ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

APLICACION WEB PARA EL CONTROL DE DESVIACIONES DE RUTAS EN EL TRANSPORTE PÚBLICO MEDIANTE IOT EN LA CIUDAD DE CAÑAR

TRABAJO DE TITULACIÓN

Previo la obtención del Título de:

Magíster en Telecomunicaciones

Presentado por:

Juan Javier Vazquez Verdugo

GUAYAQUIL - ECUADOR

Año: 2025

DECLARACIÓN EXPRESA

Yo Juan Javier Vazquez Verdugo acuerdo y reconozco que: La titularidad de los derechos patrimoniales de autor del proyecto de graduación corresponderá al autor, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores. El o los estudiantes deberán procurar en cualquier caso de cesión de sus derechos patrimoniales incluir una cláusula en la cesión que proteja la vigencia de la licencia aquí concedida a la ESPOL.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, secreto empresarial, derechos patrimoniales de autor sobre software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique al autor que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 28 de Enero del 2025.

Firmado electrónicamente por:
JUAN JAVIER VAZQUEZ VERDUGO

Juan Javier Vazquez

EVALUADORES

Firmado electrónicamente por:
MARIA ANTONIETA
ALVAREZ VILLANUEVA

PhD. María Antonieta Álvarez
PROFESOR EVALUADOR

Firmado electrónicamente por:
EDUARDO LUIS
CHANCAY ROJAS

Mgtr. Eduardo Luis Chancay
PROFESOR EVALUADOR

RESUMEN

La solución propuesta en este proyecto es una mejora tecnológica en tiempo real mediante la implementación de herramientas de geolocalización y tecnologías IoT en la gestión del transporte público en Cañar.

Como nuestra importante propuesta es el control de las unidades de transporte y el contacto con los usuarios a través de una plataforma web. Esta solución es efectiva, sostenible y adaptable, y resuelve en gran medida los desafíos del transporte urbano en la región. El uso de sistemas de monitoreo que ofrecerá un aumento en la eficiencia operativa y mejoras en la experiencia del usuario.Para desarrollar del proyecto, comenzamos explorando una arquitectura de tres fases: adquisición de datos, backend y el frontend.

El uso de un módulo GPS conectado a la Raspberry Pi para la geolocalización, mientras el protocolo MQTT será necesario para él envió de datos y el uso de redes 4G. En el backend, se procesaron y almacenaron los datos JSON recibidos en una base de datos MySQL. Por último, implementamos Mapbox y OpenStreetMap en el frontend para la visualización interactiva de rutas, paradas y ubicaciones en tiempo real.

Los resultados fueron suficientes para demostrar que el sistema está funcionando adecuadamente, visualizando rutas y localizando con precisión las unidades de transporte en un portal web que por el momento el servicio se está ejecutando a nivel local ya que en el futuro cercano se pretende expandirlo a un servicio en la red para que los usuarios pueden acceder.

También funcionó bien en él envió de datos como el almacenamiento de estos datos mismos.

En respuesta a esto, se concluyó que la gestión y monitoreo del transporte público es posible gracias a la integración entre las tecnologías IoT y herramientas de código abierto; por lo tanto, se considera una solución viable y replicable para otros escenarios similares.

Palabras Clave: Transporte Publico, IoT, Geolocalización, MQTT.

ABSTRACT

The solution proposed in this project is a real-time technological improvement by implementing geolocation tools and IoT technologies in public transport management in Cañar, Ecuador.

The main objective is to improve the control of transport units and contact with users through a web platform. This solution is effective, sustainable and adaptable, and largely solves the challenges of urban transport in the region.

The use of real-time monitoring technologies will offer an increase in operational efficiency and improvements in user experience.

To develop the project, we began by exploring a three-phase architecture: data acquisition, backend and frontend.

The use of a GPS module connected to the Raspberry Pi for geolocation, while for real-time data transmission the MQTT protocol and the use of 4G networks were required. In the backend, the received JSON data was processed and stored in a MySQL database. Finally, we implemented Mapbox and OpenStreetMap in the frontend for interactive visualization of routes, stops and locations in real time.

The results were sufficient to demonstrate that the system is working properly, visualizing routes and accurately locating transport units on a web portal that users can access.

It also worked well for real-time transmission and data storage.

In response to this, it was concluded that public transport management and monitoring is possible thanks to the integration between IoT technologies and open-source tools; therefore, it is considered a viable and replicable solution for other similar scenarios.

Keywords: Public Transport, IoT, Geolocation, MQTT.

ÍNDICE GENERAL

ABREVIATURAS

MQTT Protocolo de Telemétrica de Cola de Mensajes

IoT Internet de las Cosas

OPS Organización Panamericana de la Salud

GPS Sistema de Posicionamiento Global

HTTP Protocolo de transferencia de Hipertexto

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

El transporte público en latino América enfrenta desafíos muy importantes y significativos, siendo estos la congestión, la impuntualidad, y la falta de gestión eficiente en las rutas, esto contribuye a una experiencia insatisfactoria por parte de los usuarios.[1] Específicamente en la ciudad de Cañar, Ecuador, estos problemas son más comunes por la limitada infraestructura enfocada en el control del tránsito, así también prácticas que no están alineadas con un uso responsable y adecuado de los medios de transporte público, principalmente los buses. Esto problemas afectan directamente las zonas de cobertura generando experiencias negativas como impuntualidad y retrasos por parte de los usuarios.

La implementación de tecnologías IoT en los transportes públicos permiten tener un control al monitorear y analizar datos en tiempo real, esto ayuda a una mejor gestión en las rutas de la ciudad. El uso de sensores GPS implementados en el transporte público como los Autobuses facilita el seguimiento continuo de su ubicación en tiempo real, esto permitiría con la detección de desviaciones y actividades que no están planificadas en el uso de estas unidades. Estudios demuestran que los sistemas IoT reducen tiempos en un 20-30% mejorando la operatividad con el manejo de estos recursos.[2]

En América Latina, más del 80% habitan en zonas urbanas, lo que hace más interesante y necesario la implementación de sistemas IoT que mejore la eficiencia atacando estos problemas de congestión y contaminación. Estudios resaltan que sistemas IoT en el transporte urbano contribuyen en la reducción de emisiones contaminantes hasta un 15-25% gracias al optimizar las rutas en una ciudad. Específicamente en ciudades como Cañar carece de personal que labore en la gestión de tránsito, esto limita a las autoridades gestionar de manera más eficiente las desviaciones y paradas no planificadas, esto resalta el poder implementar un control mediante el monitoreo para la misión proactiva en la desviación y control de rutas.[3]

1.1 Descripción del Problema

Las Ciudades modernas afrontan significativos desafíos con el consumo de energía, la contaminación, la mala gestión del tráfico y una gran necesidad de descarbonización. Esta problemática ya no solo suele afectar a la calidad de aire ya que también produce un gran impacto en la movilidad y en el bienestar social. A estos problemas surgen respuestas como, las soluciones tecnológicas que combinan IoT y el Big Data, estas alternativas están transformado a las conocidas “Smart Cities” o Ciudades Inteligentes.[3]

Los Autobuses en América Latina son los medios de transporte público más utilizados por la ciudadanía, pero existe una ineficiencia en su uso, lo que produce congestión, aumento en la contaminación, estos problemas no solo afectan en lo cotidiano a la ciudadanía ya que varios estudios demuestran que afectan negativamente a la salud de los ciudadanos. En varias ciudades de Ecuador, como Cañar estos problemas se agravan, los altos índices en contaminación que provoca este tráfico a diario. La OPS demuestra que en Ecuador fallecen en el año aproximadamente 500 personas gracias a la contaminación en el aire en sus ciudades.[4]

La problemática principal está centrada en solucionar una problemática tanto en lo estructural como social que sucede en nuestro cantón con el uso del de este servicio ya mencionado. Existen indicios de que el servicio, en algunos casos, no podría estar siendo utilizado de la manera más adecuada, lo que incluye paradas no autorizadas para actividades privadas, ingresos al medio de transporte fuera de las rutas o paradas no asignadas a este. Este mal uso provoca alteraciones en los horarios programados en el desarrollo habitual de este servicio, afectando principalmente la experiencia del usuario. También se ha reportado a las autoridades de la Ciudad el uso de estas unidades para el transporte de mercancías y desplazamientos no planificados, lo que genera un impacto a la cobertura de las rutas planificadas, esto provocando la falta de servicio en ciertos sectores.
Además, que enfrenta la Ciudad a esta problemática es la falta de personal dedicado al control y gestión del tránsito. Esta falta de recurso el control y

ordenamiento del espacio público dificulta el orden en el espacio público permitiendo este tipo de irregularidades y no sean sancionadas. Es por ello por lo que juntamente con las autoridades de la ciudad se ha concluido implementar un sistema de monitoreo continuo que permita controlar en tiempo real las rutas y actividades de los autobuses que permita abordar toda esta problemática y brindar una solución eficiente para los usuarios de este servicio.

Para enfrentar estos desafíos, la adopción de un sistema de IoT destinado al monitoreo en tiempo real y al control de las alteraciones en las rutas de transporte público podría resultar ser una solución eficaz. Este sistema facilitaría la supervisión continua de los autobuses, permitiendo la optimización de las rutas, la disminución de la congestión y la reducción de la contaminación ambiental, lo que a su vez mejoraría la eficiencia en el servicio del transporte urbano como rural de la ciudad. Un sistema de IoT que realice un seguimiento en tiempo real y gestione estas desviaciones garantiza una respuesta rápida ante situaciones inesperadas, ajustando las rutas adecuadamente para mitigar el impacto negativo. Además, al optimizar las trayectorias, se pueden acortar los tiempos de viaje, aumentar la fiabilidad del servicio y elevar la satisfacción del usuario. Estos beneficios contribuyen a una movilidad urbana más sostenible, hacen el servicio de transporte sea más eficaz para los ciudadanos y ayudan a reducir la huella de carbono en la ciudad.[5]

1.2 Justificación del Problema

La adopción de tecnologías IoT en el transporte público se respalda por su potencial para aumentar la eficiencia del tráfico y disminuir la contaminación. En localidades como Cañar, la escasez de personal y recursos para la gestión del tránsito complica el control de desvíos no autorizados y actividades no previstas en los autobuses. Investigaciones indican que la implementación de sistemas IoT puede disminuir entre un 20% y un 30% reduciendo esta brecha en los tiempos de espera, además de recortar las emisiones de CO2 hasta en un 25%, lo que favorece una operación más sostenible y mejora la experiencia del usuario.[2]

El modelo de ciudades inteligentes, respaldado por la gestión de datos en tiempo real, ha evidenciado beneficios notables en América Latina, logrando avances en la movilidad urbana y disminuyendo la contaminación mediante la incorporación de soluciones de IoT en la infraestructura del transporte público. La supervisión constante de las rutas y la posibilidad de modificarlas en tiempo real favorecen una respuesta más eficaz ante situaciones inesperadas, lo que no solo mejora la confiabilidad del servicio, sino que también reduce la congestión en las vías.[5]

Este proyecto tiene como objetivo simular un sistema innovador de IoT para incrementar la productividad de los medios de transporte en este caso los buses en Cañar. Este sistema estará formado por una red de sensores GPS distribuidos en las diferentes unidades de transporte público. Estos sensores recogerán datos en tiempo real sobre la ubicación GPS, que se enviarán en forma de coordenadas de latitud y longitud a una plataforma IoT, donde se mostrarán a través de una aplicación web para el monitoreo del transporte público.

La aplicación permitirá la visualización en tiempo real de diversas métricas, como la ubicación de los autobuses, el historial de trayectos y alertas sobre desviaciones y paradas no autorizadas, informando automáticamente a los operadores y usuarios sobre cambios inesperados. Los datos serán accesibles desde un servidor local, el cual se utilizará para recolectar la base de datos y supervisar el recorrido de las unidades de transporte. Esto conducirá a una mejora en el manejo y control operativo optimizando y disminuyendo en la cantidad de tiempo para que los pasajeros utilicen el servicio.

En conclusión, la propuesta no solo pretende modernizar el monitoreo del transporte público mediante IoT, sino también ofrecer una herramienta integral para la gestión proactiva de desviaciones de rutas, mejorando así la experiencia en el ciudadano y la eficiencia del sistema de transporte.

1.3 Objetivos

1.3.1 Objetivo General

1 Simular un sistema para el monitoreo de rutas en tiempo real, mediante tecnologías IoT, para el control de desviaciones en la utilización del transporte público en la ciudad de Cañar.

1.3.2 Objetivos Específicos

1. Investigar, un sistema para la identificación de coordenadas en tiempon real implementado en el transporte publico
2. Simular un sistema de identificación de coordenadas en tiempo real, que permita la transmisión de los datos GPS a un servidor local.
3. Diseñar una aplicación WEB donde podamos visualizar en tiempo real las coordenadas GPS en longitud y amplitud de las unidades de transporte público, e integrar un sistema de alarmas que notifique cuando estas de desvíen de las rutas establecidas.
4. Elaborar una base de datos para la recepción y almacenamiento de datos transmitidos.

1.4 Propuesta de la Solución.

Esta propuesta se basa en un prototipo que permita el monitoreo continuo de un sistema basado en IoT en la mejorar y operación del transporte público de la ciudad de Cañar:

Se utilizarán varios sensores GPS en las unidades individuales de los medios de transporte público, recogiendo datos de ellas y proporcionando las coordenadas geográficas de los buses correspondientes. Luego, los datos se transmitirán automáticamente a un servidor local donde se almacenarán, procesarán y visualizarán a través de una aplicación web personalizada.

- Seguimiento de Ruta en Tiempo Real: Los sensores GPS rastrearán las unidades de transporte en tiempo real siguiendo su ubicación y detectando

los desvíos fuera de ruta. Esto permitirá una mejor gestión de rutas y minimizará las desviaciones operativas.

- Sistema de Alertas: Un sistema de alertas monitoreará las desviaciones de la ruta planificada, paradas no autorizadas y otras actividades anormales. El siguiente paso, explica, es elevar estas alertas a través del control de tráfico a los operadores de monitoreo, lo cual será crucial para la etapa de respuesta.
- Visualización para Aplicación Web: Una aplicación web interactiva para que los operadores/autoridades monitoreen la ubicación y trayectorias de las unidades de transporte público y observen las alertas en tiempo real. También incluirá herramientas para analizar datos históricos.
- Para el procesamiento del conjunto de datos del medio de transporte se elaborará una base de datos, para su posterior flujo entre sistemas y para un uso eficiente de toda la información recopilada sobre rutas, desvíos y eventos.

1.5 Marco Teórico.

1.5.1 IoT Transporte Publico

El transporte público se define como un sistema diseñado para movilizar a grandes grupos de personas, que busca satisfacer las necesidades de desplazamiento de los ciudadanos. Su objetivo principal es facilitar un transporte eficiente, ligero, agradable y seguro hacia los lugares donde los habitantes realizan sus actividades.[6]

Las ciudades inteligentes pueden describirse como aquellas que optimizan el uso de servicios e infraestructura tecnológica, integrándolos en diversos sectores como de productividad y tecnología, con el fin de abordar tanto las necesidades sociales como económicas. Las tecnologías inalámbricas han proporcionado soluciones efectivas para el aumento de dispositivos cotidianos conectados a la web, así también el crecimiento en producción de dispositivos que permiten el seguimiento y evaluación de los servicios urbanos como los autobuses. En este

contexto, el Internet de las Cosas (IoT) está desarrollando soluciones confiables y ganando popularidad mediante la implementación de aplicaciones en áreas como el transporte, la salud y el ecosistema.[6]

Varios estudios caracterizan al IoT como “una red abierta y completa de objetos inteligentes que pueden intercambiar información, datos y recursos, respondiendo a cambios en el entorno.” El IoT abarca dispositivos cotidianos que están interconectados a gran escala, formando un sistema global de redes informáticas que operan utilizando protocolos TCP/IP.[7]

1.5.2 Servidor WEB

Un servidor web actúa como una plataforma que facilita el acceso a información para un cliente mediante los protocolos HTTP o HTTPS. En el contexto de las aplicaciones IoT, el servidor web se convierte en el elemento central para la recolección y distribución de datos provenientes de los dispositivos conectados. Las solicitudes HTTP más comunes incluyen GET, POST, PUT y DELETE, las cuales se utilizan para acceder a, enviar o modificar la información que esta alojada dentro del sistema.[6]

En un contexto más técnico, el servidor WEB emplea protocolos seguros y eficaces para transmitir información de forma rápida y confiable. Esto resulta fundamental en aplicaciones que requieren una alta frecuencia de actualización, como el almacenamiento y la actualización de datos geoespaciales (latitud y longitud) necesarios para el seguimiento del transporte público.[7]

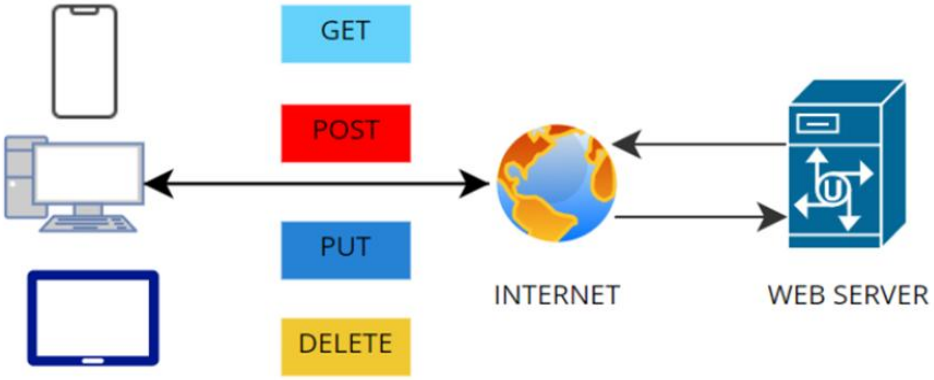


Figura 1. 1Servidor WEB. (Fuente: Autor)

Un servidor web destinado a aplicaciones de monitoreo puede configurarse como un servidor local, lo que implica que se ubica físicamente dentro de la red, en lugar de estar hospedado en la nube. Este servidor local gestiona igualmente las solicitudes HTTP para la recopilación, almacenamiento y distribución de información. Los beneficios de optar por el servidor local se encuentran en su notable disminución en la latencia, lo que optimiza la capacidad de respuesta de la aplicación.[7]

1.5.3 Protocolos MQTT

El protocolo MQTT es un protocolo ligero que facilita la transmisión de mensajes en grupos. Este sistema opera mediante un modelo de publicación/suscripción relacionada con los temas pertinentes.

la Figura 1.1 presenta una ilustración de publicación/suscripción. El publicador elige un el tema y es responsable de enviar mensajes a temas específicos. Por su parte, los suscriptores se encargan de inscribirse en los diversos temas de su interés y reciben mensajes cuando se publican en los nuevos temas a los que están suscritos. La comunicación entre el publicador y el suscriptor se realiza a través de un bróker MQTT, que se encarga de distribuir los mensajes a todos los suscriptores asociados a un tema particular.[8]

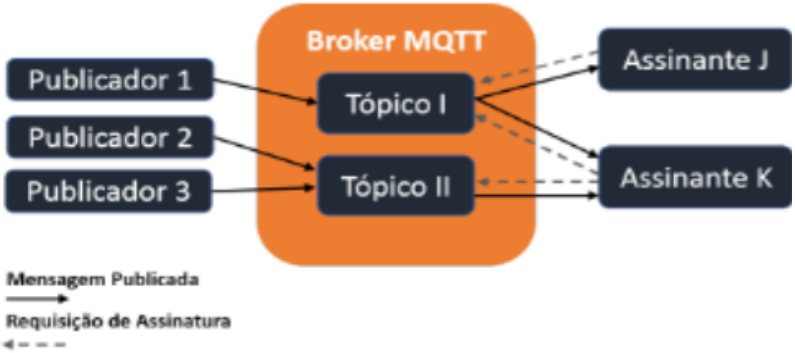


Figura 1. 2 Modelo Publicador/Suscriptor utilizado en MQTT. [8]

La Figura 1.2 representa cómo tres dispositivos, denominados Publicadores, se conectan a un bróker que alberga dos temas. También se puede notar la presencia de dos suscriptores; el suscriptor J muestra interés en los mensajes del Tema 1, que incluye la información del Publicador 1. Por su parte, el suscriptor K recibe datos del Publicador 1, ya que está suscrito al tema 1, y también accederá a la información de los publicadores 2 y 3 debido a su suscripción al tema 2.[8]

Una de las ventajas de este cliente MQTT es su versatilidad, ya que puede implementarse en dispositivos que van desde computadoras hasta microcontroladores. MQTT es particularmente adecuado para aplicaciones vinculadas al Internet de las Cosas, con fines como la monitorización y análisis en tiempo real, así como para mantenimiento, entre otros. Existen aplicaciones que se ejecutan como código abierto y se encuentran elaboradas para recopilar, guardar y examinar informaciones provenientes de sistemas de recopilación de datos.[9]

1.5.4 Brokers MQTT

Existen múltiples brokers disponibles para proyectos enfocados en el IoT, siendo algunos de ellos de código abierto y otros de pago. La siguiente Tabla 1.1 muestra las características y beneficios de estos brokers.[9]

Tabla 1. 1 Características Brokers MQTT de Código abierto.

Broker MQTT	Características y Ventajas
Mosquito	Escrito en C, “Mosquito” es el bróker más utilizado para proyectos de IoT, destacándose por ser ligero y escalable. Soporta los protocolos “MQTT” y “MQTT 5”.[8]
EMQ X	Desarrollado en “Erlang”, “EMQ X” es el bróker más escalable, capaz de manejar la mensajería para más de 10 millones de clientes en un único clúster.[8]
Cassandana	“Cassandana”, una solución basada en **Java** para MQTT, incluye todas las características que el protocolo admite.[8]
Ejjaberd	Los servicios de intermediación “MQTT 5.0”, escritos en “Erlang”, funcionan sobre un servidor **XMPP** y pueden gestionar millones de conexiones simultáneas.[8]

1.5.5 Bases de Datos

Lo siguiente constituyen un sistema fundamental para la gestión y stock de información o datos relevantes en organizaciones o empresas. Su diseño permite una organización lógica y eficiente, facilitando el acceso y el análisis de datos críticos, fundamentales para una correcta toma de acciones en los diferentes niveles, desde el operativo hasta el estratégico. La evolución de estas tecnologías ha incrementado significativamente la productividad de las bases de datos,

estableciéndolas como un componente central tanto en los procesos organizacionales como en el desarrollo de aplicaciones que demandan alta fiabilidad y seguridad de la información.

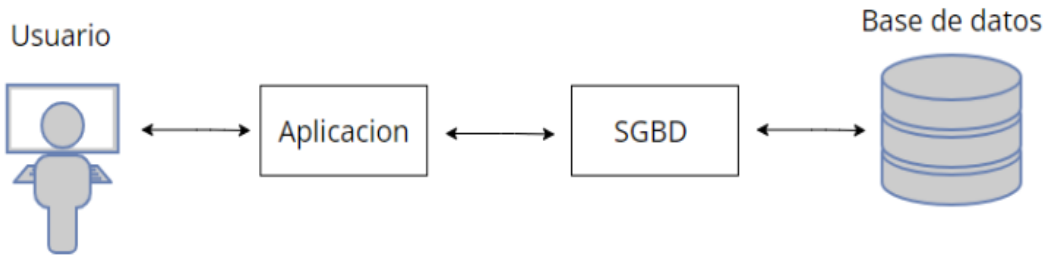


Figura 1. 3 Esquema del funcionamiento de una Base de Datos. (Fuente: Autor)

1.5.6 Sistema seleccionado como Gestión de Base de Datos (XAMMP)

XAMPP es un sistema de software libre que integra varios componentes esenciales para el desarrollo, incluyendo Apache, MySQL o MariaDB, PHP y Perl. Esta plataforma facilita la configuración de un servidor local para el desarrollo web. Su naturaleza de código abierto lo hace accesible y proporciona características valiosas para la implementación de proyectos de monitoreo y rápida ejecución.

Dentro de XAMPP, MySQL se destaca como una plataforma que permite la relación entre datos presentándose como un abanico muy grande en respuesta a proyectos de IoT gracias al manejo de grandes densidades de datos como también su fiabilidad. Al ser de software libre, MySQL proporciona una estructura fundamental para proyectos de IoT, permitiendo la gestión de las solicitudes HTTP de diversos dispositivos IoT y el almacenamiento de datos geográficos. Esto garantiza seguridad, organización y acceso de la información recopilada.[9]

1.5.7 Geo-Posicionamiento.

El comprender a que hace referencia el título de esta sección, es fundamental analizar el significado semántico de las palabras "Geo" y "Posicionamiento". El término “Posicionamiento” se refiere a la identificación del espacio de distintos objetos, ya sean móviles o fijos, lo cual puede hacerse mediante sistemas de coordenadas o en relación con un punto específico que se toma como referencia

dentro de un sistema de coordenadas. En el idioma español, “Geo” se traduce como “Tierra”.

A partir de estas definiciones, ...el Geo-Posicionamiento es técnica o procedimiento que utilizamos para identificar un determinado objeto en coordenadas como longitud y latitud en la Tierra. En el desarrollo de este prototipo, nos enfocaremos principalmente ya mencionado “Sistema de Posicionamiento Global” (GPS), que permitirá obtener las coordenadas de un objeto en territorio terrestre y seguir su trayectoria. Para entender su funcionamiento, es importante saber que el GPS funciona con un sistema completo de satélites enlazados entre sí, exactamente 24 satélites cubren alrededor del planeta con trayectorias sincronizadas, con el objetivo de identificar la ubicación de un dispositivo que recepte la señal en la Tierra. El dispositivo que recepte la señal se encuentra en determinado punto en la Tierra, y entre el dispositivo receptor y el satélite existe una distancia radial.[10]

Al recopilar información de dos satélites, podemos establecer una circunferencia en el punto donde sus áreas de cobertura se intersectan. Además, al consolidar una ubicación por parte del tercer satélite, se soluciona un problema de la desincronización entre los elementos que permiten la sincronización de estos GPS. Con determinados tiempos en los satélites, es posible ubicar con precisión las coordenadas en longitud y latitud del objeto que teníamos como objetivo su ubicación.[11]

CAPÍTULO 2

2. METODOLOGIA

2.1 Arquitectura

En este proyecto se presenta la arquitectura que se ilustra a continuación, la cual está compuesta por tres etapas: Sistema de adquisición de datos, Back END y Front END.

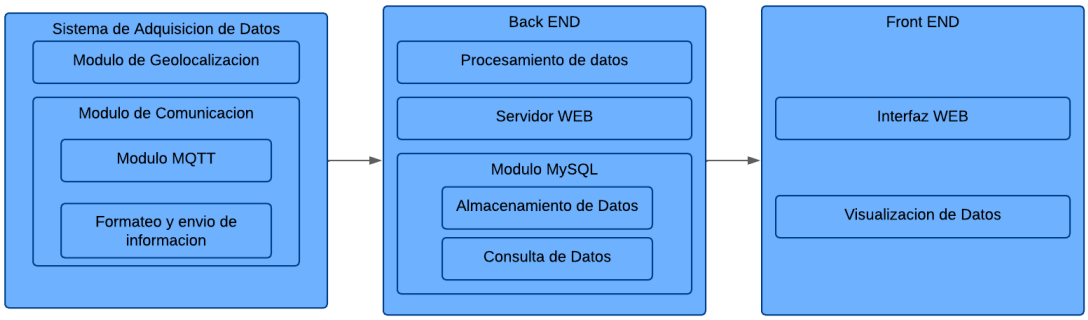


Figura 2. 1 Arquitectura de los distintos módulos para la ejecución del Proyecto. (Fuente: Autor)

- Como primer elemento de nuestro sistema lo denominaremos el Sistema para la Adquisición de Datos: Este estará encargado de recolectar la información necesaria para poder ser enviada a nuestro servidor local.
- Back END: Se encarga en el funcionamiento de la lógica del sistema para la ejecución del programa.
- Front END: Se encarga de la comunicación con el Back END como también con el diseño y la interfaz de la página WEB.

2.2 Sistema de Adquisición de Datos.

Este sistema tendrá como objetivo la adquisición de coordenadas en tiempo real luego enviárselas a un servidor local

2.2.1 Módulo de Geolocalización.

Para lograr la geolocalización en tiempo real del transporte público, es necesario recopilar datos de latitud y longitud mediante un módulo GPS que estará vinculado a la Raspberry Pi para obtener las coordenadas. Para este propósito, desarrollamos un archivo en Python este a su vez proporciona datos que contienen la latitud, longitud y la hora que se obtienen esos datos correspondientes a la ubicación en tiempo real, como se muestra en la imagen siguiente.

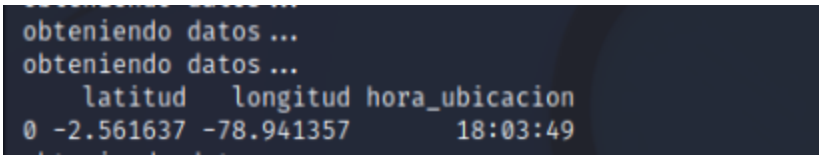


Figura 2. 2 Datos recibidos del módulo de Geolocalización. (Fuente: Autor)

En la siguiente Figura 2.3 podemos observar el proceso que sigue el módulo de Geolocalización en el proceso de adquisición de datos.

Figura 2. 3 Proceso del dispositivo GPS en el sistema. (Fuente: Autor)

2.2.2 Módulo para la comunicación

Para facilitar la comunicación, se emplea el protocolo MQTT para el envío de datos. La conectividad se logrará a través de una red 4G, utilizando un módem conectado a la Raspberry Pi.

2.2.2.1 Modulo MQTT

En el desarrollo de esta sección, empleamos una biblioteca llamada “paho-mqtt”, está basada en el lenguaje de programación Python. Esta biblioteca facilita la comunicación con un bróker MQTT, donde se podrán publicar mensajes y suscribirse a tópicos para acceder a los mensajes publicados. El bróker MQTT utilizado en este proyecto es “bróker.emqx.io”. La Raspberry Pi instalada en el transporte público será responsable de la publicación, mientras que el usuario actuará como suscriptor a través del servidor.

2.2.2.2 Envió de información

La información recopilada de la longitud y latitud están almacenadas en un archivo con un formato llamado “JSON”, la siguiente Figura 2.4 enseña la ejecución detallada de cómo es la divulgación de estos datos. Como se puede observar se detalla la información en: Latitud, Longitud, hora para poder ser publicada por el protocolo MQTT.

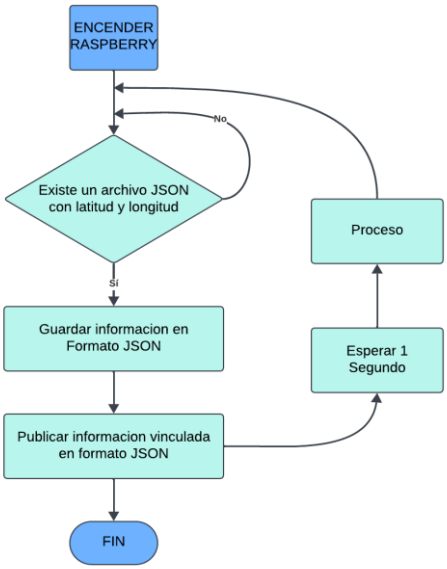


Figura 2. 4 Envió de información en formato JSON. (Fuente: Autor)

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3.9 pub_usuarios.py
{'"{\\"Unnamed: 0\\":{\\"0\\":0},\\"Unnamed: 0.1\\":{\\"0\\":0},\\"latitud\\":{\\"0\\":-2.1166833333},
longitud\\":{\\"0\\":-85.1079716667},\\"hora_ubicacion\\":{\\"0\\":\\"17:09:32\\"},\\"Unnamed: 0.2\\":
"0\\":0},\\"hora\\":{\\"0\\":\\"17:10:50\\"},\\"Unnamed: 0.3\\":{\\"0\\":0},\\"usuarios\\":{\\"0\\":2}
```

Figura 2. 5 Información en Formato JSON en la adquisición de datos. (Fuente: Autor)

El tema para publicar en el bróker MQTT se llamará “Python/bus_canar/monitoreo” donde podemos observar en la siguiente Figura como será publicada luego del tratamiento de los datos al extraer la información que necesitamos.

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3.9 informacion.py
   Unnamed: 0   latitud   longitud hora_ubicacion  Unnamed: 0      hora
0           0 -2.567315 -78.937745       12:59:26           0  16:50:31
```

Figura 2. 6 Información Filtrada. (Fuente: Autor)

2.2.3 Construcción del Módulo.

Nuestro proyecto incluye tres dispositivos que se conectarán a las Raspberry Pi: un módem 4G y un módulo GPS, como se ilustra en la imagen siguiente:

Figura 2. 7 Construcción del Sistema. (Fuente: Autor)

El módulo GPS tiene la función de proporcionar los datos geoespaciales del dispositivo instalado en la unidad de transporte así obteniendo su ubicación en ese preciso momento. Por otro lado, el módem 4G permite que las Raspberry Pi se conecten a la red 4G con el propósito de enviar la información recopilada.

2.3 Back END

El Back END es responsable de la lógica y el funcionamiento interno de nuestra aplicación. Las operaciones lógicas que realizamos incluyen: consultas a la base de datos, implementación del cliente MQTT y configuración del servidor WEB.

Tanto el Back END como el Front END requieren módulos para su correcto funcionamiento. En cada etapa de nuestro proyecto, se detallarán los módulos

necesarios para su desarrollo. Para instalar estos módulos, empleamos la herramienta “npm”, que actúa como un gestor de paquetes para Node.js. Al iniciar npm, se crea automáticamente un archivo “package.json” que compila la información sobre los paquetes requeridos.

2.3.1 Modulo MQTT

Obtener los datos necesarios en nuestra aplicación será necesario, ejecutar un módulo "mqtt". Esos datos se reciben en un formato conocido como “JSON”, el cual será procesado posteriormente.

En nuestro servidor local, tras instalar el paquete mqtt, podremos ejecutar el cliente MQTT y suscribirnos al tópico llamado "Python/bus_canar/monitoreo", lo que nos permite recibir la información que publica la Raspberry. La imagen siguiente ilustra la recepción de los mensajes en formato JSON, los cuales serán procesados más adelante para extraer la información requerida.

2.3.2 Procesamiento de Datos

Los datos recibidos en formato JSON a través del protocolo MQTT requieren procesamiento previo antes de ser guardados en nuestra base de datos, lo que permite que sea una búsqueda de manera intuitiva por parte del cliente.

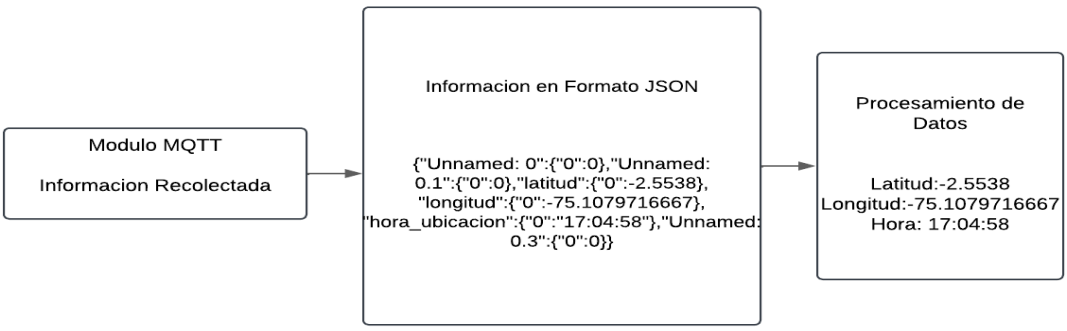


Figura 2. 8 Proceso de Filtrado de información en formato JSON. (Fuente: Autor)

2.3.3 Servidor WEB

El contenido obtenido mediante solicitudes HTTP de los usuarios es lo que permite que la página web se visualice en su navegador. Es relevante mencionar que, en este momento, nuestra aplicación opera en un entorno local debido a que se encuentra en la fase de pruebas. El Front End representa la información que se presenta al usuario, mientras que el Back End interactúa con el Front End a través de solicitudes HTTP. Para iniciar el servidor del Back End, utilizamos Express, un marco de trabajo que simplifica la creación de aplicaciones web del lado del servidor. Express, desarrollado en JavaScript, es la herramienta que empleamos para configurar nuestro servidor.

2.3.4 Modulo MySQL

Es un módulo de Node.js que facilita la conexión entre la computadora personal y la base de datos instalada en nuestro servidor local, permitiendo ejecutar consultas SQL desde nuestra aplicación.

2.3.5 Almacenamiento

Utilizando “MySQL/MariaDB”, donde la información será guardada en la base de datos denominada “transporte_urbano”. Dentro de esta base, existe una tabla llamada “datos_buses” que contiene las columnas mostradas en la imagen siguiente. Para establecer la conexión entre nuestra base de datos y la aplicación, es necesario ingresar las credenciales requeridas para realizar consultas sobre el esquema previamente configurado. Las credenciales son: Usuario: root@localhost, contraseña: “123456789”, base de datos: “transporte_urbano”.
Los datos se almacenan mediante la consulta SQL “INSERT INTO”, y esta operación se llevará a cabo cuando la Raspberry Pi comience a publicar la información recolectada.

id	unidad_bus	latitud	longitud	parada_siguiente	proxima_parada	parada_no_intervenida	timestamp Fecha y hora
158	Bus 01	-2.568634	-78.934616	Parada 1	Parada 2	Parada 3	2024-11-13 12:21:54
159	Bus 01	-2.567381	-78.934921	Parada 1	Parada 2	Parada 3	2024-11-13 15:58:57
160	Bus 01	-2.568696	-78.938568	Parada 2	Parada 3	Parada 4	2024-11-15 17:15:07
161	Bus 01	-2.565599	-78.938583	Parada 2	Parada 3	Parada 4	2024-11-15 17:15:24
162	Bus 01	-2.565366	-78.933685	Parada 2	Parada 3	Parada 4	2024-11-15 17:15:37
163	Bus 01	-2.565636	-78.933998	Parada 2	Parada 3	Parada 4	2024-11-15 17:15:49

Figura 2. 9 Almacenamiento de la información en las tablas correspondientes a la base de datos. (Fuente: Autor)

2.4 Front END

En esta parte de nuestro sistema, los usuarios tendrán oportunidad al interactuar con el backend. Para ello, hemos instalado los paquetes necesarios que permiten la visualización de datos y mejoran la estética de la aplicación.

- **Mapbox:** Se trata de un software libre que ayude con la ejecución de mapas interactivos con un alto grado de personalización. Esta herramienta utiliza datos provenientes de OpenStreetMap, una base de datos colaborativa y gratuita, para ofrecer mapas que son detallados y están constantemente actualizados. Con Mapbox, los usuarios pueden visualizar rutas, añadir marcadores y modificar la apariencia de los mapas mediante estilos avanzados. Es también compatible con dispositivos móviles y proporciona funcionalidades para el seguimiento en tiempo real, esto habilita a ser una opción sólida para aplicaciones de transporte público.
- **OpenStreetMap (OSM):** Es un proyecto colaborativo a nivel mundial que ofrece mapas abiertos y gratuitos que pueden ser editados por cualquier usuario. Este recurso sirve como la base de datos para plataformas como Mapbox, permitiendo así la entrega de mapas de alta calidad y actualizados. Su enfoque abierto fomenta que desarrolladores y organizaciones puedan utilizar y contribuir al enriquecimiento de los datos geoespaciales.

2.4.1 Interfaz WEB

Una interfaz de sistema de monitoreo de rutas se creó con HTML y CSS para la ejecución y visualización de la misma. Esta se divide en varias secciones amplias, cada una destinada a cumplir un papel particular en la plataforma.

La primera parte es un mapa interactivo que representa visualmente las rutas de autobuses y los lugares donde se encuentran en el momento de la visualización, lo que permite monitorear dinámicamente las unidades de transporte.
La segunda sección consiste en tablas de información indispensable en la ejecución de la aplicación, estas tratan sobre datos de los autobuses, su ID de unidad, ubicación, y otros datos que ayudan a comprender su estado breve en el sistema en su representación simple.

La tercera sección proporciona acceso a las ubicaciones históricas de los autobuses, una herramienta valiosa para investigar trayectorias pasadas e identificar rutas posibles o el personal que se desvía de ellas.

Y luego está la sección de entrada de paradas para nuevas paradas, en la que se pueden actualizar y procesar las paradas dentro del sistema, permitiendo que todas las rutas se mantengan actualizadas con métodos adecuados de transporte.

Figura 2. 10 Esquema de la interfaz de la aplicación WEB. (Fuente: Autor)

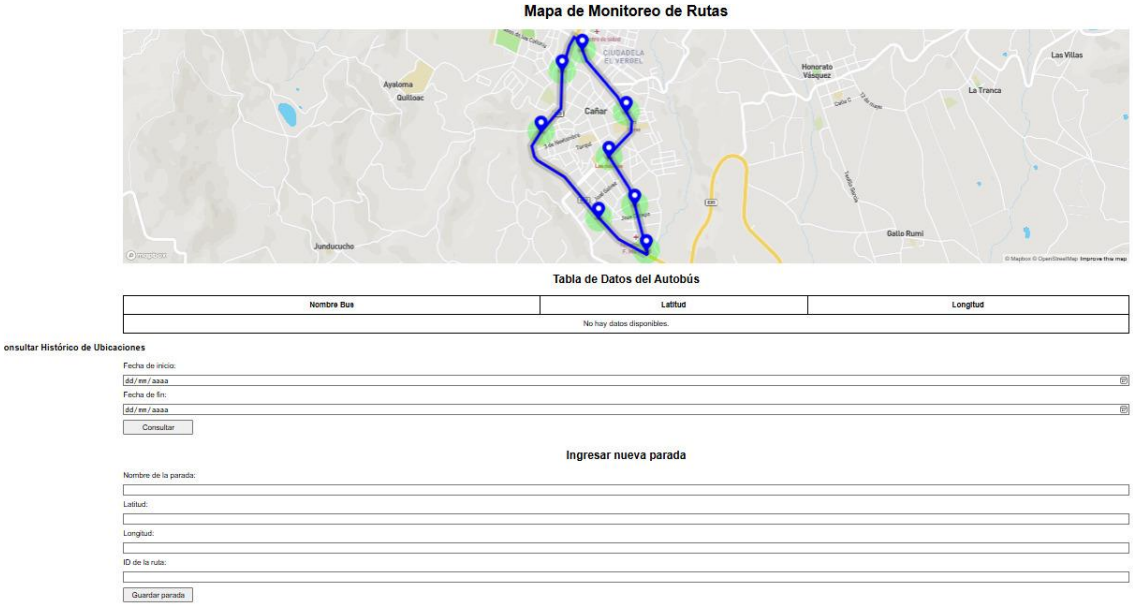


Figura 2. 11 Interfaz WEB. (Fuente: Autor)

El sistema de monitoreo de rutas, construido utilizando HTML y CSS, presenta los datos históricos en un formato tabular para ayudar en la visualización y comprensión de las posiciones históricas de los autobuses.

El encabezado en la parte superior contiene un formulario con los filtros de búsqueda para seleccionar el rango de fechas actual y la unidad de autobús para la consulta. Permite al usuario definir el acceso a datos relevantes de una manera que se adapte a los requerimientos. En la primera sección de la página se encuentra el mapa interactivo, que muestra las ubicaciones históricas del autobús

seleccionado para el rango de fechas ingresado. Muestra un mapa geográfico claro de las rutas tomadas, señalando los caminos más comunes recorridos en el pasado.

Y, por último, al final de la página, también tenemos una tabla con los datos históricos, que incluye información detallada, como el ID del autobús, la latitud y longitud de los lugares registrados y la fecha exacta de cada posición. Esta parte proporciona una representación gráfica del mapa y los detalles precisos en una vista tabular para que uno pueda tener un enfoque completo y organizado de los datos históricos.

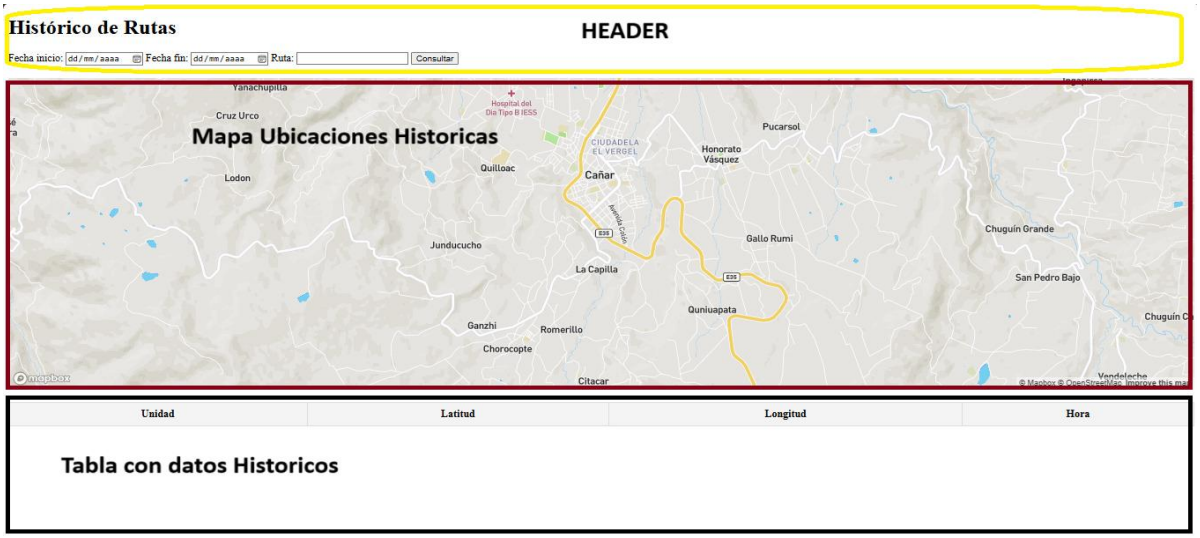


Figura 2. 12 Pestaña de Datos Históricos. (Fuente: Autor)

CAPÍTULO 3

3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Aquí se presentará el análisis y sus resultados del sistema simulado y a su vez con algunas pruebas en los buses del transporte público de la ciudad de Cañar, adicionalmente se adjunta un desglose de los costos asociados a todos los elementos requeridos para el desarrollo del proyecto. También se analizarán los resultados obtenidos de esta implementación.

3.1 Implementación

Se llevó a cabo la instalación de nuestro dispositivo en un bus del transporte público de Cañar, específicamente en el autobús que opera en la ruta 1. La instalación del dispositivo se realizó en un tiempo estimado de 2 horas, que coincide con la duración de las baterías integradas en la Raspberry. Es importante señalar que se están utilizando estas baterías debido a que el sistema aún se encuentra en fases de prueba y ajuste. En etapas posteriores, el dispositivo será conectado a las baterías del vehículo para asegurar un funcionamiento continuo del sistema.

Figura 3. 1 Instalación de sistema de detección de Coordenadas. (Fuente: Autor)

Como se puede notar, los elementos fue instalado en los asientos donde viaja pasajero, fijado a los asientos, como se muestra en la Figura 3.1. A continuación, se presenta la distribución de los componentes que integran el sistema, tal como se describió en el capítulo anterior.

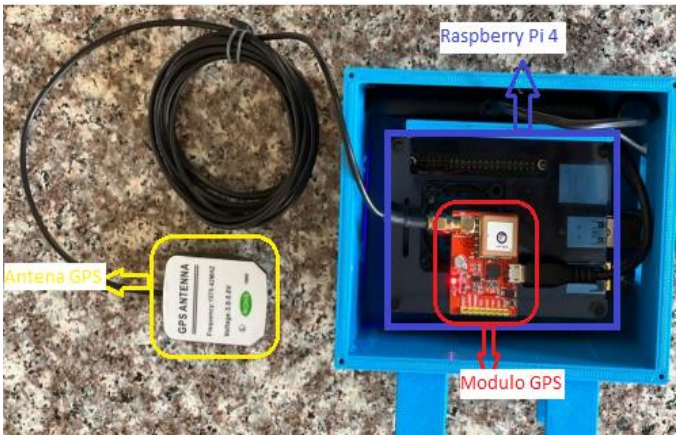


Figura 3. 2 Distribución de los componentes del Sistema. (Fuente: Autor)

En las siguientes Figuras 3.3 y Figura 3.4, se muestra cómo funciona la página WEB, tenemos 4 apartados, principalmente tenemos la sección del “Mapa de Monitoreo de Rutas” en el cual se observará la ubicación en tiempo real, esta será utilizada para monitorear las unidades de transporte en el recorrido que están designadas.

La sección de “Tabla de Datos del Autobús”, esta contiene una tabla que donde se mostraran detalles del Bus que se encuentra monitoreado como: Nombre del Bus, latitud y su longitud, esto con la finalidad de brindar más detalle de la ubicación en tiempo real del bus que se encuentra monitoreado.

A continuación, sigue el apartado de “Consultar Histórico de Ubicaciones”, este sección es importante ya que nos ayudara a consultar los datos históricos del Bus que se desee seleccionar, aquí podremos obtener más detalle del recorrido de cada unidad, donde se podrá evidenciar los desvíos del bus si es el caso o evidenciar el correcto recorrido de la unidad por su ruta establecida, estos detalles se mostraran en una tabla con datos en longitud y latitud, también en un mapa con las ubicaciones históricas.

Para el ingrese de una nueva parada, lo podremos realizar en la parte final de la aplicación donde digitaremos el nombre de la parada como sus coordenadas exactas en latitud y longitud como también el id de la ruta a la que pertenece la parada.

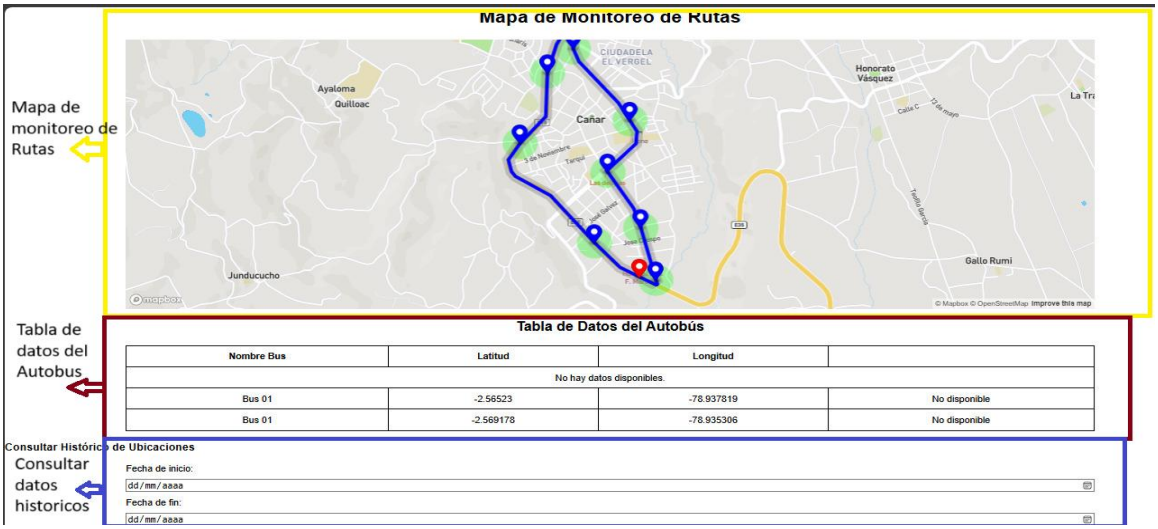


Figura 3. 3 Apartados aplicación WEB. (Fuente: Autor)

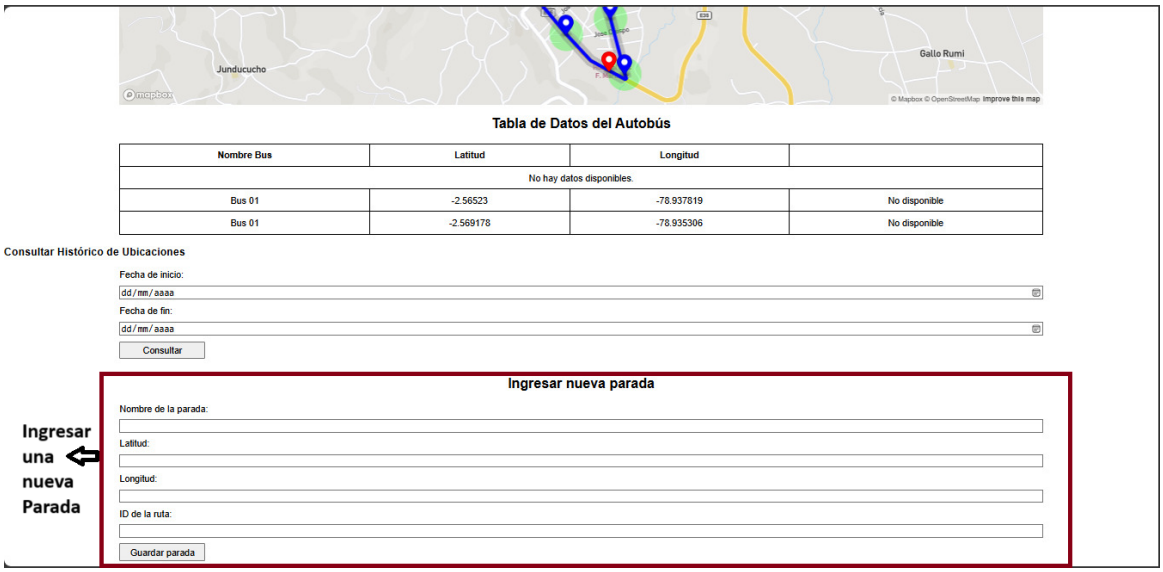


Figura 3. 4 Apartados aplicación WEB. (Fuente: Autor)

En la Figura 2.5 podremos observar el funcionamiento del sistema de alertas cuando una unidad de transporte sale de la zona de la ruta establecida, el sistema de alerta se activa si las coordenadas del autobús enviadas a nosotros en tiempo real cambian respecto a la línea poligonal que conecta las paradas de la ruta asignada a esa unidad de transporte.

Esto se hace utilizando la biblioteca Turf.js para monitorizar. Se usa la biblioteca de JavaScript para calcular la distancia de la coordenada actual del autobús al área de detección, la cual está definida alrededor de las paradas. El área de detección tiene un radio específico y también está vinculada a una parada determinada en la ruta. Luego, si la distancia desde la ubicación actual del autobús al centro de una de estas áreas es mayor que dos niveles, es posible considerar que el autobús está fuera de su zona esperada y, por lo tanto, se puede activar una alarma para informar a los administradores o prevenir errores para el conductor, usuarios o la persona encargada del monitoreo del sistema.

Esto permite controlar el recorrido con precisión, haciendo que el sistema de transporte sea más eficiente y seguro.

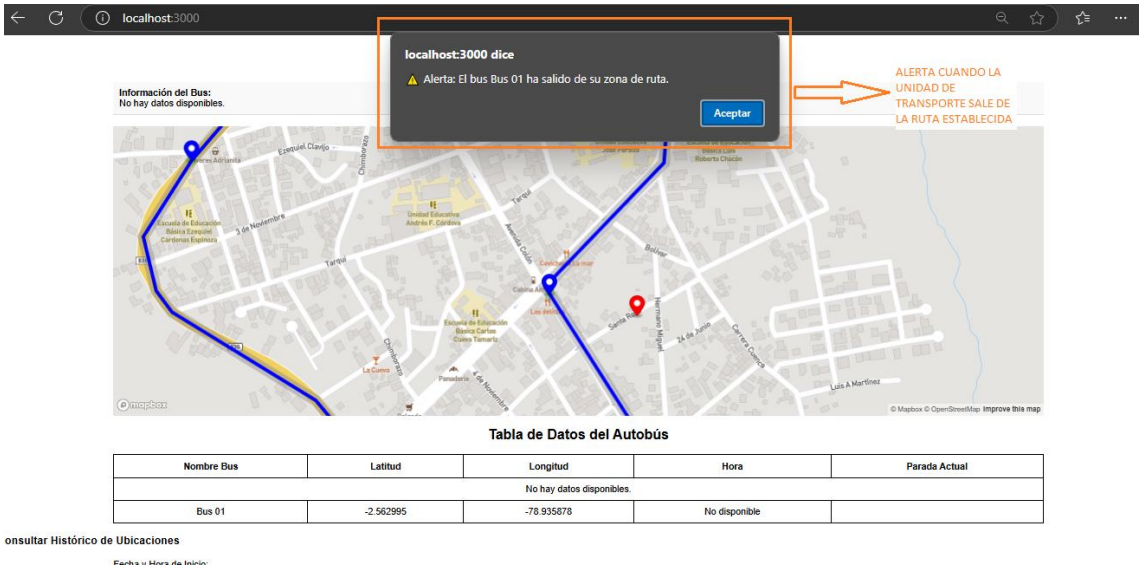


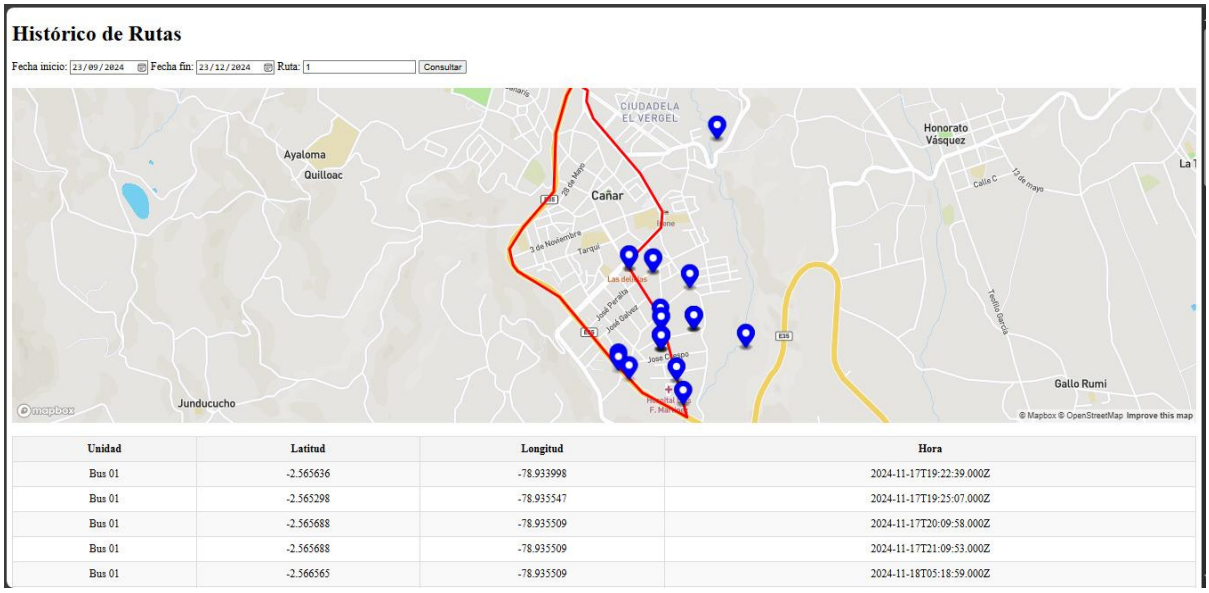
Figura 3. 5 Sistemas de Alerta de desvíos. (Fuente: Autor)

Para verificar los datos históricos utilizaremos el apartado de “Consultar datos Historicos” donde podremos filtrar por fecha estos datos de los que se quiere observar, al ingresar a este apartado ingresaremos a una nueva pestaña que podemos observar en la Figura 3. 6.

Como se puede notar, hay tres secciones: el "Header", que utilizaremos para seleccionar el rango de fechas y la ruta del autobús asociada a la consulta que deseamos realizar.

El “Mapa Ubicaciones Históricas” aquí se graficará todas las ubicaciones que ha recorrido la unidad de transporte asociada a la ruta que fue previamente filtrada en el Heider.

Finalmente, tenemos una tabla donde obtendremos los datos con más exactitud de las ubicaciones en longitud y latitud que ha recorrido el medio de transporte.



Unidad	Latitud	Longitud	Hora
Bus 01	-2.565636	-78.933998	2024-11-17T19:22:39.000Z
Bus 01	-2.565298	-78.935547	2024-11-17T19:25:07.000Z
Bus 01	-2.565688	-78.935509	2024-11-17T20:09:58.000Z
Bus 01	-2.565688	-78.935509	2024-11-17T21:09:53.000Z
Bus 01	-2.566565	-78.935509	2024-11-18T05:18:59.000Z

Figura 3. 6 Pestaña Datos Históricos con Fechas Filtradas. (Fuente: Autor)

La información recolectada durante se presentan en la Figura 3.7. En esta imagen, podemos observar el conjunto de datos con las fechas asociadas a esos días, donde las columnas especifican la unidad de transporte, así como sus coordenadas de longitud y latitud en relación con la hora de escaneo. Los registros vacíos (Null) se deben a fallos en la comunicación, puesta la conexión con la red 4G era limitada en ciertas ubicaciones de la ciudad, lo que provocaba que el dispositivo se interrumpiera su transmisión por breves períodos.

←T→		id	unidad_bus	ruta_id	latitud	longitud	parada_siguiente	timestamp Fecha y hora
☐	Editar Copiar Borrar	164	Bus 01	1	-2.565636	-78.933998	Parada 2	2024-11-17 14:22:39
☐	Editar Copiar Borrar	165	Bus 01	1	-2.565298	-78.935547	Parada 2	2024-11-17 14:25:07
☐	Editar Copiar Borrar	166	Bus 01	1	-2.565688	-78.935509	Parada 2	2024-11-17 15:09:58
☐	Editar Copiar Borrar	167	Bus 01	1	-2.565688	-78.935509	Parada 2	2024-11-17 16:09:53
☐	Editar Copiar Borrar	168	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 00:18:59
☐	Editar Copiar Borrar	169	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 20:24:36
☐	Editar Copiar Borrar	170	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 20:24:54
☐	Editar Copiar Borrar	171	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 20:36:42
☐	Editar Copiar Borrar	172	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 20:37:13
☐	Editar Copiar Borrar	173	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 20:37:28
☐	Editar Copiar Borrar	174	Bus 01	1	-2.566565	-78.935509	Parada 2	2024-11-18 20:41:00
☐	Editar Copiar Borrar	175	Bus 01	1	-2.563690	-78.934166	Parada 2	2024-11-18 20:43:38
☐	Editar Copiar Borrar	176	Bus 01	NULL	-2.563690	-78.934166	Parada 2	2024-11-20 11:13:22
☐	Editar Copiar Borrar	177	Bus 01	NULL	-2.565636	-78.933998	Parada 2	2024-11-20 11:23:33
☐	Editar Copiar Borrar	178	Bus 01	NULL	-2.565636	-78.933998	Parada 2	2024-11-20 11:24:04
☐	Editar Copiar Borrar	179	Bus 01	1	-2.565636	-78.933998	Parada 2	2024-11-20 11:25:10
☐	Editar Copiar Borrar	180	Bus 01	1	-2.565636	-78.933998	Parada 2	2024-11-20 11:25:23
☐	Editar Copiar Borrar	181	Bus 01	NULL	-2.562838	-78.936989	NULL	2024-11-22 10:13:18
☐	Editar Copiar Borrar	182	Bus 01	NULL	-2.562838	-78.936989	NULL	2024-11-22 11:39:53
☐	Editar Copiar Borrar	183	Bus 01	NULL	-2.562838	-78.936989	NULL	2024-11-22 11:51:52
☐	Editar Copiar Borrar	184	Bus 01	NULL	-2.566465	-78.931564	NULL	2024-11-22 11:52:22
☐	Editar Copiar Borrar	185	Bus 01	NULL	-2.566465	-78.931564	NULL	2024-11-25 14:48:38
☐	Editar Copiar Borrar	186	Bus 01	NULL	-2.567377	-78.937508	NULL	2024-11-25 14:50:03
☐	Editar Copiar Borrar	187	Bus 01	NULL	-2.567546	-78.937515	NULL	2024-11-25 14:50:32
☐	Editar Copiar Borrar	188	Bus 01	NULL	-2.573670	-78.938568	NULL	2024-11-25 14:51:34
Consola	itar Copiar Borrar	189	Bus 01	NULL	-2.573670	-78.938568	NULL	2024-11-28 10:45:40

Figura 3. 7 Dataset.

En la siguiente tabla se detalla los valores de la implementación para el sistema por cada unidad de transporte público.

Tabla 3. 1 Detalle de Coste del Sistema por unidad de Bus

CANTIDAD	DESCRIPCION	COSTO
1	RASPBERRY Pi	$130
1	MODEM 4G	$70
1	DATOS MOVILES	$18
1	GPS	$60
1	ANTENA PARA EL GPS	$10
1	CAJA PARA EL SOPORTE PARA ELEMENTOS DEL SISTEMA	$60
1	ADAPTADOR DE BATERIAS CON LA RASPBERRY Pi	$20
	BATERIAS DE LITIO	$30
	TOTAL	$408

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

La simulación como la implementación del dispositivo en el transporte público en la ciudad de Cañar ha representado un avance para el monitoreo en tiempo real y la gestión eficiente de las unidades de transporte. A través del conocimiento de tecnologías como Turf.js, ha sido posible supervisar cada ruta con más precisión y emitir alertas cuando esto sucede. Con la implementación de estas características, la seguridad y operatividad del servicio evidentemente estaría más supervisado y monitoreado para cumplir con las normativas en el reglamento municipal, demostrando cómo la tecnología puede cambiar la manera en que se gestiona el transporte público.

Las consultas resultantes realizada al servidor local sobre las ubicaciones históricas de las unidades de transporte prueban el funcionamiento del sistema, validando su capacidad para supervisar rutas y detectar anomalías. Sin embargo, hubo algunas limitaciones, como interrupciones en la transmisión de datos en áreas con conectividad 4G inadecuada. Esto resalta la necesidad de actualizar la infraestructura de conectividad para maximizar el rendimiento del sistema en todos los dominios operativos.

Otro detalle importante es que el sistema es flexible y expandible por naturaleza, lo que significa que se pueden añadir nuevas paradas y cambiar rutas con relativa facilidad. Por lo tanto, esto representa una herramienta adaptable y reproducible, con un potencial considerable para su implementación en otros contextos con circunstancias de movilidad urbana similares.

En conclusión, este proyecto es un gran avance en la digitalización del transporte público en Cañar. Además de lograr sus objetivos declarados, sienta las bases para una mayor innovación en la gestión del transporte, resultando en un servicio más sostenible, eficiente y accesible para todos.

Recomendaciones

- Agregar algoritmos de predicción impulsados por IA que estimen cosas como cuándo llega un autobús y si el servicio se verá interrumpido.
- Disminuir el uso de energía de los dispositivos involucrados, por ejemplo, alimentar la Raspberry Pi con recursos solares en lugar de los convencionales.
- Escalar el sistema a ciudades adicionales y realizar comparaciones de rendimiento en diversas condiciones urbanas y rurales.
- Obtener características de interacción con los usuarios finales, desde notificaciones sobre cambios de ruta o retrasos, a través de una aplicación móvil o plataforma web.

BIBLIOGRAFÍA

[1] N. Unidas, “El desafío de la sostenibilidad ambiental en América Latina y el Caribe. Textos seleccionados 2012-2014.” [Online]. Available: www.cepal.org/es/suscripciones

[2] X. Kong, X. Liu, B. Jedari, M. Li, L. Wan, and F. Xia, “Mobile Crowdsourcing in Smart Cities: Technologies, Applications, and Future Challenges,” *IEEE Internet Things J*, vol. 6, no. 5, pp. 8095–8113, Oct. 2019, doi: 10.1109/JIOT.2019.2921879.

[3] M. Shafiulla, P. S. Abdul Lateef Haroon, S. M. Naveed, S. Ahmed, M. Mahika Dhar, and U. Kumar, “Revolutionizing Bus Tracking and Arrival Prediction in Real Time for Both Public and Private Sectors, Powered by Firebase,” in *3rd IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics, ICDCECE 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICDCECE60827.2024.10549731.

[4] J. Ochoa-Zambrano and J. Garbajosa, “Social Internet of Things: Architectural Approaches and Challenges 1.”

[5] A. S. López Borja, “ANÁLISIS DEL COMPORTAMIENTO DEL TRANSPORTE PÚBLICO 1 Análisis Del Comportamiento Del Transporte Público A Nivel Mundial.”

[6] C. De Telecomunicaciones, “UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO,” 2024.

[7] O. Bhat, P. Gokhale, and S. Bhat, “Introduction to IOT,” *International Advanced Research Journal in Science, Engineering and Technology ISO*, vol. 3297, no. 1, 2007, doi: 10.17148/IARJSET.2018.517.

[8] I. Latin and A. Transactions, “MQTT Protocol: Fundamentals, Tools and Future Directions,” 2019.

[9] P. John, J. Hynek, T. Hruška, and M. Valný, “Application of Time Series Database for IoT Smart City Platform,” in *2023 Smart Cities Symposium Prague, SCSP 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/SCSP58044.2023.10146237.

[10] “ALTERNATIVA TECNOLÓGICA CON GEO POSICIONAMIENTO”.

[11] Á. A. Guzmán Castañeda, J. W. Pinto Uribe, and D. A. Velosa Castañeda, “Implementación de un sistema de geoposicionamiento con transmisión de

coordenadas por radiofrecuencia," *Perspectivas en Inteligencia*, vol. 12, no. 21, pp. 275–291, Oct. 2021, doi: 10.47961/2145194x.227.

APÉNDICES

APÉNDICE A: Back END

```
document.addEventListener('DOMContentLoaded', () => {
                                                    mapboxgl.accessToken                    =
'pk.eyJ1IjoianVhbnZhenF2IiwiYSI6ImNtMnpucW00NDBjYXYyaXEyeW5tdjZtZHUifQ.ho
hudabBeifjNIvfQHU4mQ';

    window.map = new mapboxgl.Map({
        container: 'map',
        style: 'mapbox://styles/mapbox/streets-v11',
        center: [-78.93705, -2.56269],
        zoom: 13,
    });

    window.markers = {};
    window.routes = {};
    window.zonas = {};

    // Esperar a que el mapa cargue
    window.map.on('load', () => {
        console.log('Mapa cargado correctamente');
        cargarParadas(); // Llama a la función que ya tienes fuera del evento
    });

    const socket = io();

    // Escuchar actualizaciones en tiempo real
    socket.on('ubicacionBusActualizada', (data) => {
        actualizarUbicacionBus(data); // Usa la función que ya tienes definida fuera
        actualizarTablaBuses(data);  // Usa la función que ya tienes definida fuera
```

```
});

socket.on('paradasActualizadas', () => {
    cargarParadas(); // Llama a la función que ya tienes definida fuera
});

socket.on('ubicacion-actualizada', (data) => {
    const { unidad_bus, latitud, longitud, proxima_parada } = data;

    // Actualizar los datos del bus en la parte superior
    const busDetailsElement = document.getElementById('bus-details');
    busDetailsElement.innerHTML = `
        <p>Unidad: <strong>${unidad_bus}</strong></p>
        <p>Latitud: <strong>${latitud}</strong></p>
        <p>Longitud: <strong>${longitud}</strong></p>
        <p>Próxima parada: <strong>${proxima_parada || 'No disponible'}</strong></p>
    `;
});

// Manejo del botón para consultar histórico
const btnHistorico = document.getElementById('btn-consultar-historico');
if (btnHistorico) {
    btnHistorico.addEventListener('click', (event) => {
        event.preventDefault(); // Evita la recarga de la página.

        const fechaInicio = document.getElementById('fecha-inicio').value;
        const fechaFin = document.getElementById('fecha-fin').value;

        if (!fechaInicio || !fechaFin) {
            alert('Por favor, selecciona ambas fechas para consultar el histórico.');
            return;
        }
```

```
        // Generar la URL con los parámetros
                                                    const url =
`/historico.html?fechaInicio=${encodeURIComponent(fechaInicio)}&fechaFin=${encode
URIComponent(fechaFin)}`;

        // Abrir la nueva página en una pestaña
        window.open(url, '_blank');
    });
}

if (window.location.pathname === '/historico.html') {
                                        mapboxgl.accessToken =
'pk.eyJ1IjoianVhbnZhenF2IiwiYSI6ImNtMnpucW00NDBjYXYyaXEyeW5tdjZtZHUifQ.ho
hudabBeifjNIvfQHU4mQ'; // Reemplaza con tu token

const map = new mapboxgl.Map({
    container: 'map',
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [-78.4678, -0.1807],
    zoom: 12
});

const tablaHistorico = document.getElementById('tabla-historico');

async function cargarDatosHistoricos() {
    try {
            const response = await fetch('/api/historico?fechaInicio=2024-01-
01&fechaFin=2024-01-31');
        const datos = await response.json();

        tablaHistorico.innerHTML = '';
```

```
        const bounds = new mapboxgl.LngLatBounds();

        datos.forEach(dato => {
            const fila = document.createElement('tr');
            fila.innerHTML = `
                <td>${dato.unidad_bus}</td>
                <td>${dato.latitud}</td>
                <td>${dato.longitud}</td>
                <td>${dato.timestamp}</td>
                <td>${dato.enRuta ? 'En ruta' : 'Fuera de ruta'}</td>
            `;
            if (!dato.enRuta) fila.classList.add('fuera-de-ruta');
            tablaHistorico.appendChild(fila);

            const color = dato.enRuta ? '#007cbf' : '#ff0000';
            const marker = new mapboxgl.Marker({ color })
                .setLngLat([dato.longitud, dato.latitud])
                .setPopup(new mapboxgl.Popup().setHTML(`
                    <strong>Unidad:</strong> ${dato.unidad_bus}<br>
                    <strong>Fecha/Hora:</strong> ${dato.timestamp}<br>
                    <strong>Estado:</strong> ${dato.enRuta ? 'En ruta' : 'Fuera de ruta'}
                `))
                .addTo(map);

            bounds.extend([dato.longitud, dato.latitud]);
        });

        if (datos.length > 0) map.fitBounds(bounds, { padding: 50 });
    } catch (error) {
        console.error('Error al cargar los datos históricos:', error);
    }
}
```

```
    cargarDatosHistoricos();
}


});

// Función para limpiar zonas previas del mapa
function limpiarZonas() {
    Object.keys(window.zonas || {}).forEach((idZona) => {
        if (window.map.getSource(idZona)) {
            window.map.removeLayer(idZona);
            window.map.removeSource(idZona);
        }
    });

    window.zonas = {};
}

// Función para cargar ubicaciones iniciales de los buses desde el backend
function cargarUbicacionesBuses() {
    fetch('/api/obtener-ubicaciones-buses')
        .then(response => response.json())
        .then(buses => buses.forEach(bus => actualizarUbicacionBus(bus)))
        .catch(err => console.error('Error al cargar ubicaciones de buses:', err));
}

// Función para actualizar la ubicación del bus en tiempo real
function actualizarUbicacionBus({ unidad_bus, latitud, longitud }) {
    if (!unidad_bus || isNaN(latitud) || isNaN(longitud)) {
        console.error('Datos de bus inválidos:', { unidad_bus, latitud, longitud });
        return;
```

```
    }

    // Actualizar o crear marcador del bus
    if (window.markers[unidad_bus]) {
        window.markers[unidad_bus].setLngLat([longitud, latitud]);
    } else {
        window.markers[unidad_bus] = new mapboxgl.Marker({ color: 'red' })
            .setLngLat([longitud, latitud])
            .setPopup(new mapboxgl.Popup().setHTML(`<b>Bus: ${unidad_bus}</b>`))
            .addTo(window.map);
    }

    // Verificar si el bus está dentro de al menos una zona
    let dentroDeZona = false;
    Object.values(window.zonas || {}).forEach(zona => {
        if (estaDentroDeZona(latitud, longitud, zona)) {
            dentroDeZona = true; // Si está dentro de una zona, marcamos como verdadero
        }
    });

    if (!dentroDeZona) {
        // El bus está fuera de todas las zonas
        console.warn(`⚠ El bus ${unidad_bus} está fuera de su ruta asignada.`);
        alert(`⚠ Alerta: El bus ${unidad_bus} ha salido de su zona de ruta.`);
    }
}

function verificarZonaDeteccion(latitudBus, longitudBus, unidadBus) {
    Object.values(window.zonas || {}).forEach((zona) => {
        const distancia = turf.distance(
            turf.point([longitudBus, latitudBus]),
            turf.point(zona.geometry.coordinates),
```

```
            { units: 'meters' }
        );

        if (distancia <= zona.properties.radius) {
                        console.log(`El bus ${unidadBus} está dentro de la zona: ${zona.geometry.coordinates}`);
        }
    });
}

// Función para cargar paradas y dibujar polilíneas
function cargarParadas() {
    fetch('/api/obtener-paradas')
        .then((response) => response.json())
        .then((paradas) => {
            limpiarZonas();
            paradas.forEach((parada) => {
                agregarParadaAlMapa(parada); // Usamos la función de agregar la parada en el mapa
            });

            // Llamamos a trazar rutas después de cargar todas las paradas
            trazarRutas();  // Esta nueva función se encargará de trazar las rutas
        })
        .catch((err) => console.error('Error al cargar paradas:', err));
}

function trazarRutas() {
    fetch('/api/obtener-paradas-ruta/1') // Suponiendo que estamos trazando la ruta 1, ajusta según sea necesario
        .then((response) => response.json())
        .then((paradasRuta) => {
```

```
            const coordenadasRuta = paradasRuta.map((parada) => [parada.longitud, parada.latitud]);
            trazarRutaConMapbox(coordenadasRuta);
        })
        .catch((err) => console.error('Error al obtener las paradas de la ruta:', err));
}

function agregarParadaAlMapa(parada) {
    const { id, nombre_parada, latitud, longitud } = parada;

    // Crear el marcador de la parada
    new mapboxgl.Marker({ color: 'blue' })
        .setLngLat([longitud, latitud])
        .setPopup(new mapboxgl.Popup().setHTML(`<b>${nombre_parada}</b>`))
        .addTo(window.map);

    // Agregar la zona de detección en el mapa
    window.zonas[`zona-${id}`] = crearZonaDeDeteccion(latitud, longitud, 200); // Usamos `id` aquí como `zona-${id}`
}

// Función para crear zonas de deteccion de las unidades de transporte
function crearZonaDeDeteccion(latitud, longitud, radioMetros) {
    const idZona = `zona-${latitud}-${longitud}`;
    const circulo = {
        type: 'Feature',
        geometry: {
            type: 'Point',
            coordinates: [longitud, latitud],
        },
        properties: { radius: radioMetros },
    };
```

```
    window.map.addSource(idZona, { type: 'geojson', data: circulo });

    window.map.addLayer({
        id: idZona,
        type: 'circle',
        source: idZona,
        paint: {
            'circle-radius': 30,
            'circle-color': '#00ff00',
            'circle-opacity': 0.3,
        },
    });

    return circulo;
}

function estaDentroDeZona(latitudBus, longitudBus, zona) {
    const distancia = turf.distance(
        turf.point([longitudBus, latitudBus]),
        turf.point(zona.geometry.coordinates),
        { units: 'meters' }
    );
    return distancia <= zona.properties.radius;
}

// Función para trazar rutas utilizando Mapbox Directions
async function trazarRutaConMapbox(coordenadas) {
                                        const                    url                    =
`https://api.mapbox.com/directions/v5/mapbox/driving/${coordenadas.map(coord       =>
coord.join(',')).join(';')}?geometries=geojson&access_token=${mapboxgl.accessToken}`;
```

```
try {
    const response = await fetch(url);
    const data = await response.json();

    if (data.routes && data.routes.length > 0) {
        const ruta = data.routes[0].geometry.coordinates;

        // Remover capa y fuente si ya existen
        if (window.map.getSource('ruta')) {
            window.map.removeLayer('ruta-layer');
            window.map.removeLayer('ruta-border-layer');
            window.map.removeSource('ruta');
        }

        // Agregar nueva fuente
        window.map.addSource('ruta', {
            type: 'geojson',
            data: {
                type: 'Feature',
                geometry: {
                    type: 'LineString',
                    coordinates: ruta
                }
            }
        });

        // Capa para bordes más gruesos (simula ancho de la calle)
        window.map.addLayer({
            id: 'ruta-border-layer',
            type: 'line',
            source: 'ruta',
            paint: {
```

```
                'line-color': '#808080', // Color del borde (gris)
                'line-width': 20, // Ancho del borde
                'line-opacity': 0.4
            }
        });

        // Capa para la ruta principal
        window.map.addLayer({
            id: 'ruta-layer',
            type: 'line',
            source: 'ruta',
            paint: {
                'line-color': '#0000FF', // Color de la línea principal (azul)
                'line-width': 6, // Ancho de la línea
            }
        });
    } else {
        console.error('No se encontraron rutas:', data);
    }
} catch (error) {
    console.error('Error al trazar la ruta:', error);
}
}

function guardarParada(event) {
    event.preventDefault();

    const nombreParada = document.getElementById('nombre_parada').value;
    const latitud = parseFloat(document.getElementById('latitud').value);
    const longitud = parseFloat(document.getElementById('longitud').value);
   const rutaId = parseInt(document.getElementById('ruta_id').value); // Asegúrate de que esto esté en el formulario correctamente
```

```
    if (!nombreParada || isNaN(latitud) || isNaN(longitud) || isNaN(rutaId)) {
        alert('Por favor, completa todos los campos correctamente.');
        return;
    }

    const data = { nombre_parada: nombreParada, latitud, longitud, ruta_id: rutaId }; // Usa
`rutaId` aquí

    fetch('/api/guardar-parada', {
        method: 'POST',
        body: JSON.stringify(data),
        headers: { 'Content-Type': 'application/json' }
    })
        .then(response => {
            if (response.ok) {
                alert('Parada guardada correctamente.');
                cargarParadas(); // Recarga las paradas
            } else {
                alert('Error al guardar la parada.');
            }
        })
        .catch(err => console.error('Error al guardar la parada:', err));
}

function actualizarTablaBuses({ unidad_bus, latitud, longitud, proxima_parada }) {
    const tableBody = document.getElementById('bus-table-body'); // Asegúrate de tener
este ID en tu HTML
    const row = document.createElement('tr');
    row.innerHTML = `
        <td>${unidad_bus}</td>
        <td>${latitud}</td>
```

```
        <td>${longitud}</td>
        <td>${proxima_parada || 'No disponible'}</td>
    `;
    tableBody.appendChild(row);
}
```

APÉNDICE B: Front END

```
const express = require('express');
const mysql = require('mysql');
const cors = require('cors');
const path = require('path');
const http = require('http');
const socketIo = require('socket.io');

// Configuración de Express y Socket.IO
const app = express();
const server = http.createServer(app);
const io = socketIo(server);

app.use(cors());
app.use(express.json());
app.use(express.static(path.join(__dirname)));

// Configuración de la base de datos
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'transporte_urbano',
});
```

```
db.connect((err) => {
    if (err) {
        console.error('Error al conectar a la base de datos:', err);
        return;
    }
    console.log('Conexión a la base de datos exitosa');
});

app.get('/api/obtener-paradas-ruta/:idRuta', (req, res) => {
    const idRuta = req.params.idRuta;

    const query = `
        SELECT p.*
        FROM paradas_buses p
        JOIN rutas_paradas rp ON rp.parada_id = p.id
        WHERE rp.ruta_id = ?
        ORDER BY rp.orden ASC;
    `;

    db.query(query, [idRuta], (error, resultados) => {
        if (error) {
            console.error('Error al obtener las paradas:', error);
            return res.status(500).json({ error: 'Error interno del servidor' });
        }
        res.json(resultados);
    });
});

app.get('/api/obtener-paradas', (req, res) => {
    const sql = 'SELECT * FROM paradas_buses';
    db.query(sql, (err, results) => {
        if (err) {
```

```
            console.error('Error al obtener paradas:', err);
            return res.status(500).json({ error: 'Error al obtener paradas' });
        }
        res.json(results);
    });
});
// Ruta para guardar una nueva parada de bus

app.post('/api/guardar-parada', (req, res) => {
    const { nombre_parada, latitud, longitud } = req.body;
    if (!nombre_parada || !latitud || !longitud) {
        return res.status(400).json({ error: 'Faltan datos obligatorios' });
    }

    const sql = 'INSERT INTO paradas_buses (nombre_parada, latitud, longitud) VALUES
(?, ?, ?)';
    db.query(sql, [nombre_parada, latitud, longitud], (err, result) => {
        if (err) {
            console.error('Error al guardar parada:', err);
            res.status(500).json({ error: 'Error al guardar parada' });
        } else {
            io.emit('paradasActualizadas');
             res.json({ message: 'Parada guardada correctamente', paradaId: result.insertId
});
        }
    });
})
// Ruta para obtener ubicaciones de los buses

app.get('/api/obtener-ubicaciones-buses', (req, res) => {
    const sql = 'SELECT unidad_bus, latitud, longitud FROM datos_buses';
```

```
    db.query(sql, (err, results) => {
        if (err) {
            console.error('Error al obtener ubicaciones:', err);
            res.status(500).json({ error: 'Error al obtener ubicaciones' });
        } else {
            res.json(results);
        }
    });
});

app.post('/api/guardar-ubicacion-bus', (req, res) => {
    const { unidad_bus, latitud, longitud } = req.body;

    const sql = 'INSERT INTO datos_buses (unidad_bus, latitud, longitud) VALUES (?, ?, ?)';
    db.query(sql, [unidad_bus, latitud, longitud], (err) => {
        if (err) {
            console.error('Error al guardar ubicación:', err);
            return res.status(500).json({ error: 'Error al guardar ubicación' });
        }
        io.emit('ubicacionBusActualizada', { unidad_bus, latitud, longitud });
        res.json({ message: 'Ubicación guardada correctamente' });
    });
});

app.get('/api/ubicaciones-buses-por-ruta/:ruta_id', (req, res) => {
    const { ruta_id } = req.params;

      const sql = 'SELECT unidad_bus, latitud, longitud, parada_siguiente FROM datos_buses WHERE ruta_id = ?';
    db.query(sql, [ruta_id], (err, results) => {
        if (err) {
```

```
            console.error('Error al obtener ubicaciones de buses por ruta:', err);
            return res.status(500).json({ error: 'Error interno del servidor' });
        }
        res.json(results);
    });
});

app.post('/api/asociar-paradas-ruta', (req, res) => {
    const { ruta_id, paradas } = req.body; // paradas será un array [{ parada_id, orden }, ...]

    if (!ruta_id || !paradas || paradas.length === 0) {
        return res.status(400).json({ error: 'Faltan datos necesarios' });
    }

    const values = paradas.map(p => [ruta_id, p.parada_id, p.orden]);
    const sql = 'INSERT INTO rutas_paradas (ruta_id, parada_id, orden) VALUES ?';

    db.query(sql, [values], (err, result) => {
        if (err) {
            console.error('Error al asociar paradas a la ruta:', err);
            return res.status(500).json({ error: 'Error al asociar paradas' });
        }
        res.status(201).json({ message: 'Paradas asociadas correctamente', affectedRows:
result.affectedRows });
    });
});

app.get('/api/ruta-por-bus/:unidad_bus', (req, res) => {
    const { unidad_bus } = req.params;

    const sql = `
        SELECT r.*
```

```
        FROM rutas r
        INNER JOIN datos_buses b ON r.id = b.ruta_id
        WHERE b.unidad_bus = ?
    `;

    db.query(sql, [unidad_bus], (err, results) => {
        if (err) {
            console.error('Error al obtener la ruta:', err);
            return res.status(500).json({ error: 'Error al obtener la ruta del bus' });
        }

        if (results.length === 0) {
            return res.status(404).json({ error: 'No se encontró ninguna ruta para este bus' });
        }

        res.json(results[0]); // Enviar la ruta asociada
    });
});

app.get('/api/historico-buses', (req, res) => {
    const { fechaInicio, fechaFin } = req.query;

    // Validar que las fechas existen
    if (!fechaInicio || !fechaFin) {
        return res.status(400).json({ error: 'Debe proporcionar fechaInicio y fechaFin' });
    }

    // Formatear las fechas para que coincidan con el formato de MySQL (YYYY-MM-DD HH:MM:SS)
    // Si solo estás usando fechas sin hora, puedes agregar una hora por defecto como '00:00:00'
    const fechaInicioFormateada = `${fechaInicio} 00:00:00`;
```

```
    const fechaFinFormateada = `${fechaFin} 23:59:59`;

    // Realizar la consulta a la base de datos para obtener las ubicaciones históricas
    const query = `
        SELECT unidad_bus, latitud, longitud, timestamp
        FROM datos_buses
        WHERE timestamp BETWEEN ? AND ?
    `;

    db.query(query, [fechaInicioFormateada, fechaFinFormateada], (err, results) => {
        if (err) {
            console.error('Error al consultar la base de datos:', err);
            return res.status(500).json({ error: 'Error del servidor' });
        }

         // Si no se encuentran resultados, podemos devolver una respuesta vacía o un mensaje adecuado
        if (results.length === 0) {
                return res.status(404).json({ message: 'No se encontraron registros para las fechas indicadas' });
        }

        // Enviar los resultados encontrados
        res.json(results);
    });
});

app.get('/api/historico', (req, res) => {
    const { fechaInicio, fechaFin } = req.query;

    // Verificar si las fechas están presentes y son válidas
    if (!fechaInicio || !fechaFin) {
```

```
        return res.status(400).json({ error: 'Faltan parámetros de fecha.' });
    }

    const fechaInicioDate = new Date(fechaInicio);
    const fechaFinDate = new Date(fechaFin);

    if (isNaN(fechaInicioDate) || isNaN(fechaFinDate)) {
        return res.status(400).json({ error: 'Las fechas proporcionadas no son válidas.' });
    }

    const query = `
        SELECT unidad_bus, latitud, longitud, timestamp
        FROM datos_buses
        WHERE timestamp BETWEEN ? AND ?
        ORDER BY timestamp ASC
    `;

    // Asegurarse de pasar las fechas en un formato que la base de datos entienda
        db.query(query,  [fechaInicioDate.toISOString(),  fechaFinDate.toISOString()],  (err,
results) => {
        if (err) {
            console.error('Error al consultar datos históricos:', err);
            return res.status(500).json({ error: 'Error interno del servidor.' });
        }

        res.json(results);
    });
});

// Iniciar el servidor
server.listen(3000, () => {
    console.log('Servidor escuchando en el puerto 3000');
```

});

APÉNDICE C: Visualizacion de Datos

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mapa de Monitoreo de Rutas</title>
                <link        href="https://api.mapbox.com/mapbox-gl-js/v2.10.0/mapbox-gl.css"
rel="stylesheet" />
    <style>
         body, html { margin: 0; padding: 0; width: 100%; height: 100%; font-family: Arial,
sans-serif; }
        #bus-info { padding: 10px; background-color: #f9f9f9; border-bottom: 1px solid #ddd;
width: 80%; margin: 20px auto 0 auto; box-sizing: border-box; }
        #map { width: 80%; height: 500px; margin: 20px auto; display: block; }
        table { width: 80%; margin: 20px auto; border-collapse: collapse; text-align: center;
}
        table, th, td { border: 1px solid black; }
        th, td { padding: 10px; }
         #alertas { color: red; font-weight: bold; margin: 20px auto; width: 80%; text-align:
center; }
        h1, h2 { text-align: center; }
        form { width: 80%; margin: 20px auto; display: flex; flex-direction: column; }
        label, input, button { margin-bottom: 10px; font-size: 16px; }
        button { width: 150px; align-self: flex-start; padding: 5px 10px; }
    </style>
</head>
<body>
    <h1>Mapa de Monitoreo de Rutas</h1>
```

```
<div id="map"></div>
<div id="alertas"></div>
<h2>Tabla de Datos del Autobús</h2>
<table id="bus-details">
    <thead>
        <tr>
            <th>Nombre Bus</th>
            <th>Latitud</th>
            <th>Longitud</th>
        </tr>
    </thead>
    <tbody id="bus-table-body">
        <tr><td colspan="5">No hay datos disponibles.</td></tr>
    </tbody>
</table>
<div id="consulta-historico">
    <h3>Consultar Histórico de Ubicaciones</h3>
    <form id="form-historico" onsubmit="openHistorico(event)">
        <label for="fecha-inicio">Fecha de inicio:</label>
        <input type="date" id="fecha-inicio" name="fechaInicio" required>
        <label for="fecha-fin">Fecha de fin:</label>
        <input type="date" id="fecha-fin" name="fechaFin" required>
        <button type="submit">Consultar</button>
    </form>

    <script>
        function openHistorico(event) {
            event.preventDefault(); // Evita el comportamiento por defecto del formulario
            const fechaInicio = document.getElementById('fecha-inicio').value;
            const fechaFin = document.getElementById('fecha-fin').value;

            if (!fechaInicio || !fechaFin) {
```

```
            alert('Por favor, ingresa las fechas de inicio y fin.');
            return;
        }

        // Crea la URL con los parámetros
        const url = `historico.html?fechaInicio=${fechaInicio}&fechaFin=${fechaFin}`;

        // Abre la nueva pestaña
        window.open(url, '_blank');
    }
</script>


<div id="resultado-historico"></div>
</div>

<h2>Ingresar nueva parada</h2>
<form id="formParada">
    <label for="nombre_parada">Nombre de la parada:</label>
    <input type="text" id="nombre_parada" name="nombre_parada" required />
    <label for="latitud">Latitud:</label>
    <input type="number" id="latitud" name="latitud" step="any" required />
    <label for="longitud">Longitud:</label>
    <input type="number" id="longitud" name="longitud" step="any" required />
    <label for="ruta_id">ID de la ruta:</label>
    <input type="number" id="ruta_id" name="ruta_id" required />
    <button type="submit">Guardar parada</button>
</form>
<script src="https://api.mapbox.com/mapbox-gl-js/v2.10.0/mapbox-gl.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Turf.js/6.5.0/turf.min.js"></script>
<script src="/socket.io/socket.io.js"></script>
<script src="app.js" defer></script>
```

</body>
</html>