de viaje Reporta la ruta, dirección, fecha y hora de salida de un viaje. Al iniciar el viaje la *app* hará un reporte cada 10 segundos de su posición, velocidad y orientación.

Interfaz de ocupación Habilita el reporte de las categorías de ocupación en GTFS de parte del conductor, que no es una medición directa sino una estimación que puede hacer en cada parada.

Las categorías de ocupación son: vacío, muchos asientos disponibles, pocos asientos disponibles, solo espacio de pie, espacio de pie estrujado, completo, no acepta pasajeros, no hay datos disponibles, no abordable.

CONCLUSIONES

1. En este trabajo realizamos una investigación bibliográfica para determinar cuáles estándares, modelos de información, arquitecturas tecnológicas y modelos de comunicación son relevantes para la recopilación de datos de telemetría y rastreo en tiempo real en el transporte público.

2. Hicimos una selección de los datos, la arquitectura y el protocolo de comunicación para diseñar un sistema de telemetría y rastreo con base en criterios de interoperabilidad y neutralidad tecnológica.

3. El API diseñado tiene 21 puntos finales (*endpoints*) para intercambio de datos, junto con un esquema detallado de los parámetros de consulta y los datos tanto en la solicitud (*request*) como la respuesta (*response*).

4. La especificación abierta del API *Databús v1.0* es de uso general y puede ser utilizado en Costa Rica con cualquier equipo de telemetría y rastreo instalado en los autobuses. La especificación completa está en la dirección https://datahub.bucr.digital/api/docs/.

5. La aplicación móvil *Databús* descrita ofrece una alternativa limitada pero práctica y de bajo costo para una implementación inicial del equipo electrónico del autobús. Esto ofrece la posibilidad de una implementación gradual sin altos costos iniciales.

6. La investigación reveló que, a pesar de la relevancia de este tipo de sistemas tecnológicos para todas las partes involucradas en el sistema de transporte público, el Programa para un Sistema Integrado de Transporte Público Masivo para la GAM (SITGAM 2020 - 2035) [14] –que actualmente constituye una referencia importante para el desarrollo del sector en el país– no menciona explícitamente en sus 41 indicadores nada relacionado con las tecnologías de información y comunicación en el transporte público, revelando un vacío en la política pública sobre este tema.

REFERENCIAS

[1] M. Alam, J. Ferreira y J. Fonseca, eds., *Intelligent Transportation Systems, Dependable Vehicular Communications for Improved Road Safety*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-28183-4.

[2] W. Barfield y T. A. Dingus, *Human factors in intelligent transportation systems*. Psychology Press, 1998. DOI: 10.4324/9781315806624.

[3] MDIP. «Modernizing transit technology with open standards.» (15 de ago. de 2023), dirección: https://www.interoperablemobility.org/.

[4] J. Guerrero-Ibáñez, S. Zeadally y J. Contreras-Castillo, «Sensor technologies for intelligent transportation systems,» *Sensors*, vol. 18, n.º 4, pág. 1212, 2018. DOI: 10.3390/s18041212.

[5] M. Tubaishat, P. Zhuang, Q. Qi e Y. Shang, «Wireless sensor networks in intelligent transportation systems,» *Wireless communications and mobile computing*, vol. 9, n.º 3, págs. 287-302, 2009. DOI: 10.1002/wcm.616.

[6] Y. Wang, D. Zhang, Y. Liu, B. Dai y L. H. Lee, «Enhancing transportation systems via deep learning: A survey,» *Transportation Research Part C: Emerging Technologies*, vol. 99, págs. 144-163, feb. de 2019. DOI: 10.1016/j.trc.2018.12.004.

[7] U.S. Department of Transportation, «Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT) v9.0, The National ITS Reference Architecture,» U.S. Department of Transportation, inf. téc., 2020. dirección: http://www.arc-it.net/.

[8] M. Cordero Parra, «Sistemas inteligentes permitirán vigilancia y control del transporte público,» *Semanario Universidad*, 26 de abr. de 2018. dirección: https://semanariouniversidad.com/ultima-hora/sistemas-inteligentes-permitiran-vigilancia-y-control-del-transporte-publico/.

[9] J. A. Céspedes, «Nueva app de Incofer informa sobre horas de llegada y posibles retrasos de tren,» *La Nación*, 19 de jul. de 2021. dirección: https://www.nacion.com/el-pais/servicios/nueva-app-de-incofer-informa-sobre-horas-de/77VIHEFUWBD5RCOTXFMAKL4XO4/story/.

[10] S. Kumar, «A Review on Client-Server based applications and research opportunity,» *International Journal of Recent Scientific Research*, vol. 10, n.º 7, págs. 33 857-3386, 2019.

[11] L. Lan, B. Wang, L. Zhang, R. Shi y F. Li, «An Event-driven Service-oriented Architecture for Internet of Things Service Execution,» *International Journal of Online Engineering (iJOE)*, vol. 11, mar. de 2015. DOI: 10.3991/ijoe.v11i2.3842.

[12] A. Rahmatulloh, F. Nugraha, R. Gunawan e I. Darmawan, «Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems,» en *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, 2022, págs. 01-06. DOI: 10.1109/ICADEIS56544.2022.10037390.

[13] C. Rodríguez-Domínguez, K. Benghazi, M. Noguera, J. L. Garrido, M. L. Rodríguez y T. Ruiz-López, «A Communication Model to Integrate the Request-Response and the Publish-Subscribe Paradigms into Ubiquitous Systems,» *Sensors*, vol. 12, n.º 6, págs. 7648-7668, 2012, ISSN: 1424-8220. DOI: 10.3390/s120607648.

[14] Secretaría de Planificación Sectorial, «Programa para un Sistema Integrado de Transporte Público Masivo para la GAM (SITGAM 2020 - 2035),» Ministerio de Obras Públicas y Transportes, inf. téc., 2020.