UrbanTracker: Sistema de geolocalización de flota de vehículos en tiempo real

Autor: Andres Felipe Suaza Bustos

Afiliación: Servicio Nacional de Aprendizaje (SENA), Regional Huila, CIES, Ficha 2899747

Contacto: afsuaza29@soy.sena.edu.co, asuazb13@gmail.com

Fecha: 2025

Resumen

UrbanTracker es un proyecto de geolocalización de vehículos en tiempo real diseñado con tecnologías modernas para el seguimiento de vehículos en entornos urbanos. El sistema tiene una aplicación móvil desarrollada en React Native y un servidor backend desarrollado en Java con Spring Boot, comunicados mediante el protocolo ligero MQTT para lograr actualizaciones instantáneas de ubicación. La arquitectura implementada fue siguiendo un enfoque a microservicios, es decir, un monolito separado por módulos aplicando un DDD simple.

Palabras clave por categoría

Categoría	Términos
Dominio	Rastreo en tiempo real; Geolocalización
Arquitectura	Arquitectura de microservicios; Monolito modular (DDD simple)
Tiempo real / Mensajería	MQTT; Broker Mosquitto
Front-end	React Native; React; Next.js; Mapbox
Back-end / Seguridad / Datos	Spring Boot; JWT; PostgreSQL; Docker Compose

Introducción

La geolocalización en tiempo real se ha convertido en una herramienta clave en diversos ámbitos, desde la gestión de flotas vehiculares hasta la seguridad ciudadana. Por ejemplo, existen sistemas de monitoreo de buses urbanos que manejan millones de datos GPS al día, requiriendo arquitecturas escalables para funcionar correctamente (Torres-Berru, Camacho-Macas, Solano-Cabrera & León-Pinzón, 2020). El seguimiento en vivo de vehículos también es crucial en la recuperación de autos robados; un estudio en Uganda diseñó un sistema de rastreo de vehículos robados con notificaciones en tiempo real a autoridades, demostrando el impacto positivo de estas tecnologías en la seguridad urbana (Gorret & Vydehi, 2025). Al mismo tiempo, la logística moderna demanda soluciones en la nube para monitorear entregas y optimizar rutas en tiempo real (Desarrollo de un sistema de seguimiento..., 2025). Estas aplicaciones evidencian que la computación en la nube y el Internet de las Cosas (IoT) pueden integrarse para crear plataformas de seguimiento altamente eficientes y confiables.

A pesar de los avances, implementar un sistema propio de rastreo en tiempo real no es tan fácil. Es necesario capturar datos de geolocalización desde dispositivos móviles o sensores GPS, transmitirlos de forma instantánea a un servidor, procesarlos y mostrarlos en una interfaz amigable. Todo esto debe lograrse manteniendo bajos tiempos de respuesta y garantizando la seguridad de los datos transmitidos. Además, aparecen consideraciones de privacidad cuando se rastrea la ubicación de personas en tiempo real – por ejemplo, durante la pandemia COVID-19 se desarrollaron aplicaciones de rastreo de contactos con enfoques especiales para proteger la privacidad de los usuarios (*REACT, 2021*).

Este artículo presenta UrbanTracker, un sistema de geolocalización en tiempo real que aborda los aspectos mencionados mediante tecnologías accesibles y modernas. Este proyecto está

compuesto de una aplicación móvil desarrollada en React Native (*framework multiplataforma*) y un backend construido en Java con Spring Boot, que se comunican mediante el protocolo MQTT para lograr actualizaciones instantáneas de la posición geográfica sin tener altos recursos de conexión. El backend sigue una arquitectura con un enfoque a microservicios (*Monolito modular*), favoreciendo la escalabilidad y mantenibilidad del sistema (*Torres-Berru, Camacho-Macas, Solano-Cabrera & León-Pinzón, 2020*).

Marco teórico

Un sistema de rastreo en tiempo real combina una aplicación o dispositivo que toma la ubicación (GPS) con un canal de comunicación que la envía de inmediato a un servidor para visualizarla en mapas, generar alertas o reportes. En la práctica ya existen soluciones probadas tanto en seguridad urbana—como el seguimiento antirrobo con GPS e IoT en Uganda, que notificó en tiempo real y mejoró la recuperación de vehículos (Gorret & Vydehi, 2025)—como en logística en la nube, donde se usan aplicaciones móviles y un backend para optimizar la distribución de mercancías en tiempo real (Desarrollo de un sistema de seguimiento…, 2025).

Cuando crecen la cantidad de dispositivos o la frecuencia de envío, la arquitectura es crucial: el caso Kbus en Ecuador, con millones de posiciones GPS diarias, migró de un monolito a microservicios para soportar la carga y mejorar tiempos de respuesta, evidenciando que dividir el sistema en servicios pequeños facilita escalar y mantener un sistema de este tipo (Torres-Berru et al., 2020). En términos simples, un monolito concentra todo en una sola aplicación del lado servidor, mientras que los microservicios separan funciones y se comunican por APIs o mensajería, esto agrega un grado mayor de complejidad operativa, pero permite crecer por partes y actualizar sin afectar el conjunto (Torres-Berru et al., 2020).

En el cliente móvil, optar por un framework multiplataforma acelera el desarrollo: comparativas muestran buen rendimiento tanto en Flutter como en React Native, este último suele elegirse cuando el equipo domina JavaScript por su flexibilidad y comunidad (Macías Vera, 2021). Además, integrar mapas en la aplicación es directo con librerías como `react-native-maps` o APIs de Google Maps. Hay experiencias documentadas de geolocalización y trazado de trayectorias en React Native que sirven como guía para mostrar rutas y posiciones en pantalla de forma clara (De Dios García, 2021).

Para el “tiempo real” se emplea con frecuencia MQTT, un protocolo de mensajería ligero basado en un patrón publicador/suscriptor con un broker: la app publica mensajes de ubicación en un tópico(topic) y el backend, suscrito a ese tópico, recibe las actualizaciones al instante. Este patrón pub-sub desacopla emisores y receptores y resulta eficiente para flujos continuos de eventos (Ortiz Samboni, 2022). Hay, además, proyectos en React Native que envían ubicación al backend por MQTT, lo que confirma su viabilidad técnica en móviles. Su baja sobrecarga y la posibilidad de ajustar la calidad de servicio ayudan cuando la conexión es inestable o el dispositivo tiene recursos limitados (Terrones Albarracín, 2022; Ortiz Samboni, 2022).

En seguridad, conviene una autenticación coherente entre web y móvil; existen diseños de autenticación unificada para React y React Native que incluyen registro, inicio de sesión y recuperación de credenciales y pueden adaptarse a las necesidades del proyecto (Ye, 2022). Para proteger datos sensibles de ubicación, se recomienda controlar el acceso y cifrar las comunicaciones usando protocolos que encripten la comunicación.

Más allá de lo técnico, es clave aplicar principios de privacidad cuando se manejan ubicaciones personales, aprendiendo de aplicaciones de rastreo de contactos con enfoque de privacidad reforzada (REACT, 2021).

Enfoque de desarrollo

Para la creación y escritura de este artículo, el proyecto no se encuentra 100% completado. Las pruebas unitarias no están todas implementadas y las pruebas de integración se encuentran en progreso aún.

Se utilizó Scrum al inicio, pero por restricciones de tiempo y cambios de agenda se dejó de aplicar de forma estricta. A partir de ese punto se continuó con un enfoque ágil más ligero y orientado más a entregables, manteniendo la priorización y el seguimiento informal de tareas.

Se tomó como referencia una arquitectura de microservicios, pero la estructura actual corresponde a un monolito modular en Spring Boot. Esta elección deja el camino abierto para migrar a microservicios en el futuro sin refactorizaciones profundas.

Como inició del desarrollo se empezó con la aplicación móvil del conductor con React Native, se habilitó la geolocalización y se hizo una configuración básica del broker Mosquitto. Se realizaron pruebas básicas MQTT. Por tiempos de entrega, la mensajería se pospuso para concentrarnos en el backend.

Se diseñó el diagrama de clases y se implementaron controladores REST, servicios y repositorios (JPA). Se integró Spring Security + JWT para autenticación/autorización y se definieron contratos para la integración futura con MQTT.

Se desarrollaron los clientes en React con Next.js para visualizar mapas con la última ubicación y recorridos. El dashboard consume el API REST del backend para la administración general de los datos. Para mapas se utilizó Mapbox por su fácil integración y APIs disponibles con muchas funcionalidades.

Con el backend y webs estables, se empezó a retomar el desarrollo de la mensajería MQTT. En la aplicación móvil se integró el cliente para publicar JSON de ubicación en tópicos definidos según la ruta a la cual se encuentra asociado el conductor. En el lado del backend se activó el suscriptor MQTT para procesar, validar, persistir los datos y servirlos a la interfaz web.

Las pruebas unitarias fueron implementadas parcialmente en algunos servicios hasta el momento. Por otro lado, con las pruebas de integración se han ejecutado escenarios de punta a punta (E2E) con datos simulados para verificación, pero no cubre todos los casos.

La base de datos utilizada durante el desarrollo fue PostgreSQL, organizando esquemas separados por cada módulo del backend, lo que facilita aislar dominios, controlar permisos a nivel de esquema. Para el entorno local se preparó Docker Compose con PostgreSQL y Mosquitto, estandarizando las variables de entorno.

Resultados

Es importante aclarar nuevamente que este proyecto aún no se encuentra 100% completado. Las pruebas unitarias no están todas implementadas y las pruebas de integración aún se encuentran en progreso. Aun así, hasta el momento se han logrado los siguientes resultados:

Aplicación móvil funcional: Se desarrolló una aplicación móvil con React Native, capaz de obtener la ubicación GPS del dispositivo y enviarla en tiempo real al servidor. La app cuenta con una interfaz sencilla con un indicador del estado de conexión y algunos datos más sobre el conductor y su trabajo realizado. Se está comprobando su correcto funcionamiento tanto en un dispositivo Android físico como en emuladores. La tasa de envío de datos es configurable. En las pruebas actuales se está utilizando un envío cada 5 segundos para balancear precisión y consumo de batería.

Backend en Spring Boot: Se está implementando el servidor backend en Spring Boot, el cual recibe los datos de ubicación, los almacena y los sirve a la interfaz web. El backend está demostrando ser estable bajo pruebas con datos simulados. Se está desarrollando un servicio de consulta que devuelve la última posición conocida de cada dispositivo y el historial de posiciones en un intervalo de tiempo dado. Este servicio alimenta la aplicación web y podría ser utilizado por futuras funcionalidades adicionales.

Comunicación en tiempo real eficiente: La integración vía MQTT está resultando exitosa. En las pruebas realizadas, desde que la aplicación móvil publica una nueva coordenada hasta que el backend la registra transcurren típicamente menos de 300 ms aproximadamente. La mayor parte del tiempo total hasta la visualización proviene de la actualización en la interfaz web. Este rendimiento está confirmando que el uso de un canal de mensajería asíncrono es adecuado para la tarea, evitando la sobrecarga que tendría un enfoque basado en polling.

Conclusiones

En conclusión, UrbanTracker demuestra que la combinación de React Native en el cliente móvil, Spring Boot en el servidor y MQTT como mensajería pub-sub es efectiva para seguimiento en tiempo real a baja escala, logrando latencias cercanas a 1s aproximadamente en pruebas y una comunicación fluida entre componentes. El uso de un framework multiplataforma reduce el esfuerzo y tiempo de desarrollo, mientras que el diseño inspirado en microservicios facilita la escalabilidad del sistema y posibles intercambios tecnológicos sin refactorizaciones masivas. Además, se integra seguridad básica con JWT, adecuando el acceso a datos sensibles, y el sistema muestra flexibilidad para distintos casos de uso.

Referencias

- De Dios García, E. E. (2021). *Desarrollo de software de geolocalización y personalización de trayectorias de Google Maps utilizando React Native*. Anuario de Investigación UM, 2(2), 9-22. Recuperado de anuarioinvestigacion.um.edu.mx/index.php/anuarioium/article/view/208.

- Gorret, A. E., & Vydehi, K. (2025). *Design and Implementation of a GPS-GSM based Real-Time Vehicle Theft Tracking System for Urban Security in Uganda*. International Journal of Innovative Science and Research Technology, 10(6). Recuperado de https://ijisrt.com/vehicle-theft-realtime-tracking-system-in-uganda.

- Macías Vera, E. V. (2021). *Estudio comparativo de los frameworks del desarrollo móvil nativo "Flutter" y "React Native"*. [Tesis de Licenciatura]. Repositorio Nacional CEDIA. Recuperado de https://redi.cedia.edu.ec/document/207158.

- Madrigal Chaves, W. (2020). *Buenas prácticas en programación: Seguridad*. Revista SENA, 5(1), 15-22.

- Ortiz Samboni, J. O. (2022). *Módulo de comunicación SmolComm*. [Trabajo de grado, Universidad de Antioquia]. Recuperado de https://hdl.handle.net/10495/31737.

- REACT: *Rastreo de contactos en tiempo real y monitoreo de riesgos mediante rastreo móvil con privacidad mejorada*. (2021). [Artículo de conferencia]. IEEE Xplore. Recuperado de https://ieeexplore.ieee.org/document/9458685.

- Terrones Albarracín, M. (2022). *Diseño y desarrollo de un app en React Native y back-end para gestión de presencia de trabajadores*. [Trabajo de Fin de Grado, Universidad Politécnica de Cartagena]. Recuperado de http://hdl.handle.net/10317/11662.

- Torres-Berru, Y., Camacho-Macas, J., Solano-Cabrera, J., & León-Pinzón, L. F. (2020). *Migración de un monolito a una arquitectura basada en microservicios, caso de estudio sistema "Kbus"*. Dominio de las Ciencias, 6(2), 763-781. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=7425696.

- Ye, X. P. (2022). *Diseño e implantación de un sistema de autenticación multiplataforma para React y React Native*. [Trabajo de Fin de Grado, Universidad Politécnica de Madrid]. Recuperado de https://oa.upm.es/71336/.

- *Desarrollo de un sistema de seguimiento de aplicaciones móviles basado en la nube para funciones de distribución logística de salida*. (2025). Journal of Logistics Technology, 15(3), 50-60. Recuperado de https://www.sciencedirect.com/science/article/pii/S2352146525004533.

- Torres-Berru, Y., Camacho-Macas, J., Solano-Cabrera, J., & León-Pinzón, L. F. (2020). Migración de un monolito a una arquitectura basada en microservicios: caso de estudio sistema Kbus. Dominio de las Ciencias, 6(2), 763–781. Disponible en: https://dominiodelasciencias.com/ojs/index.php/es/article/view/1193

- Gorret, A. E., & Vydehi, K. (2025). Design and implementation of a GPS-GSM based real-time vehicle theft tracking system for urban security in Uganda. International Journal of Innovative Science and Research Technology, 10(6). Disponible en: https://ijisrt.com/vehicle-theft-realtime-tracking-system-in-uganda

- Ortiz Samboni, J. O. (2022). Módulo de comunicación SmolComm. [Trabajo de grado, Universidad de Antioquia]. Disponible en: https://bibliotecadigital.udea.edu.co/server/api/core/bitstreams/b10cf405-9348-4ab9-9ff5-8cf210fd5495/content

- Terrones Albarracín, M. (2022). Diseño y desarrollo de una app en React Native y back-end para gestión de presencia de trabajadores. [Trabajo de Fin de Grado, Universidad Politécnica de Cartagena]. Disponible en: https://repositorio.upct.es/entities/publication/f9ea9a9e-0ac1-42a0-bf55-7139c6177362

- Ye, X. P. (2022). Diseño e implantación de un sistema de autenticación cross-platform para React y React Native. [Trabajo de Fin de Grado, Universidad Politécnica de Madrid]. Disponible en: https://oa.upm.es/71336/1/TFG_XIAO_PENG_YE.pdf