UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DEL SOFTWARE

Gemelo digital del servicio de autobuses urbanos de Málaga

Digital twin of the Málaga's city bus service

Realizado por
Daniel Roura Sepúlveda

Tutorizado por
José Carlos Canal Velasco

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, MAYO DE 2023

Fecha defensa: junio de 2023

Resumen

Este TFG tiene como objetivo principal desarrollar un sistema de gestión de autobuses basado en un gemelo digital, aprovechando tecnologías de recopilación y análisis de datos. El gemelo digital es una réplica virtual del sistema de autobuses en tiempo real, lo que permite realizar simulaciones para tomar decisiones informadas. Se recopilan datos actualizados cada minuto, como la ubicación de los autobuses y su estado, para obtener una visión precisa de la operación en curso. El trabajo utiliza tecnologías avanzadas, como la computación en la nube y el análisis de grandes cantidades de datos, para gestionar grandes volúmenes de información y realizar cálculos complejos. La aplicación cuenta con un módulo de visualización que proporciona la información sobre la ubicación y el rendimiento de los autobuses, lo que facilita la toma de decisiones en tiempo real. Los usuarios pueden acceder a través de una interfaz web que muestra los últimos datos obtenidos, así como un informe de tiempos a distintas paradas.

El desarrollo de este sistema representa un avance significativo en la gestión inteligente del transporte público, brindando un gran avance al servicio de autobuses urbanos de Málaga. Con el gemelo digital y el análisis de datos en tiempo real, se logra una optimización del sistema, una mayor eficiencia en la planificación y la capacidad de tomar decisiones basadas en información precisa. Esto mejora la calidad del servicio, optimiza los recursos disponibles y ofrece una experiencia de transporte más eficiente para los usuarios de autobuses en la ciudad.

Este trabajo se integra dentro de las actividades del proyecto nacional "Desarrollo de aplicaciones para Smart Cities teniendo en cuenta a las personas" (PID2021-125527NB-I00), financiado por el Ministerio de Ciencia e Innovación, dentro del Plan Nacional de Investigación Científica. Los resultados obtenidos en este trabajo se han utilizado para la elaboración de un articulo científico: "Modeling Urban Digital Twins over he Cloud-to-Thing Continuum", enviado a MESS @ STAF2023 (https://conf.researchr.org/home/staf-2023/mess-2023).

Palabras clave:

Gemelo Digital, Análisis de Datos, Optimización de Sistemas, Simulación, Validación

Abstract

The main objective of this TFG is to develop a bus management system based on a digital twin, taking advantage of data collection and analysis technologies. The digital twin is a virtual replica of the bus system in real time, allowing simulations to make informed decisions. Up-to-the-minute data, such as bus location and status, is collected to provide an accurate view of the ongoing operation. The project uses advanced technologies, such as cloud computing and big data analytics, to manage large volumes of information and perform complex calculations. The application features a visualization module that provides information on the location and performance of buses, facilitating real-time decision making. Users can access it through a web interface that displays the latest data obtained, as well as a report of times to different stops.

The development of this system represents a significant advance in the intelligent management of public transport, bringing a breakthrough to the urban bus service in Malaga. With the digital twin and real-time data analysis, system optimization, greater efficiency in planning and the ability to make decisions based on accurate information are achieved. This improves service quality, optimizes available resources, and provides a more efficient transportation experience for bus users in the city.

Keywords:

Digital Twin, Data Analysis, System Optimization, Simulation, Validation.

Índice

1. Introducción

1.1. Objetivos

El objetivo principal de este trabajo es desarrollar un gemelo digital del transporte público de Málaga, concretamente del servicio de líneas de autobuses urbanos, utilizando como recurso principal la plataforma de datos abiertos del ayuntamiento. Para ello, se recolectarán de forma continua los datos abiertos que ofrece el ayuntamiento sobre el estado del sistema. Estos datos se almacenarán en una base de datos alojada en la nube, y se usarán para alimentar un modelo estadístico sobre el funcionamiento y comportamiento del sistema, constituyendo un gemelo digital del servicio de autobuses de Málaga.

Además, se desarrollará una aplicación web para interactuar con el gemelo digital, y los usuarios podrán consultarla desde cualquier dispositivo con un navegador web moderno para conocer el estado y predicciones de funcionamiento del servicio. La aplicación web se adaptará a diferentes formatos y dispositivos de visualización, desde teléfonos móviles hasta televisores. Todo esto se enmarca en el proyecto de investigación “Desarrollo de aplicaciones para Smart Cities teniendo en cuenta a las personas (IPSCA, 2022-2026, PID2021-125527NB-I00)”, financiado por el plan nacional de investigación, y que se desarrolla actualmente en el grupo de investigación del tutor del TFG.

1.2. Gemelos digitales

En esta sección, se explicará de manera concisa el funcionamiento de los gemelos digitales, que son representaciones virtuales precisas de sistemas, objetos o procesos del mundo real. Se analizará el concepto de gemelos digitales, sus diferentes tipos y el proceso de creación, que consta de fases clave como la recopilación de datos, la creación y validación del gemelo, y la simulación y predicción. Con esto se espera poder comprender la importancia y el funcionamiento del TFG.

1.2.1. Introducción al concepto

En la actualidad, la digitalización está transformando la forma en que interactuamos con el mundo físico. Los gemelos digitales son un ejemplo de esta transformación, ya que permiten crear una representación virtual de un sistema, objeto o proceso del mundo real. Estas réplicas son tan precisas que pueden actuar como copias idénticas del sistema u objeto en el que se basan, lo que permite analizar su comportamiento, prever posibles problemas e incluso influir en su funcionamiento en tiempo real. Con esta tecnología, las empresas pueden mejorar la eficiencia y la productividad de sus procesos, así como planificar el futuro de sus productos y servicios.

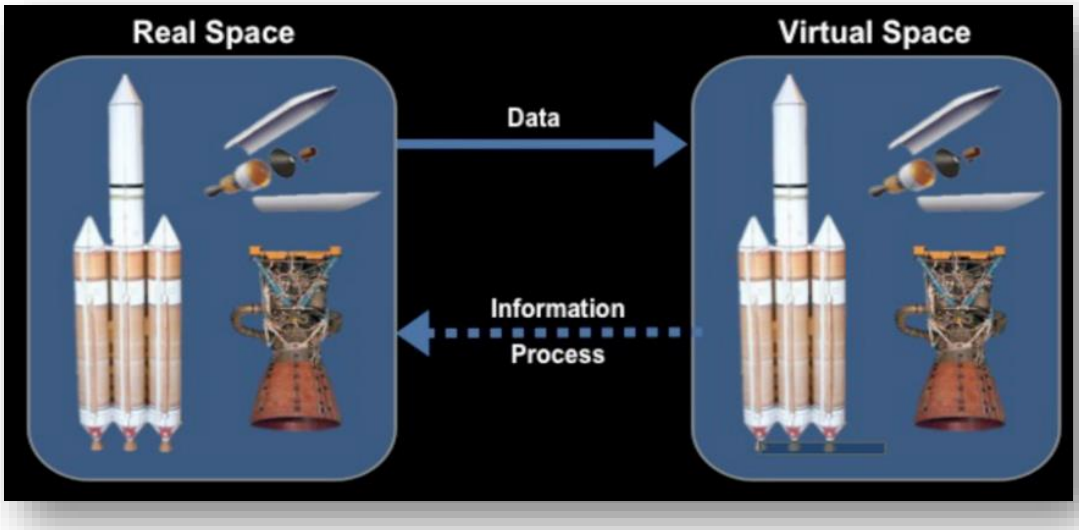


Figura 1. Un primer concepto de gemelo digital, de Grieves y Vickers.

En este contexto es importante conocer que el origen de este tipo de tecnologías se remonta a la década de los 60, cuando la NASA desarrolló un "modelo vivo" (*living model)* de una nave espacial para analizar los eventos que desembocaron en un accidente durante una misión. Este modelo fue la primera versión desarrollada de lo que hoy en día conocemos como gemelo digital.

Hay diferentes tipos de gemelos digitales, entre ellos se encuentran los Gemelos Digitales de componentes, productos, sistemas y procesos. Cada uno se enfoca en crear una representación virtual de lo que a su nombre se refiere, por ejemplo, los Gemelos de sistemas realizan copias digitales de sistemas completos, lo que permite a los expertos elegir el diseño más adecuado de tecnología para el problema que deban solucionar.

1.2.2. Funcionamiento

Para construir un gemelo digital, se requiere seguir un proceso que consta de varias fases clave. En esta explicación, se dividirán en tres secciones para comprender mejor cada una de ellas. Cada fase requiere un enfoque diferente y herramientas específicas para garantizar el correcto funcionamiento del gemelo digital. Estas fases son esencialmente: recopilación de datos, creación y validación del gemelo, y simulación y predicción. Es importante destacar que estas fases se realizan de forma constante, no tienen un punto final ya que el gemelo debe evolucionar y adaptarse a los nuevos cambios del mundo real.

El primer paso para crear un gemelo digital es la recolección de datos. Para asegurar que el gemelo digital sea una representación fiel del sistema u objeto en el mundo real, se necesita recopilar datos tanto históricos como en tiempo real. Para ello, se utilizan sensores, dispositivos IoT (Internet de las cosas) y otras fuentes de información, como APIs que permiten recopilar información como el clima, por ejemplo. Es importante tener información actualizada para que el gemelo digital refleje la realidad y no solo el pasado.

Después de la fase de recopilación de datos, se procede a la creación del gemelo digital, donde se procesan y analizan los datos para crear una copia virtual precisa del sistema u objeto del mundo real. En esta fase, existen varias técnicas que se pueden utilizar,

como modelos matemáticos, inteligencia artificial y aprendizaje automático, entre otros. En este trabajo, se ha optado por un enfoque más manual y matemático que permite realizar ajustes de forma sencilla sobre los análisis y predicciones. Es importante validar el gemelo digital antes de utilizarlo, y para ello se comparan los resultados de las simulaciones con los datos reales obtenidos en pruebas reales.

Una vez que el gemelo digital se valida por primera vez, se pueden realizar simulaciones y predicciones sobre el sistema u objeto en el mundo real. Con los resultados obtenidos de las simulaciones y predicciones se pueden prever posibles problemas y tomar decisiones informadas para mejorar el funcionamiento del sistema real. Por ejemplo, en el caso de un gemelo digital de un autobús urbano, se podría simular todo el recorrido que el autobús debe realizar en un día y prever posibles retrasos o problemas que pudieran surgir en el camino.

1.2.3. Ventajas

Los gemelos digitales presentan diversas ventajas en el ámbito de la simulación y predicción de sistemas y objetos del mundo real. Para empezar, permiten identificar problemas temprano y hacer ajustes antes de que se produzcan fallos graves. En el ámbito de la manufacturación, se pueden utilizar gemelos digitales para simular líneas de creación de productos en un entorno virtual antes de implementarlas en la producción real, lo que permite detectar posibles errores y hacer ajustes antes de que los productos se produzcan en grandes cantidades.

Otra ventaja importante de los gemelos digitales es que permiten la realización de experimentos y simulaciones en entornos virtuales sin afectar el sistema real, lo que reduce el coste y el tiempo necesario para el desarrollo y pruebas de nuevos sistemas de producción. En el ámbito de la manufacturación, se pueden utilizar gemelos digitales para simular la creación de nuevos productos, probar procesos de producción y optimizar los flujos de trabajo para mejorar la eficiencia y reducir costes antes de su implementación en el mundo real.

Por último, los gemelos digitales permiten la recopilación y análisis de grandes cantidades de datos, lo que puede ser utilizado para la toma de decisiones y la optimización de los procesos en tiempo real.

1.2.4. Inconvenientes

Los gemelos digitales son una herramienta valiosa para la ingeniería y el diseño de sistemas complejos, pero también presentan desafíos significativos. Uno de los principales problemas es la necesidad de contar con una gran cantidad de datos precisos en tiempo real para crear y mantener los gemelos digitales. Si los datos de entrada no son lo suficientemente concretos, la calidad de las predicciones y simulaciones puede verse afectada, generando resultados que difieren de la realidad.

Además, la disponibilidad de datos puede ser un desafío. En algunos casos, los sistemas antiguos no están diseñados para recopilar datos en tiempo real, lo que limita la cantidad y calidad de los datos disponibles. En estos casos, es posible instalar sensores y otras tecnologías para recopilar la información necesaria en tiempo real y mejorar la calidad de los datos.

La seguridad también es un aspecto crucial en la creación y mantenimiento de gemelos digitales. Al almacenar copias virtuales de sistemas y entornos reales, los gemelos digitales se convierten en objetivos atractivos para los cibercriminales. Por lo tanto, se deben implementar medidas de seguridad sólidas para proteger los datos y garantizar la integridad del sistema.

En resumen, aunque los gemelos digitales ofrecen beneficios significativos, su creación y mantenimiento implican costos, complejidad y desafíos de seguridad. Sin embargo, con la inversión adecuada y un enfoque en la calidad de los datos, los gemelos digitales pueden ser una herramienta valiosa en el ámbito de la ingeniería y el diseño de sistemas complejos.

1.3. Ciudades inteligentes (Smart Cities)

Las ciudades inteligentes, o *Smart Cities*, utilizan tecnología y datos para mejorar la calidad de vida de sus habitantes, optimizar la gestión de los recursos y servicios públicos, y reducir el impacto ambiental. Los gemelos digitales son una herramienta clave en la construcción de ciudades inteligentes, ya que permiten la simulación y análisis de situaciones complejas en un entorno virtual antes de implementar cambios en la ciudad real.

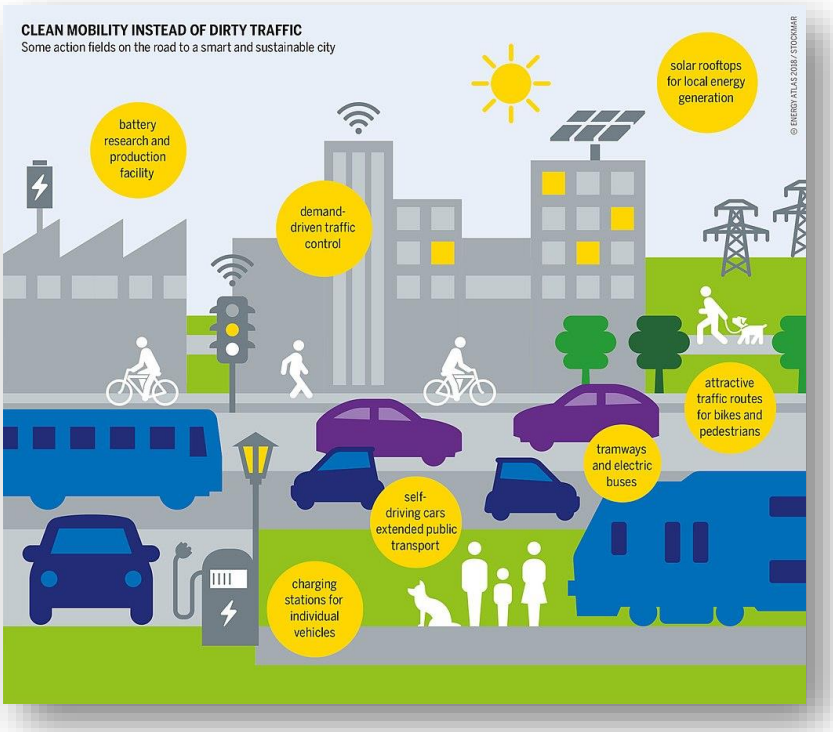


Figura 2. By Bartz/Stockmar.

Con gemelos digitales, las ciudades pueden modelar y simular diferentes situaciones, como el impacto de la construcción de nuevos edificios o infraestructuras en el tráfico o en la calidad del aire, o cómo una nueva norma de transporte público que afectaría a los tiempos de viaje y a la movilidad de los ciudadanos. Esto se debe a que los gemelos digitales ayudan en la toma de decisiones al proporcionar datos precisos y en tiempo real sobre la ciudad y sus habitantes.

La aplicación de gemelos digitales en ciudades inteligentes también puede mejorar la eficiencia de la gestión de los servicios públicos, como el suministro de agua y energía, la recogida de residuos y el mantenimiento de infraestructuras. Al tener una visión más clara y precisa de la ciudad, las autoridades pueden identificar problemas y solucionarlos más rápidamente, y también planificar y optimizar los servicios de manera más efectiva. Otro ejemplo de uso en las ciudades podría ser la prevención de catástrofes naturales. Un gemelo digital podría simular un terremoto y ayudar a los equipos de emergencia a prepararse mejor para una situación real. Además, también se podría utilizar para diseñar edificios más resistentes a los terremotos y planificar mejor la evacuación de las

zonas afectadas. De la misma manera, se podría utilizar para simular y planificar la respuesta a otros desastres naturales, como huracanes, inundaciones o incendios forestales.

1.4. Caso de estudio

El sistema actual de autobuses urbanos en Málaga presenta deficiencias en la precisión de los horarios y la capacidad de adaptación ante imprevistos, como averías en los autobuses. Esto se debe al uso de un enfoque tradicional en la planificación de horarios, que no permite ajustarlos en tiempo real en función de las circunstancias del momento. El objetivo principal de este trabajo es desarrollar un gemelo digital del sistema de autobuses urbanos de Málaga, con el propósito de mejorar el servicio y superar las limitaciones del sistema tradicional. Se ha optado por esta solución para abordar los problemas existentes en la capacidad de adaptación del servicio, buscando así lograr una mayor eficiencia en el transporte público.

Es importante destacar que el gemelo digital del sistema de autobuses urbanos de Málaga no tiene influencia directa sobre el sistema real de autobuses, sino que se presenta como una herramienta de apoyo para mejorar la toma de decisiones. Este gemelo digital permitirá simular diversos escenarios y evaluar el impacto de posibles mejoras en el sistema de transporte público. Una de las funcionalidades clave es la capacidad de recalcular automáticamente los horarios en función de las circunstancias actuales de la línea, lo que proporciona una mayor adaptabilidad y eficiencia en la gestión del servicio.

Para el desarrollo del gemelo digital, se seleccionó la línea 11 como línea de entrenamiento del sistema, a pesar de que en un gemelo digital completo se requeriría entrenar con todas las líneas. Esta elección se debe a varios factores. En primer lugar, el proceso de entrenamiento del gemelo es el mismo para cada línea, por lo que realizarlo con todas las líneas habría llevado al mismo diseño, pero con una mayor complejidad en el desarrollo y gestión del mismo. En segundo lugar, este trabajo se ha diseñado para ser desarrollado utilizando plataformas y bases de datos cloud gratuitas, y el procesamiento adicional de las demás líneas habría obstaculizado su desarrollo completo. Sin embargo, estos dos motivos no suponen un problema, ya que una vez que

se haya desarrollado una base sólida del gemelo digital, simplemente se pueden añadir más líneas a la lista de entrenamiento para comenzar a procesarlas. La elección de la línea 11 como línea de entrenamiento tiene como motivo principal la variedad de zonas que visita a lo largo de su recorrido. Al cruzar gran parte de Málaga, esta línea nos genera una amplia gama de situaciones y escenarios diferentes, lo que resulta muy beneficioso para el estudio y análisis del sistema. Al entrenar el gemelo digital con esta línea, podemos obtener una comprensión más completa y representativa de los desafíos y patrones que pueden surgir en el transporte público urbano.

Aunque el cliente web desarrollado permite visualizar el funcionamiento del gemelo digital e interactuar con él, es importante destacar que el objetivo principal de esta aplicación no son los usuarios en sí, sino la tecnología sobre el gemelo digital. Se ha investigado sobre diferentes áreas de los gemelos digitales y cómo se podría implementar uno sobre el sistema de autobuses de Málaga. A continuación, se presenta el diagrama de componentes de la aplicación completa, que muestra el funcionamiento externo y la interacción de los distintos componentes cloud. Las dos aplicaciones desarrolladas, Servicio Bus y Cliente Bus, describen el backend y el frontend del gemelo digital respectivamente.

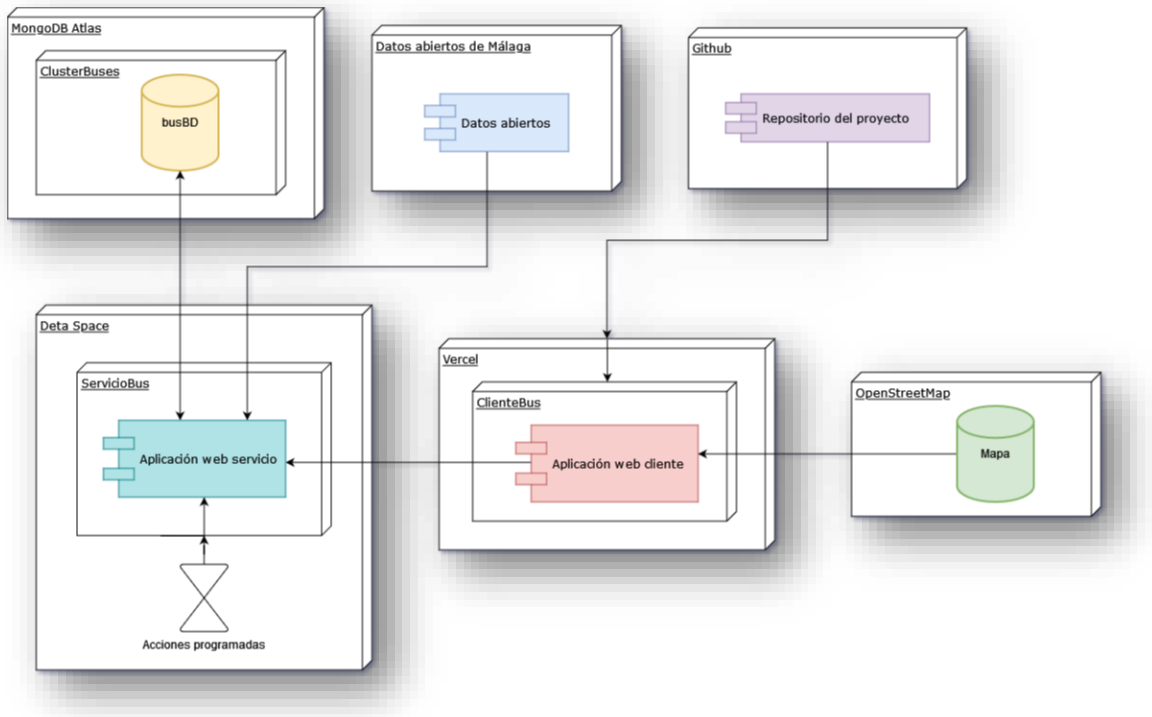


Figura 3. Diagrama de componentes.

En el caso específico del gemelo digital de los autobuses urbanos, se han seguido las siguientes fases de funcionamiento:

1. Recopilación de datos: Se recopilan los datos de los autobuses cada minuto y se almacenan en una base de datos para crear un histórico de datos. Este conjunto de datos es fundamental para desarrollar la fórmula o modelo del gemelo digital.
2. Validación: Se realizan pruebas para validar que las simulaciones generadas por el gemelo digital sean coherentes y se correspondan con el comportamiento del sistema real. Además, se incluye un proceso de autoajuste en el que el sistema se adapta y recalibra en función de los nuevos datos que se van recopilando.
3. Predicciones a largo plazo: Una vez que el modelo ha sido validado, se procede a generar predicciones a largo plazo. Estas predicciones permiten evaluar el sistema en diferentes escenarios y situaciones. Por ejemplo, se pueden utilizar para optimizar las frecuencias y horarios de los autobuses, así como para analizar cómo afectarían los posibles inconvenientes o incidentes a una ruta normal.

En conclusión, el desarrollo del gemelo digital del sistema de autobuses urbanos de Málaga representa un avance significativo en la gestión y mejora del transporte público. Mediante la simulación, predicción y evaluación de diferentes escenarios, este trabajo proporciona una valiosa herramienta para la toma de decisiones y la optimización del servicio.

1.5. Estructura de la memoria

La memoria del TFG está estructurada en varios capítulos, Introducción, Tecnologías, Desarrollo, Validación y Conclusiones. Se da por explicado el primer capítulo.

El segundo capítulo trata las tecnologías, este se encarga de describir de forma general la tecnología correspondiente, además se introducen algunas utilidades que se han utilizado de ella.

En el tercer capítulo de la memoria, se abordan los aspectos relacionados con el desarrollo de la aplicación. Se comienza describiendo la metodología de trabajo seguida, seguido de la exposición de las diferentes fases del desarrollo, con sus respectivos

objetivos y problemas abordados. Se detallan las pruebas generales realizadas para garantizar el correcto funcionamiento de la aplicación, así como la estructura interna y el despliegue de los componentes. Además, se proporciona una pequeña guía de uso para el usuario y se incluye un diario de desarrollo que registra algunos de los datos de los apartados más técnicos del trabajo.

El cuarto capítulo demuestra la Validación, después de una pequeña introducción se exponen tres Pruebas realizadas a través de tablas de tiempo, además de dos análisis que se han desarrollado utilizando los datos generados por el gemelo. Al final podemos encontrar una conclusión de los resultados obtenidos.

El último capítulo es el Conclusión, detalla las conclusiones del trabajo, así como las mejoras que se podrían realizar. Al final, nos encontramos con la bibliografía.

2. Tecnologías

2.1. Introducción

En la sección de Tecnologías, se abordarán las herramientas utilizadas en el trabajo, así como algunas alternativas. Además, se describirá la infraestructura en la nube utilizada, los lenguajes de programación empleados, los formatos de almacenamiento de datos, la base de datos utilizada, los entornos de desarrollo, las plataformas de control de versiones y las aplicaciones utilizadas para la planificación del trabajo. Además se mencionarán algunas librerías y frameworks que hayan sido de especial importancia para el trabajo.

2.2. Eclipse Ditto

Eclipse Ditto es una herramienta principal *open source* que puede utilizarse para construir gemelos digitales. Es un *framework* que integra diversas tecnologías y protocolos de conexión, como HTTP y MQTT, permitiendo la conexión de diferentes dispositivos, como Arduinos, teléfonos móviles y servidores externos, entre otros.

Además de estos protocolos, Ditto también es compatible con *WebSockets*, lo que aumenta la versatilidad del sistema al permitir la conexión con otros dispositivos *IoT* y sistemas externos como *backends* para el procesamiento de datos. También cuenta con una base de datos interna, MongoDB, que se utiliza para guardar las "cosas" (*things*) que representan los diferentes elementos, como sensores, puertas, e incluso dispositivos más complejos como personas o vehículos.

Inicialmente, este trabajo estaba pensado para utilizar Eclipse Ditto, aprovechando la base de datos interna para guardar los datos y aprovechando uno de los ejemplos de trabajos que tiene publicado Eclipse sobre Ditto en un repositorio público, se podría alojar todo el sistema en Microsoft Azure. Sin embargo, como este trabajo no requiere la compatibilidad con dispositivos MQTT, los datos utilizados provienen de la web de datos abiertos de Málaga y el procesamiento de datos tendría que ser igualmente externo, la integración con Ditto resulta innecesaria.

En resumen, alojar todo el sistema en una máquina de Azure es más costoso en tiempo y dinero ya que habría que utilizar máquinas en la nube, es más simple y eficiente utilizar alternativas gratuitas para las diferentes aplicaciones que componen el gemelo digital.

2.3. MongoDB

En el TFG se ha decidido utilizar MongoDB como sistema gestor de base de datos. MongoDB utiliza documentos BSON, una representación binaria de los JSON con mejor rendimiento durante el filtrado de elementos, lo que lo convierte en una base de datos no relacional con un gran rendimiento en comparación con las bases de datos relacionales tradicionales.

Además, MongoDB presenta una gran flexibilidad y puede integrarse fácilmente con casi cualquier lenguaje de programación mediante sus librerías oficiales, que soportan tecnologías como la familia C, Java, Node, Python, etc. También existen librerías desarrolladas y mantenidas por la comunidad. Esto permite que exista una forma fácil de conectarse a una base de datos Mongo independientemente de la tecnología se utilice.

MongoDB Atlas es una plataforma gratuita que ofrece almacenamiento y procesamiento de datos en la nube, lo que significa que no se requiere una infraestructura local para acceder a los datos. MongoDB Atlas también proporciona una interfaz gráfica de usuario y una amplia gama de herramientas de gestión de bases de datos que facilitan el tratamiento de estos.

La seguridad en MongoDB es una característica destacable. Al crear un clúster con la base de datos en la nube, se genera una dirección de acceso al servicio de MongoDB. Sin embargo, es importante tener en cuenta que esta dirección de acceso debe incluir

un usuario con permisos y su correspondiente contraseña para poder acceder a los datos almacenados. Esta medida de autenticación añade una capa adicional de seguridad para proteger tus datos.

Además, desde la web oficial de MongoDB, se da la posibilidad de configurar la lista blanca de direcciones IP. Esto significa que solo las direcciones IP especificadas en la lista blanca, junto con sus respectivos puertos, podrán establecer una conexión con la base de datos. Esta configuración permite restringir el acceso aumentando así la seguridad y control sobre quién puede acceder a los datos almacenados en MongoDB.

En resumen, MongoDB Atlas es una solución de base de datos escalable, flexible y fácil de usar, que permite a los desarrolladores trabajar con una gran variedad de datos y lenguajes de programación, sin necesidad de preocuparse por la infraestructura de la base de datos. MongoDB Atlas se adapta perfectamente a aplicaciones complejas como los gemelos digitales que requieren filtrados rápidos de grandes cantidades de datos, y ofrece herramientas de gestión y análisis que hacen que trabajar con bases de datos en la nube sea una tarea sencilla.

2.4. Lenguajes de programación

En esta sección, se describirán las dos tecnologías principales utilizadas en las dos aplicaciones web que conforman la aplicación del gemelo digital.

2.4.5. Python

Python es un lenguaje de programación moderno y ampliamente utilizado que se destaca por su comunidad activa y colaborativa de desarrolladores. Esta comunidad ofrece un apoyo constante y una amplia gama de recursos para resolver cualquier problema que pueda surgir durante el desarrollo, incluso en casos muy específicos. Otra ventaja es que se trata de un lenguaje interpretado, lo que significa que no es necesario compilar el código antes de ejecutarlo simplificando el desarrollo y depuración del código. La sintaxis de Python es fácil de entender, ya que está basada en la sintaxis inglesa permitiendo escribir sentencias utilizando palabras como is y not, lo que lo hace ideal para programadores principiantes.

Python ofrece una gran cantidad de librerías y frameworks que simplifican el desarrollo de aplicaciones en diversos dominios. Por ejemplo, en este trabajo se han utilizado librerías matemáticas como Math, Scipy y Statistics para realizar cálculos estadísticos y para simplificar la manipulación de datos. Además, Python tiene una excelente capacidad para trabajar con archivos CSV, lo cual ha sido aprovechado en este trabajo para acceder a la ubicación de los autobuses que alimentan el Gemelo.

FastAPI se destaca como uno de los frameworks más importantes para el desarrollo de aplicaciones web en Python, especialmente cuando se trata de construir API REST. Sus ventajas más destacadas incluyen su facilidad de uso, velocidad y robustez. Además, al estar basado en OpenAPI, FastAPI inicia automáticamente una herramienta llamada Swagger durante la ejecución. Esto elimina la necesidad de utilizar aplicaciones externas como Postman para la depuración de la API. La interfaz gráfica de Swagger proporciona una forma intuitiva y sencilla para que los desarrolladores prueben su aplicación web sin necesidad de preocuparse por un frontend adicional o utilizar aplicaciones de terceros.

En resumen, Python y FastAPI ofrecen flexibilidad y robustez para desarrollar el *backend* del TFG, permitiendo construirlo como aplicación web dividida en diferentes ficheros como un único servicio. Además, las librerías científicas han permitido simplificar el procesamiento de los datos.

2.4.6. ReactJS

ReactJS es una librería de código abierto de JavaScript desarrollada para construir interfaces de usuario en aplicaciones web, en el caso de este trabajo se ha utilizado para desarrollar la vista del usuario. Una de las principales ventajas de ReactJS es su sintaxis declarativa y fácil de leer, que permite la creación de componentes modulares y la reutilización de código.

ReactJS utiliza JSX, una sintaxis que combina HTML y JavaScript para crear y controlar componentes dinámicos. Al renderizarse todo del lado del cliente, la aplicación se actualiza de manera eficiente al cambiar solo las partes necesarias de la interfaz. En este trabajo, se ha aprovechado al máximo estas ventajas para crear componentes dinámicos que actualizan el mapa, los marcadores y la tabla de tiempos y el cuadro de mandos.

Por otro lado, Node es el entorno de ejecución de JavaScript utilizado en este trabajo, Node y ReactJS se integran para ampliar la cantidad de módulos y permite construir aplicaciones altamente escalables.

Además, ReactJS se integra bien con otras tecnologías, lo que lo hace ideal para trabajos complejos donde se unifiquen diversas cosas. Existen muchas bibliotecas y *frameworks* adicionales que se pueden utilizar con ReactJS.

Material UI es una librería de componentes que incluye una amplia variedad de estilos y personalizaciones pensadas para que los desarrolladores puedan crear interfaces de usuario atractivas y coherentes adaptadas a las necesidades del desarrollador. En este trabajo casi todo el apartado del estilo ha sido realizado con Material UI y su profunda cantidad de opciones para iconos, botones, *tooltips*, etc.

También hay que destacar la librería de Leaflet para React. Esta tecnología, combinada con OpenStreetMap, permite construir mapas dinámicos que muestran la información actualizada en tiempo real de todo lo que se necesite, desde marcadores y formas hasta recorridos completos utilizando coordenadas.

2.5. Plataformas cloud

Hay varias plataformas en la nube que permiten alojar proyectos desarrollados en diferentes lenguajes de programación. No obstante, para el *backend*, las opciones gratuitas o para estudiantes son escasas. Heroku es una alternativa popular que ofrece un plan para estudiantes, y aunque tiene un límite de uso en su versión gratuita y posiblemente no se fuese a alcanzar en este trabajo, se decidió evitar a favor de una plataforma completamente gratuita. En este caso la opción preferida para alojar el backend ha sido Deta Space, una plataforma que permite el uso de diversas tecnologías en un solo proyecto. Sin embargo, aún se encuentra en una fase beta y por lo tanto tiene graves carencias, como la capacidad de borrar proyectos que aún no está implementada, o que utiliza una aplicación de consola que tiene numerosos problemas, esto ha producido inconvenientes durante todo el TFG a la hora de desplegar la aplicación.

Por otro lado, entre las opciones del frontend, se consideró inicialmente Netlify, pero se acabó utilizando Vercel, que funciona de manera similar para el desarrollador y también

tiene un límite de uso en el plan gratuito. Cabe destacar que Vercel también permite desplegar proyectos Python, pero contribuirían a alcanzar el límite de uso del plan gratuito.

2.5.7. Deta Space

La plataforma Deta Space ha sido creada especialmente para aficionados y estudiantes que buscan una forma de alojar sus aplicaciones sin incurrir en costos elevados. Entre sus características destacadas se encuentra su gratuidad y la posibilidad de descargar aplicaciones desarrolladas por otros usuarios. Aunque al inicio del desarrollo, Deta Space había experimentado cambios significativos que ampliaron las capacidades del trabajo, actualmente se encuentra en una fase de desarrollo constante, presentando algunas limitaciones y carencias en cuanto a funcionalidades básicas, como la capacidad de eliminar proyectos.

Figura 4. Icono del Servicio Bus en Deta Space.

Inicialmente, la idea principal era alojar todo el trabajo en Deta, combinando la aplicación de procesamiento de datos en Python y la aplicación cliente web en ReactJS. Sin embargo, surgieron dificultades al agregar la segunda aplicación, ya que se alcanzó el límite de memoria permitido. Además, resultó más conveniente y funcional utilizar la base de datos MongoDB Atlas en lugar de la base de datos relacional integrada en Deta.

Por esta razón, se decidió utilizar diferentes plataformas para la vista del usuario y el procesamiento de datos del Gemelo.

En la actualidad, el funcionamiento de Deta se basa en la creación de un proyecto Space localmente, que se vincula con un proyecto en la nube, y dentro de este se crean Micros, que son los distintos servicios tecnológicos que componen las aplicaciones. La configuración de la aplicación se realiza a través de un archivo de texto YAML llamado *Spacefile*, donde se especifican propiedades como el nombre y el icono de la aplicación, así como los Micros.

En este trabajo en particular, se ha creado un único Micro llamado "recoger-datos", que utiliza el motor de Python 3.9, cinco variables de entorno y tres acciones programadas. Estas acciones son llamadas automáticas a la API, por lo que se pueden ejecutar manualmente. Las variables de entorno contienen la URL y las credenciales del servidor de MongoDB, el origen de la aplicación (Deta Space) y formas de controlar la depuración. Por otro lado, las acciones automáticas desempeñan un papel importante en el Gemelo Digital, una de ellas realiza una llamada cada minuto al servicio de datos abiertos para recopilar información sobre los autobuses, mientras que las otras dos se encargan del procesamiento de datos para realizar simulaciones y construir tablas de predicciones. Las acciones y variables de entorno se pueden modificar mediante una interfaz gráfica una vez está instalada la aplicación en la nube.

En resumen, Deta Space es una plataforma cloud gratuita que ofrece la posibilidad de desarrollar aplicaciones web complejas. Aunque es importante tener en cuenta que se encuentra en una etapa de desarrollo temprana, su comunidad es muy activa por lo que resulta útil para resolver problemas y obtener ayuda. Con estas características, Deta Space se presenta como una excelente opción para los desarrolladores que deseen crear aplicaciones en la nube sin preocuparse por límites de uso de procesamiento, como es el caso en este trabajo.

2.5.8. Vercel

Vercel es una plataforma de alojamiento en la nube que se integra a la perfección con GitHub y que es ampliamente utilizada por desarrolladores de React. Una de las características más destacadas de Vercel es su capacidad para automatizar

completamente los despliegues de la aplicación. Esto se logra gracias a su integración con GitHub, lo que permite que Vercel se encargue de todo el proceso de despliegue cada vez que se realiza un push al repositorio de GitHub.

Para direccionar las consultas del cliente sobre autobuses, paradas, formas, tablas de tiempos y líneas disponibles, se utiliza una variable de entorno que contiene la dirección URL de la última versión de la API en Deta, esto permite cambiar fácilmente la dirección cuando se despliega una nueva versión del Gemelo.

El comando de Node.js "CI= npm run build" se utiliza para crear una versión de producción de la aplicación cliente. La parte "CI=" fuerza la ejecución a pesar de que Node se queje, a través de una advertencia, de utilizar librerías desactualizadas.

Asimismo, Vercel ofrece una URL personalizada única para cada despliegue, lo que permite visitar cualquier versión desplegada de la aplicación sin necesidad de borrarlas. Además, Vercel ofrece una URL única para el que sea el último despliegue, lo que facilita el acceso a la última versión de la aplicación en todo momento.

2.6. Entornos de desarrollo (IDEs)

Durante la fase inicial del desarrollo del software del trabajo, se creó una versión preliminar de la aplicación encargada de recopilar periódicamente los datos de los autobuses y almacenarlos en la base de datos. Esta aplicación se ejecutó en un entorno Linux, y para realizar modificaciones en la misma se utilizó el editor de texto Nano a través de una conexión segura SSH.

Posteriormente, en el transcurso del desarrollo, se empleó un entorno único de desarrollo para Python y JavaScript, Visual Studio Code. En el caso de JavaScript, se utilizó una extensión dedicada a ReactJS, junto con otras extensiones más genéricas para optimizar el flujo de trabajo. Al igual que en JavaScript, con Python se utilizaron las extensiones adecuadas para facilitar la programación y depuración del código en este lenguaje.

Además, se recurrió al uso de Notepad++ como herramienta adicional para editar archivos en formato JSON y YAML. Este editor de texto proporcionó una interfaz sencilla para editar los archivos más pequeños, comparar resultados de ejecuciones de la API y manejar datos de MongoDB.

En resumen, se optó por hacer uso de los editores de código y texto más versátiles y potentes disponibles, con el fin de evitar la necesidad de utilizar múltiples entornos con sus diferentes atajos, opciones y extensiones para garantizar un desarrollo fluido.

2.7. Control de versiones

En este TFG, se ha utilizado la tecnología de control de versiones de GitHub para gestionar los cambios realizados en el código. GitHub ha permitido mantener un registro ordenado de los cambios, facilitando organización y asegurando un control eficiente de versiones que permitía ver los cambios realizados cuando se producía un nuevo error.
Además, se ha integrado GitHub con Vercel, lo que ha permitido simplificar el proceso de despliegue del *frontend*. Esta integración ha facilitado la implementación automática de los cambios realizados en el repositorio de GitHub en la plataforma de alojamiento y despliegue de aplicaciones web de Vercel.

2.8. Planificación

En cuanto a las herramientas de planificación, se han utilizado principalmente el correo electrónico, Google Docs y Google Calendar. Estas herramientas han sido utilizadas para sincronizar eventos, programar reuniones y organizar el flujo de trabajo del trabajo.
La utilización de un documento de texto en la nube como Google Docs ha traído ventajas significativas con respecto a otras herramientas, facilitando la gestión de la información del trabajo. Mediante el uso de listas de cambios, argumentaciones, capturas de pantalla y explicaciones detalladas, se ha logrado documentar los errores y problemas encontrados de una manera eficiente. Además, la utilización de esta herramienta ha permitido tener en un mismo documento una lista de posibles mejoras futuras y diferentes versiones de los mismos bocetos. Estas funcionalidades proporcionadas por un editor de texto en la nube han mejorado la comprensión de la información de una forma que fuera de un editor de texto no hubiese sido tan fácil.

2.9. Visual Paradigm

Para la elaboración de los diagramas de componentes y de paquetes en la memoria del TFG, se empleó la herramienta Visual Paradigm. Esta herramienta proporciona una

interfaz gráfica intuitiva que permite diseñar y representar visualmente la estructura y las relaciones entre los distintos componentes y paquetes de la aplicación.

3. Desarrollo

3.1. Metodología de trabajo

Para llevar a cabo el desarrollo del TFG se ha implementado una metodología ágil que ha permitido reunirse con regularidad, aproximadamente cada dos semanas, para evaluar el progreso del trabajo y afrontar cualquier impedimento surgido. En estas reuniones, el tutor y el realizador del trabajo discuten el trabajo completado desde la última reunión, así como los desafíos encontrados. Asimismo, se planifica el trabajo que se llevará a cabo en las próximas dos semanas, en preparación para la próxima reunión. La metodología ágil ha sido fundamental para mantener un seguimiento y una comunicación efectiva entre el tutor y el realizador del trabajo durante todo el proceso de desarrollo.

3.2. Fases de desarrollo

El TFG se ha desarrollado en varias fases, algunas de las cuales han sido continuas a lo largo del curso del trabajo. La fase de planificación ha sido fundamental para mantener el orden y el desarrollo del trabajo actualizado. La fase de análisis ha sido otra fase continua, donde se ha realizado la investigación inicial sobre tecnologías y el funcionamiento de los gemelos digitales. La investigación ha ido de la mano con el desarrollo de las diferentes entidades del trabajo. Se ha investigado cómo construir cada sección, realizar predicciones, estructurar componentes y el flujo de datos, así como validar el gemelo digital. Además, se ha llevado a cabo la investigación sobre los datos abiertos proporcionados por el portal de datos abiertos del ayuntamiento de Málaga, identificando aquellos datos relevantes para el trabajo y cómo utilizarlos.

La fase de investigación comenzó en enero, con la elaboración del anteproyecto y una idea general del funcionamiento del TFG. Sin embargo, fue a finales de febrero cuando se aceleró el desarrollo del trabajo y se confirmaron las tecnologías a utilizar, siendo las más esenciales Python, ReactJS y MongoDB. En este momento se inició la recolección de datos, pero el problema es que la aplicación en la nube no estaba aún preparada y había que recoger todos los datos posibles. La solución fue utilizar una Raspberry Pi en local con una aplicación muy escueta que accedía a la web de datos abiertos una vez por minuto, esta solución tenía un problema grave y es que no siempre estaba conectada a la red por lo que no todos los datos de estas fechas fueron recaudados.

En marzo, se comenzó la segunda fase del TFG, que implicó el diseño del gemelo, el cliente y el servicio, así como las interacciones básicas entre el backend y el frontend. Durante esta etapa, se trabajó en la estructuración de la base de datos, realizando modificaciones continuas hasta validar el modelo. Hacia finales de marzo, el trabajo estaba bien encaminado en la tercera fase, el desarrollo, aunque las fases anteriores aún no se habían completado. Durante el desarrollo, se integraron los datos abiertos con el backend y el frontend, lo que permitió mostrar los últimos autobuses en el mapa y actualizarlos automáticamente con nuevos datos. Sin embargo, surgieron problemas de integración debido a las diferentes franjas horarias de los componentes en la nube. Además, se enfrentaron dificultades en la automatización de la recolección de datos para la actualización del mapa, lo que resultó en un rendimiento deficiente y ralentización del navegador debido a la alta cantidad de solicitudes por segundo. Paralelamente a la integración del mapa y los datos, se llevó a cabo una investigación para encontrar iconos adecuados que no infringieran derechos de autor para su uso en toda la aplicación. También se exploró la posibilidad de utilizar la plataforma Cloud Heroku como alojamiento para el backend, debido a los problemas de estabilidad experimentados con Deta Space.

En abril, se logró establecer el recorrido de la línea 11 en el mapa, lo que permitió corregir las posiciones geográficas de los autobuses y calcular las distancias necesarias. Se construyó una tabla de tiempos preliminar para estimar el tiempo de llegada de cada autobús a cada parada del recorrido desde su posición actual.

En cuanto a la herramienta Eclipse Ditto, se avanzó en su investigación y se realizaron pruebas de integración de datos con el contenedor virtual de Docker de Ditto, logrando almacenar datos de forma automática. Sin embargo, se determinó que esta herramienta era demasiado compleja para las necesidades del trabajo, por lo que se decidió abandonar su uso.

En términos de desarrollo, se encontró que la vista del cliente carecía de automatización ya que fue desactivada por sus problemas y el código era complicado de navegar y utilizar. Por tanto, se dedicaron las siguientes iteraciones de la aplicación a refactorizar, limpiar y reconstruir partes del código. Tanto la tabla de tiempos como el mapa se reconstruyeron desde cero para permitir su actualización dinámica cada minuto con los últimos datos de la base de datos.

Estas nuevas versiones fueron denominadas internamente en el trabajo como “Mapa dinámico” y “Mapa de tiempos”. La tabla de tiempos tuvo ese nombre internamente debido a que dejó de ser una tabla de todos los tiempos para pasar a ser un mapa desde el punto actual de autobús seleccionado hasta las próximas paradas, sin olvidar por donde ya había pasado. Esto ayudó a la automatización futura que permitió devolver este “mapa” a la base de datos para que fuese actualizado con nueva información. Aunque este cambio produjo serios problemas de rendimiento en la API nube, cada vez que se actualizaba la tabla de tiempos se realizaba una llamada a la base de datos por cada parada que hubiese que analizar, en esta iteración de la aplicación la actualización de la tabla tardaba alrededor de cinco segundos en calcularse lo que producía una experiencia pésima además de bloquear el Servicio bus durante demasiado tiempo. Las labores de división y organización de código continuaron su curso hasta principios de mayo, en ellas se crearon desde cero todas las funciones involucradas en la creación de la tabla de tiempos para tener que realizar solo tres llamadas a MongoDB en lugar de alrededor de alrededor de 70.

En mayo, aunque el TFG estaba en funcionamiento, se identificaron varios errores que se fueron corrigiendo hasta los últimos días antes de la entrega. Además, se realizó una optimización del código y se llevó a cabo una importante actualización visual de la página para adaptarla a navegadores de ordenadores y dispositivos móviles. Se utilizaron las tecnologías mencionadas anteriormente para garantizar la adaptabilidad

de la página a futuros cambios. En esta etapa, se consideró que la investigación estaba completada, lo que permitió finalizar el trabajo. Los diseños estaban finalizados, con excepción de algunos cambios imperceptibles.

En la elaboración de la memoria del trabajo, se utilizaron herramientas adicionales para ayudar a la redacción y la visualización de información. Se emplearon varias aplicaciones temporales de Python construidas con librerías de visualización de datos para generar gráficas basadas en datos seleccionados, lo cual permitió ilustrar algunos aspectos del trabajo de una manera muy clara y visual.

Además, se utilizó la herramienta Visual Paradigm para crear diagramas que representaran la estructura e interacción de la aplicación desarrollada, no se han desarrollado diagramas de gran complejidad porque se ha considerado que el funcionamiento que describen estos diagramas ya creados es el mejor para comprender la aplicación y sus interacciones de forma general. Estas herramientas adicionales contribuyeron a mejorar la calidad y presentación de la memoria.

3.2.1. Pruebas

Durante el desarrollo del TFG se llevaron a cabo diversas pruebas en la vista, la API en la nube y la base de datos. Se realizaron pruebas exhaustivas en la aplicación web Cliente ya que esta utiliza todas las tecnologías del trabajo a la vez, se realizan seleccionando un autobús y permitiendo que se actualice periódicamente. Estas pruebas abarcaban diferentes aspectos, como la correcta actualización de la interfaz, la visualización de los datos actualizados, la correcta representación de los marcadores en el mapa y la validez de la tabla de tiempos. Además, se verificaba la veracidad de los datos mostrados mediante el uso de la API y la base de datos, realizando las mismas consultas y comprobando los resultados. Estas pruebas permitieron encontrar los errores de integración de los datos además de los problemas únicos de la vista.

Durante las pruebas realizadas, se identificaron principalmente problemas relacionados con la actualización de datos y el mapa. En cuanto a la actualización de datos, surgieron desafíos debido a la necesidad de ajustar la tabla de tiempos, el autobús seleccionado y las tarjetas correspondientes cuando se seleccionaba o deseleccionaba un autobús. Sin embargo, la actualización de datos no ocurría de forma inmediata, lo que podía generar

inconsistencias. Por ejemplo, al actualizar el autobús seleccionado, la tabla de tiempos podría generarse utilizando los datos del autobús anteriormente seleccionado, lo que generaba resultados incorrectos. Para solucionar este problema, se implementó una solución que aseguraba que la llamada a la API para obtener una nueva tabla de tiempos fuera ejecutada de manera sincronizada con la función de selección de autobús. De esta manera, se evitaban casi todas las inconsistencias garantizando una solución sólida a la mayoría de los problemas de la vista.

En el caso de problemas relacionados con el mapa, específicamente en la gestión de los marcadores de los autobuses, a veces, los marcadores de autobuses antiguos no se eliminaban después de una actualización, lo que resultaba en una acumulación de marcadores sin datos en el mapa. Este problema fue intermitente y difícil de diagnosticar. Además, se implementó una funcionalidad en la que se mostraba en rojo el camino que el autobús debía recorrer hasta la próxima parada al seleccionarlo. Sin embargo, esta característica presentaba dificultades en su actualización y generaba una gran carga computacional debido a la necesidad de iterar a través de cientos de segmentos del mapa en cada cambio.

Para abordar estos problemas, se decidió utilizar capas de marcadores en el mapa. Esto permitió eliminar capas antiguas y crear nuevas capas en función de los elementos que se actualicen. Esta solución se implementó específicamente en la capa de marcadores de los autobuses, ya que era la única que requería actualizaciones automáticas después de haber acabado con la funcionalidad de marcar el camino hasta la próxima parada. Esto logró resolver los problemas relacionados con la acumulación de marcadores en el mapa y optimizó el rendimiento general.

Durante el desarrollo del trabajo, se realizaron pruebas exhaustivas para validar el funcionamiento de la aplicación en diferentes navegadores y plataformas. Se evaluaron la compatibilidad y el rendimiento en navegadores como Chrome, Edge y Firefox, en sistemas operativos Windows, Linux y Android. Estas pruebas permitieron detectar posibles problemas de compatibilidad y garantizar una experiencia de usuario coherente y de alta calidad en todos los entornos probados. La mayoría de las pruebas se centraron en la apariencia visual de la aplicación en dispositivos móviles y pantallas grandes. Gracias a la interoperabilidad de las tecnologías utilizadas y a los avances en

los navegadores web modernos, no se encontraron problemas significativos en las últimas versiones de los mismos. Esto aseguró que la aplicación se adaptara correctamente a diferentes tamaños de pantalla y ofreciera una experiencia consistente para los usuarios en distintos dispositivos y sistemas operativos.

3.3. Funcionamiento

Para comprender el funcionamiento, despliegue y validación del sistema primero hay que entender su estructura interna.

3.3.2. Estructura interna

La aplicación se divide en dos aplicaciones web, Servicio Bus (*backend*) y Cliente Bus (*frontend*), junto con una base de datos y la utilización de una web de datos abiertos.

En el *backend*, el paquete Servicio Bus contiene diferentes clases y paquetes. El primer tipo de paquetes es de *endpoints*, que agrupa las clases relacionadas con las funcionalidades específicas de la aplicación web. Cada clase en este paquete tiene sus propias funciones únicas. Los paquetes Buses, Datos, BD y Líneas y Formas se relacionan con la base de datos para insertar o extraer elementos almacenados. El paquete Buses también se conecta a la web de datos abiertos. Todos estos paquetes, excepto BD, interactúan con el Cliente Bus. La segunda clase de paquetes en el *backend* contiene funciones auxiliares que son accedidas por el resto de las clases de la aplicación. Además de las funciones auxiliares, hay una clase controladora de variables de entorno.

En el *frontend*, el Cliente Bus se divide en diferentes secciones: Página Principal, Mapa, Funciones auxiliares y Tabla de tiempos. La Página Principal engloba todas las secciones e incluye la selección de línea, el mapa, el cuadro de mandos y, si está disponible, la tabla de tiempos. Las funciones auxiliares son utilizadas para renderizar el cuadro de mandos, controlar los relojes, calcular el autobús de predicción y recopilar datos para su distribución en los demás componentes. El Mapa utiliza funciones auxiliares para representar los marcadores y líneas que simbolizan los autobuses, trayectos, paradas y usuarios. Por último, la Tabla de tiempos también utiliza funciones externas para introducir iconos y transformar los tiempos en un formato más legible para el usuario.

A continuación, se incluye el diagrama que represente la estructura de paquetes de la aplicación.

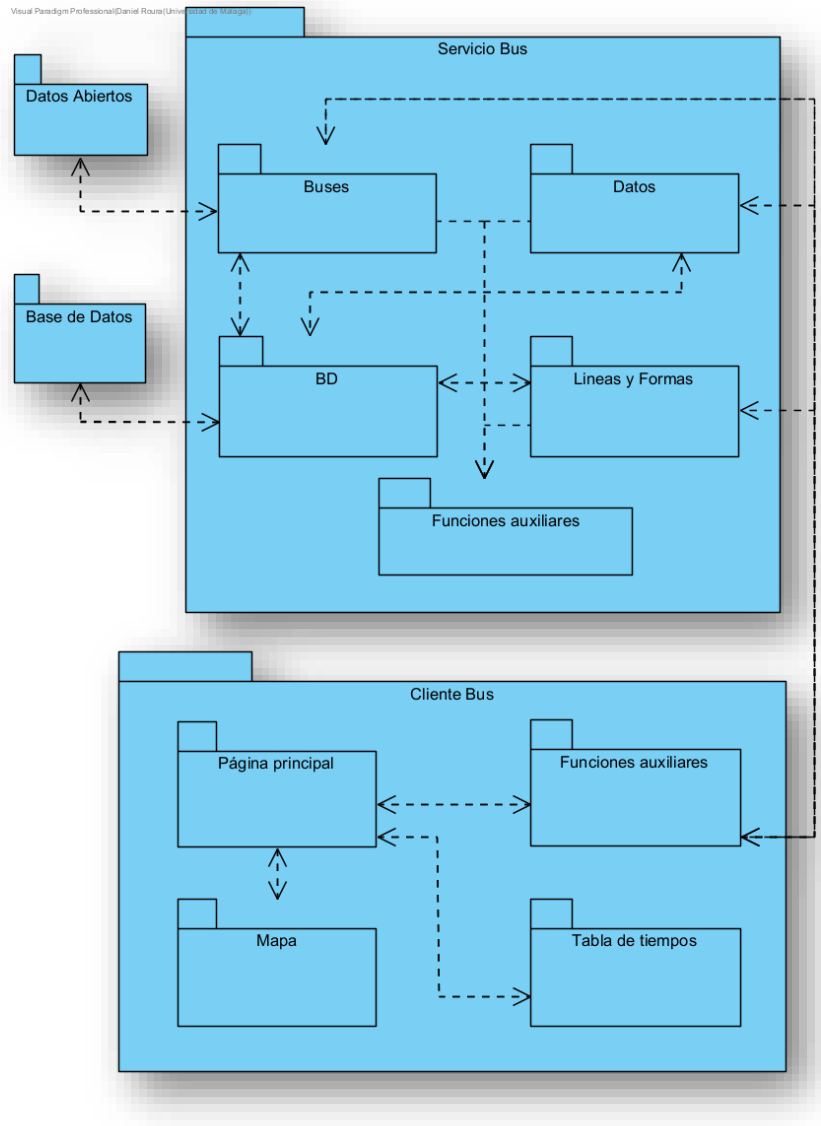


Figura 5. Diagrama de paquetes.

La Figura 6 es un diagrama de clases que representa la estructura y organización del backend Servicio Bus. Este diagrama proporciona una visión detallada de las clases, funciones y paquetes involucrados en el sistema, mostrando cómo se relacionan entre sí y cómo interactúan para realizar diversas operaciones. La representación visual de las clases y sus relaciones facilita la comprensión de la arquitectura del backend y ayuda a visualizar la estructura general del sistema.

Al enfocarse en las clases y sus relaciones, el diagrama proporciona una visión clara de la organización de los componentes del backend y cómo se interconectan para lograr su funcionalidad.

En el backend del Servicio Bus, destacamos varias características importantes. En primer lugar, la mayoría de las asociaciones con líneas y formas están diseñadas para permitir la selección de una línea y forma de línea específicas utilizando su código correspondiente, esto es utilizado en muchas funciones para poder ubicar elementos en el recorrido, o calcular distancias y tiempos. En el backend del Servicio Bus, cabe destacar que la mayoría de las funciones hacen uso del archivo auxiliar.py, el cual contiene una variedad de funciones genéricas de utilidad. Entre estas funciones se incluyen cálculos de tiempos, distancias, transformaciones de cadenas de texto a fechas, búsqueda de elementos en listas según ciertos parámetros, y acceso a URLs externas, entre otras. Estas funciones genéricas proporcionan una base sólida y facilitan la implementación de diversas operaciones en el sistema.

Además, los constructores.py desempeñan un papel fundamental al aligerar el código de algunas funciones que almacenan elementos en la base de datos. El archivo calendario se enfoca en detectar si una fecha específica es un día festivo o no. El archivo env.py incluye variables de entorno y algunas variables codificadas internamente que no deben ser modificadas. Por otro lado, extras.py permite acceder a las variables de depuración y mostrar el index.html del backend, que se utiliza con el mismo propósito que las variables mencionadas anteriormente. Como se ha mencionado en el diagrama de paquetes previo, la mayoría de las clases que interactúan con la API están vinculadas a la base de datos a través de su controlador, representado por la clase intermedia BD.py. Las clases buses, datos, líneas y formas son responsables de almacenar los datos en MongoDB, y, además, actúan como recursos para la API. Estas clases procesan toda la información relacionada con sus respectivos nombres, desempeñando un papel crucial en la gestión de los datos en el sistema.

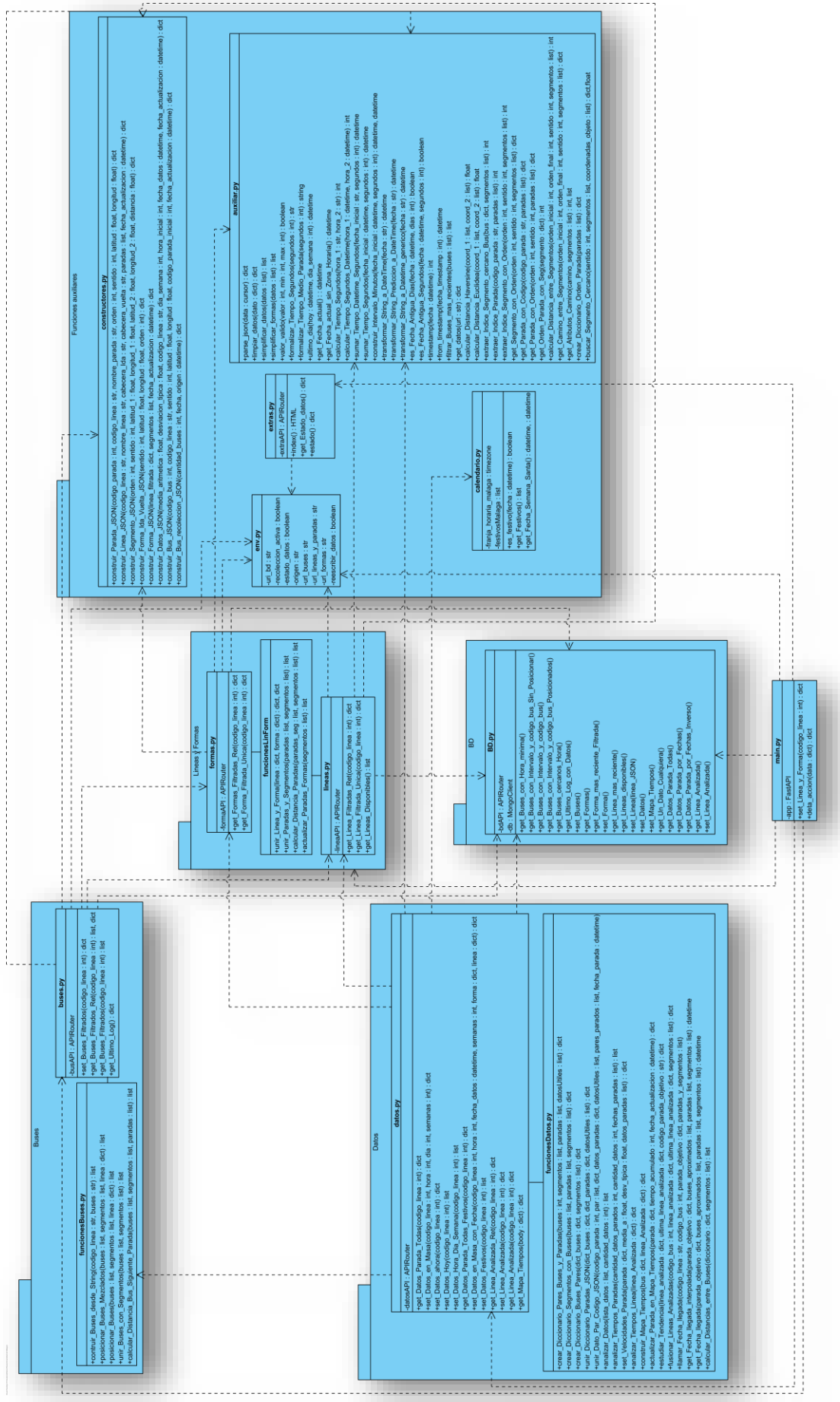


Figura 6. Diagrama de clases de Servicio Bus.

Los componentes mencionados anteriormente conforman el núcleo central de este TFG, desempeñando un papel fundamental en la recopilación, procesamiento y simulación de predicciones. Además, asumen la responsabilidad de gestionar el flujo de información de la base de datos, garantizando la correcta utilización y gestión de todos los datos en el sistema en su totalidad.

3.3.3. Despliegue

En esta sección se va a redactar cómo se ha sido desplegado el trabajo entero, es decir la aplicación Servicio, o (*back*), de recogida y procesamiento de datos, y la aplicación Cliente (*front*), que es la aplicación que permite visualizar una parte del procesamiento.

3.3.3.1. Despliegue del Servicio

La aplicación Servicio es un proyecto de Deta Space en el que se encuentra una sola aplicación de Python. Para crear un proyecto de Deta hay que instalar la aplicación en línea de comandos y ejecutar en la propia consola el comando *space new*, luego te piden que introduzcas un nombre para el proyecto y te crean una carpeta con el nombre de la aplicación y un archivo de configuración YAML llamado *Spacefile*. Además, en la consola se muestra un enlace a la versión en construcción de tu aplicación, aún vacía, un enlace a la documentación del *Spacefile* y el comando *space push* que permite lanzar tu aplicación local a la nube. Para construir el Micro (servicio dentro de un proyecto) de la aplicación se ha utilizado de base un ejemplo de un repositorio público de Deta. A partir de ahí se tiene que actualizar el *Spacefile* indicando qué motor utiliza el Micro creado, si es principal, y las rutas de acceso públicas, en este caso son todas las rutas públicas, dentro de /api. Además, se han creado variables de entorno y acciones automáticas que enlazan con la página web de la plataforma *cloud*, esto está descrito en la sección de Deta.

Dentro de la carpeta recoger-datos se encuentra la aplicación web de FastAPI, el funcionamiento de su esqueleto se describe en la sección de Python. Para convertir el proyecto local en una aplicación en la nube, se utiliza la consola de Deta, que permite validar, construir y subir el código utilizando la configuración especificada en el archivo *Spacefile*. A medida que el proyecto se va construyendo en la nube, la consola

proporciona información sobre el estado del proceso. Una vez que el proyecto está construido en la nube, se puede probar y publicar una nueva versión de producción que se instala en la cuenta de Deta y comienza a ejecutarse en la nube. La URL de esta versión instalada se utiliza en la aplicación cliente de ReactJS para realizar las peticiones.

Es importante no utilizar la versión del *builder* de Deta porque cambia a cada push que se haga, hay que publicar la versión e instalarla para que no desaparezca con un push.

Para replicar el proceso habría que crear un proyecto nuevo desde la consola y meter en la carpeta de este nuevo proyecto el *Spacefile* y recoger-datos. Si hacemos un push de la carpeta del proyecto, la aplicación de consola debería impedirlo por no tener instalado el token de seguridad de la cuenta donde está subido oficialmente.

Para acceder al Servicio Bus, se puede utilizar el enlace proporcionado en el pie de página de la vista. Este enlace, denominado "Acceso a la API".

3.3.3.2. Despliegue del Cliente

Para desplegar la aplicación Cliente en la nube, se utiliza la plataforma de hosting Vercel. Gracias a Vercel para desplegar una aplicación después de la primera vez solo hay que hacer *git push*, pero en la primera hay que seleccionar varias opciones.

Cuando se despliega por primera vez un proyecto hay que conectar el repositorio Git, si el proyecto está en una subcarpeta debemos indicar cuál es para que Vercel pueda desplegarla. Durante el proceso de despliegue inicial debemos introducir una variable de entorno que hace relación a la API del *backend*, además hay que sobrescribir el comando de ejecución por el ya mencionado CI= npm run build debido a las librerías que utilizamos. Si no incluimos este cambio en el comando, no se terminará de ejecutar su despliegue.

Una vez desplegado, Vercel nos ofrece una URL única para la versión desplegada, y una URL permanente que hace referencia siempre a la última versión desplegada. La versión única está pensada para hacer pruebas y la permanente es la versión de producción. El enlace al despliegue de producción de la vista de este trabajo es la siguiente:

https://cliente-bus-danielroura.vercel.app

Dentro de un proyecto en Vercel tenemos un apartado donde se encuentran todas las versiones desplegadas de la aplicación con sus enlaces únicos. Desde este apartado

podemos desplegar versiones concretas de la aplicación por si hemos realizado algún cambio en la configuración, como, por ejemplo, un cambio en una variable de entorno. Una vez tenemos esto ya podemos acceder a cualquier versión de la aplicación creada, ver errores de despliegue y realizar pruebas sobre el proyecto.

3.3.4. Funcionamiento general

Para comprender el funcionamiento general de la aplicación, es importante comenzar por el backend, que representa la mayor parte del trabajo. En el núcleo de la aplicación, se encuentran tres acciones programadas que se ejecutan en intervalos de tiempo predefinidos. La primera acción se activa cada minuto y realiza una solicitud a una URL específica para obtener la posición de todos los autobuses de la línea 11.

Antes de explorar el funcionamiento mediante acciones automáticas, es necesario mencionar que, además de los autobuses, se ejecuta una función de forma manual que realiza una descarga de las líneas y los puntos de coordenadas que conforman dicha línea. Al ejecutar la función de la API para actualizar las líneas y formas de recorrido, se descargan ambos documentos desde una fuente de recursos abiertos y se procesan. Es importante destacar que su procesamiento se realiza simultáneamente debido a que ambos elementos tienen atributos que hacen referencia al otro. Durante el procesado, se construyen segmentos que recorren todo el trayecto utilizando las coordenadas, y se almacenan las distancias correspondientes para poder asignar distancias entre paradas a cada parada de la línea. Por último, se asigna a cada parada un segmento y a cada segmento su próxima parada. Este enfoque permite que el procesamiento de los datos sea más eficiente desde el punto de vista computacional.

La primera acción recopila las ubicaciones de los autobuses, las cuales no se guardan directamente en MongoDB. En cambio, se someten a un procesamiento previo que utiliza la línea a la que pertenecen y su forma. Al combinar esta información, se obtienen las posiciones de los vehículos dentro del recorrido. Además, se ajustan las coordenadas para que se encuentren dentro del trayecto, ya que los datos del GPS son aproximados y, en ocasiones, permiten que un autobús aparezca dentro de un edificio. Este ajuste de coordenadas también proporciona un error entre la posición GPS original y la corregida,

que se almacena para futuros análisis. Además, se agrega información sobre la próxima parada de cada autobús y la distancia para llegar a ella.

La segunda acción se ejecuta cada hora y agrupa los datos en paquetes diferenciados por la fecha, los códigos de línea y parada, la hora inicial y el día de la semana. Estos grupos se obtienen al recopilar las ubicaciones de los autobuses cada hora, desde el minuto 0 hasta el 59, dentro de una hora específica. Además, ya que los datos pueden variar según el día, se utiliza el día de la semana como diferenciador. Por ejemplo, para las 10 de la mañana de un martes, se recopilan los datos desde las 10:00 hasta las 10:59, y se guarda una lista de los recorridos realizados en el tramo entre la parada anterior y la parada objetivo. También se registran los tiempos que los autobuses han estado detenidos en la parada objetivo, aunque esta información puede no ser totalmente fiable debido a que solo se obtiene la ubicación de los autobuses aproximadamente cada 60 segundos, por lo que estas “fotos” no te muestran el autobús en cada parada, sino que se las salta.

La última acción también se ejecuta cada hora, esta función coge los grupos de datos relacionados con la hora y día a la que ocurre y calcula entre cada parada: las medias de tiempo, medias de velocidad, desviaciones típicas y si es festivo. Con esto construye una línea analizada de un día y hora específicos para que luego se puedan calcular los tiempos reales.

Para terminar, hay que construir las predicciones, estás son accionadas por los usuarios al seleccionar un autobús en el mapa del Cliente, y son una fusión entre una línea analizada y una instancia de un autobús a una hora concreta. De aquí sale una tabla de tiempos que empieza por la próxima parada a la que va a llegar el autobús, con un límite en el número de paradas futuras calculadas. Cada parada tiene atributos calculados de datos procesados, como la hora de llegada que ha simulado el gemelo, el tiempo acumulado para llegar a esa parada desde donde se encuentra el autobús, el tiempo entre la parada anterior a la actual y la actual, la tendencia de la diferencia de la predicción pasa saber si el autobús se está adelantando o atrasando, las velocidades del tramo anterior a la parada y algunos atributos más que pueden ayudar a calcular la predicción de llegada o a realizar un estudio posterior.

Cada minuto esta tabla vuelve al Servicio Bus y se actualiza con más paradas y nuevas predicciones sobre las paradas ya simuladas, también se analizan las horas de llegada reales a las paradas ya pasadas, estas horas de llegada no son 100% exactas por el problema de una foto cada 60 segundos. Además, la tabla de tiempos tiene una lista de tablas de tiempos anteriores para validar posteriormente si las predicciones eran coherentes con lo que estaba pasando en el sistema de autobuses real. Cuando tenemos al menos 10 elementos en la lista, estas se empiezan a guardar en la base de datos. Esta actualización periódica de cada minuto se realiza obteniendo el segundo en el que se han actualizado los últimos autobuses en la base de datos para asegurar tener siempre los datos al día en la vista del Cliente. Cuando se actualizan los datos se renuevan las posiciones de los autobuses y si entra o sale alguno, además si hay un autobús seleccionado se actualiza también su tabla de tiempos.

3.3.4.1. Funcionalidades adicionales del Servicio

En esta sección se describen características adicionales del Servicio que no están incluidas en el funcionamiento general. Una de ellas es la capacidad de ampliar el número de líneas disponibles. Aunque los estudios se centran actualmente en la línea 11, es posible utilizar el recurso mencionado en el segundo párrafo de la sección de Funcionamiento General para obtener información de más líneas de autobuses. Además de esto, también se ha implementado un recurso adicional que permite obtener los códigos de línea de las líneas disponibles en la base de datos para poder ser seleccionados en el Cliente.

Otra funcionalidad ofrece la opción de habilitar o deshabilitar el modo de pruebas. Cuando se activa este modo, el servicio continúa recopilando datos, pero los guarda en una tabla de pruebas separada en la base de datos. Esto permite probar nuevas versiones de la aplicación sin comprometer el sistema de recogida de datos.

Además, se han implementado recursos para solicitar el análisis o la recolección de datos específicos. Los usuarios pueden enviar solicitudes para analizar ciertos datos o para recopilar información adicional. También se ofrece la posibilidad de visualizar los datos más recientes almacenados en la base de datos, lo que proporciona una visión actualizada de la información disponible.

Otra funcionalidad destacada es la capacidad de acceder a los últimos registros (logs) insertados en MongoDB, que corresponden a la última recopilación de datos de los autobuses. Esto permite realizar un seguimiento de las actualizaciones más recientes y obtener información sobre el estado y el rendimiento del sistema.

Por último, se ha implementado la opción de ejecutar funciones que obtienen datos de fuentes externas sin guardar esos datos. Esta función resulta útil para depurar el sistema y verificar si los datos externos están disponibles y funcionando correctamente.

3.3.4.2. Funcionalidades adicionales del Cliente

El Cliente permite seleccionar la línea que se desea mostrar en el mapa de entre las disponibles en la base de datos, aunque solo la línea 11 cuenta con autobuses. Una característica relacionada con la actualización periódica de datos en el Cliente es la inclusión del autobús de predicción. Al seleccionar un autobús, se crea una representación ficticia en el mapa que muestra el trayecto que el autobús seleccionado debería seguir en tiempo real, hay que considerar que los datos disponibles están desfasados. Es importante destacar que la posición de este autobús de predicción es una aproximación, por lo que, si el autobús original se detiene en un semáforo, experimenta una avería o realiza una parada demasiado larga, el autobús de predicción continuará avanzando en la ruta cada pocos segundos. Cuando se actualizan las posiciones reales de los autobuses, la predicción del autobús se ajusta a su nueva posición actualizada.

La última funcionalidad no mencionada del Cliente es el Cuadro de mandos, que se encuentra entre el mapa y la tabla de tiempos. El Cuadro de mandos muestra información útil sobre el autobús seleccionado y una parada elegida. Si se han seleccionado ambos y la parada se encuentra en la tabla de tiempos, también muestra el tiempo estimado de llegada del autobús seleccionado a esa parada. Esta sección del Cliente proporciona un resumen visual de los datos relevantes para facilitar la monitorización y seguimiento del servicio de autobuses, sobre todo en dispositivos móviles donde manejar una tabla con los tiempos es mucho más complicado.

3.4. Manual de usuario

El cliente web se centra en proporcionar una interfaz intuitiva para que los usuarios puedan interactuar con el gemelo digital. Aunque hay que volver a destacar que esto no es el foco de este TFG si no la ingeniería que hay detrás.

Al acceder a la página, se muestra un mapa con la línea 11 como carga predeterminada. El mapa, basado en OpenStreetMaps, presenta un zoom de la ciudad de Málaga y muestra claramente el recorrido de la línea, las paradas y la última ubicación conocida de los autobuses de la propia línea. En la esquina superior derecha, se encuentra un botón que permite mostrar u ocultar las tres capas disponibles en el mapa, dándole al usuario la opción de personalizar su visualización según sus necesidades.

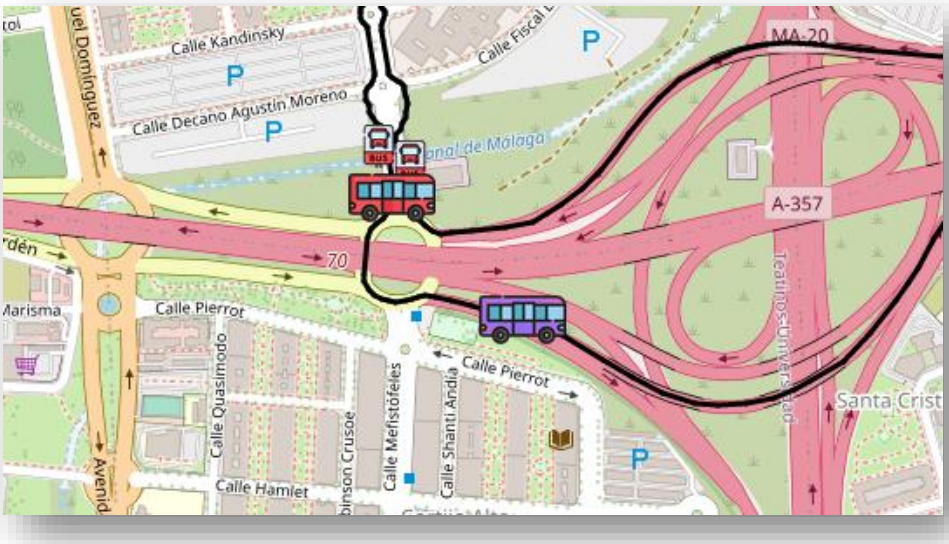
Captura de pantalla 1. Mapa.

Justo debajo del mapa se sitúa el Cuadro de mandos, el cual se encuentra inicialmente vacío. Esta sección ofrece diversas funcionalidades y controles que permiten al usuario

conseguir información del gemelo digita sin tener que recurrir a la Tabla de tiempos. A continuación, se encuentra la sección oculta que acabamos de mencionar, la Tabla de tiempos, la cual ofrece información detallada sobre los tiempos de llegada de los autobuses a cada parada. Al final de la página, se encuentra el pie de página que contiene el nombre del autor del trabajo y un enlace a la API en Deta.

Captura de pantalla 2. Cuadro de mandos vacío.

Al seleccionar un autobús en el mapa, se actualizarán las vistas de la aplicación y se mostrará la sección oculta correspondiente a ese autobús en particular. Además, si seleccionamos la próxima parada con el código 2055, que es la Plaza de Manual Azaña, podremos obtener información específica sobre dicha parada. Por último, se muestra el autobús de predicción en el mapa que se actualiza cada dos segundos y medio con una posición simulada.



Captura de pantalla 3. Autobús de predicción en el mapa.

Cuando un usuario selecciona un autobús en el mapa, las vistas se actualizan para mostrar la información importante. Además, se revela la sección oculta de la Tabla de tiempos. Al seleccionar también la próxima parada con el código 2055, que es la Plaza de Manuel Azaña, se proporcionará información detallada sobre los tiempos de llegada y posibles retrasos en esa parada específica.

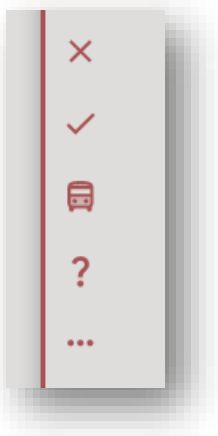
Captura de pantalla 4. Cuadro de mandos completo.

Después de seleccionar un transporte en el mapa, la Tabla de tiempos se calcula y aparece debajo del Cuadro de mandos, garantizando información reciente sobre todas

las paradas analizadas con una actualización por minuto de la propia tabla. Esta permite al usuario conocer los tiempos estimados de llegada a cada parada. Además, a medida que el autobús se acerca a cada parada, se actualiza una columna en la tabla que muestra la hora confirmada de llegada a esa parada en particular.

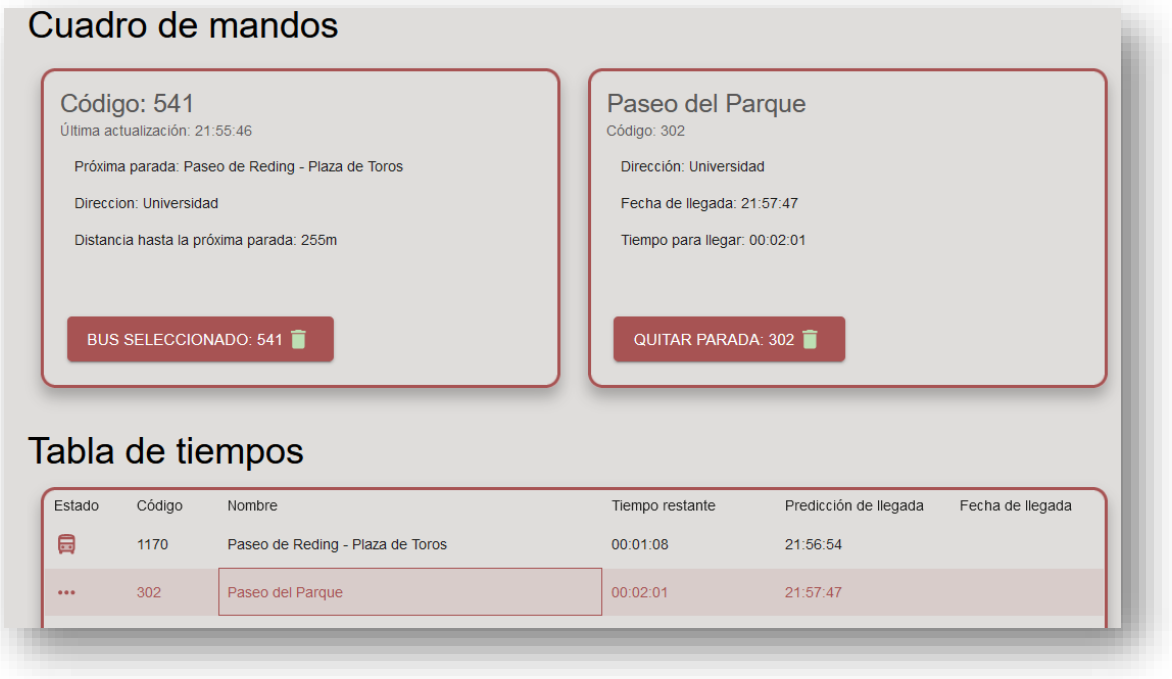
Captura de pantalla 5. Tabla de tiempos.

En la primera columna de la tabla de tiempos se muestra el estado del viaje, y al colocar el cursor sobre dicho icono, se muestra su significado. Por ejemplo, el símbolo del autobús indica el tramo por el que se ha confirmado que el autobús está transitando, mientras que los tres puntos indican que aún no ha llegado a esa parada y la interrogación simboliza que el autobús debería haber llegado, pero no hay confirmación. Los últimos dos iconos, un verificado y una cruz, representan si el autobús ha llegado cerca o lejos de la hora predicha por última vez.



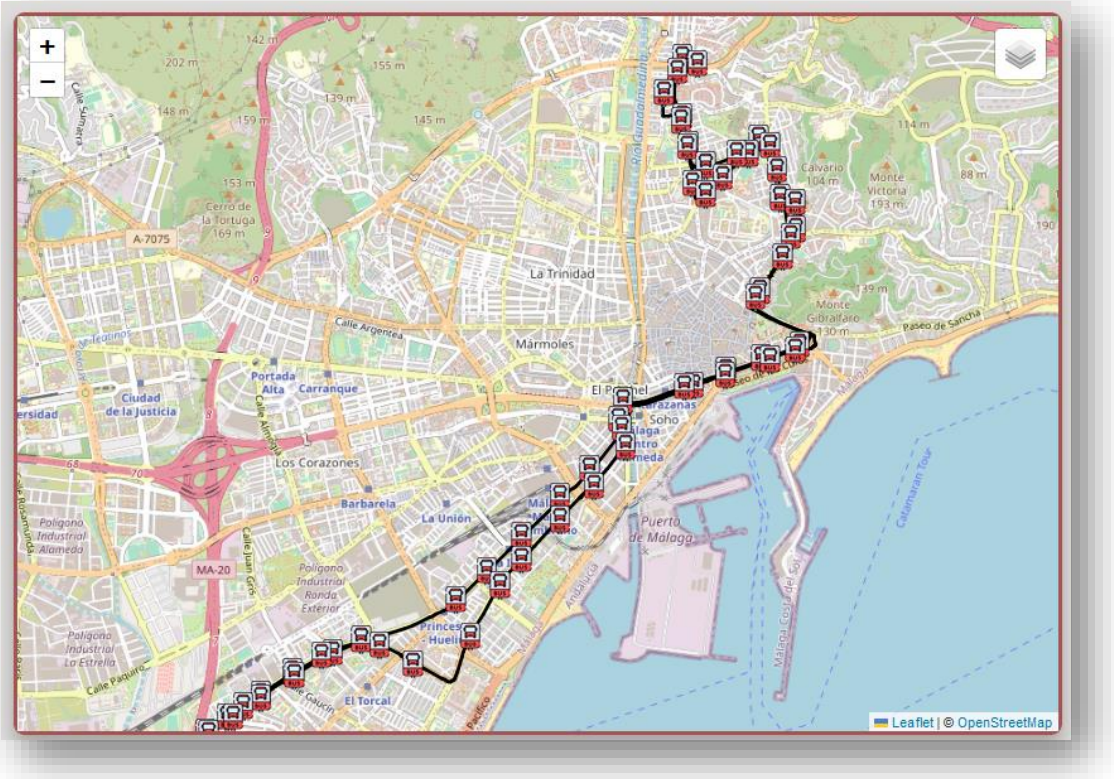
Captura de pantalla 6. Iconos de estado.

La tabla de tiempos también nos permite seleccionar la parada que queramos visionar en el Cuadro de mando.



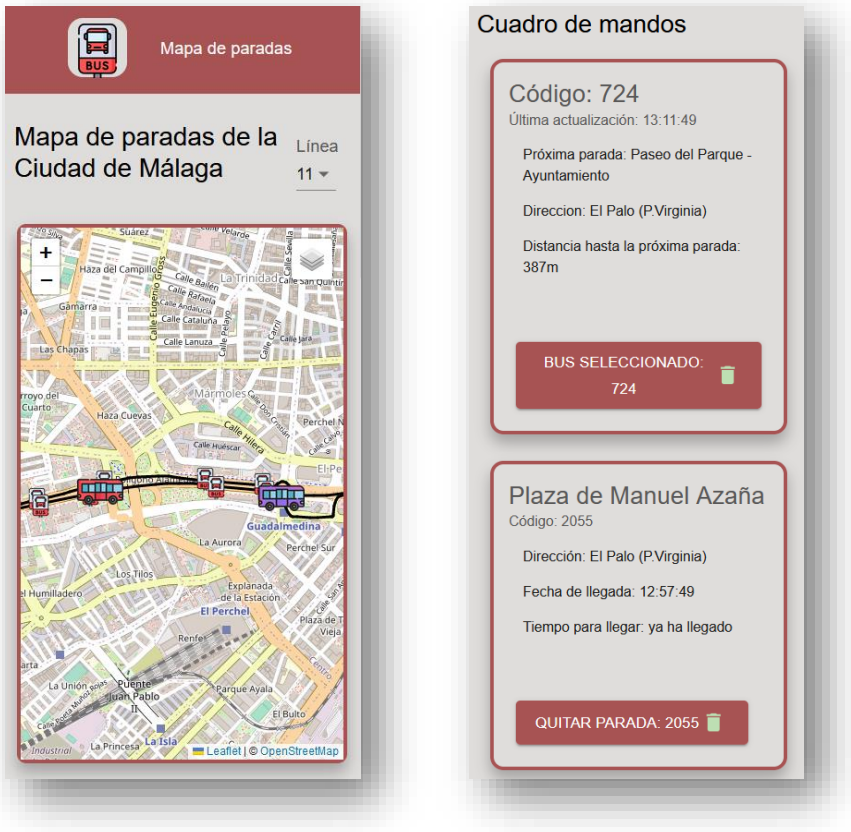
Captura de pantalla 7. Selección en la tabla de tiempos.

Otra funcionalidad de la página es la habilidad de cambiar de línea seleccionada, aunque solo la línea 11 está analizada podemos ver algunas otras líneas dibujadas sobre el mapa.



Captura de pantalla 8. Línea 1.

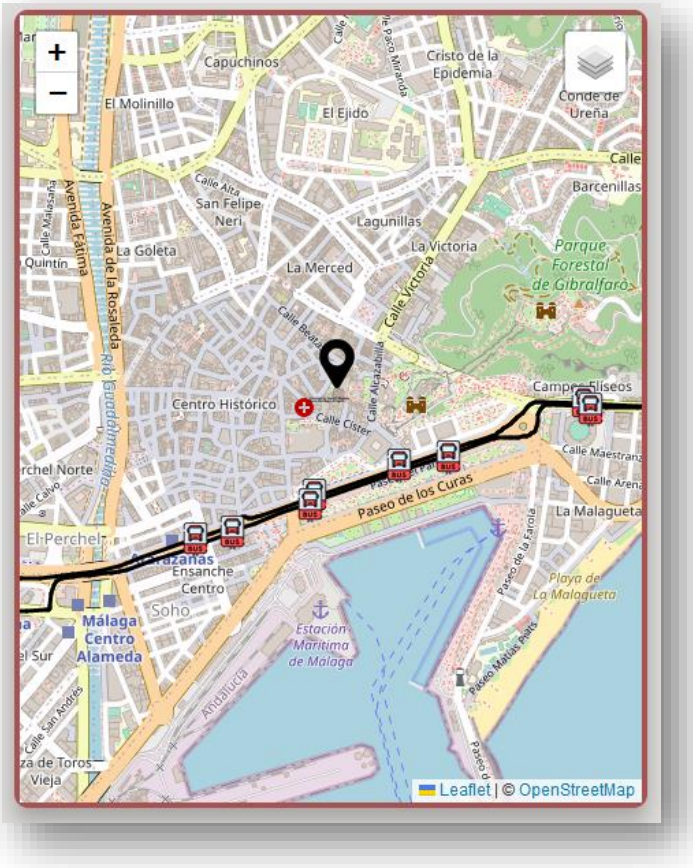
Además de la versión web, también existe una versión móvil de la aplicación que se adapta a la pantalla del dispositivo, brindando una experiencia más cómoda para los usuarios. Sin embargo, se ha identificado que la experiencia con la Tabla de tiempos en la versión móvil puede resultar confusa e incómoda. Para abordar este problema, se diseñó y construyó el Cuadro de Mandos, que proporciona una interfaz más intuitiva y fácil de interactuar, sobre todo, para la versión móvil.



Capturas de pantalla 9 y 10. La vista en móvil.

Captura de pantalla 11. La vista en móvil.

En esta página, se ha implementado la funcionalidad de localización del usuario. Al acceder, el navegador solicita permiso para acceder a la ubicación, y al aceptar, la posición del usuario se actualiza periódicamente en el mapa, siendo marcada con un marcador negro. Es importante seleccionar la opción de "Recordar permisos" cuando se solicite, para que la ubicación se actualice automáticamente en el mapa. Esta característica es bastante conveniente a la hora de utilizar el mapa desde un dispositivo móvil.



Captura de pantalla 12. Ubicación del usuario en tiempo real.

3.5. Diario de desarrollo

Aunque ya se ha descrito un registro general del desarrollo del TFG, en este apartado se pretende abordar algunos aspectos adicionales que podrían resultar de interés para ciertas personas.

En la base de datos MongoDB, se ha registrado un amplio conjunto de datos relacionados con el trabajo. En el momento de entrega de la memoria, el histórico de

ubicaciones de los autobuses cuenta con más de 620.000 elementos, lo que equivale a un peso mayor a 140 megabytes. Estos datos han sido recopilados a través de más de 47.000 accesos a los datos abiertos, cada uno registrado en la base de datos junto con su origen y hora de acceso, siendo la mayoría de ellos realizados desde Deta. Además, otra tabla relevante en la base de datos es la de *datosPool*, que agrupa datos para su posterior análisis. Esta tabla contiene alrededor de 47.000 elementos y tiene un tamaño de aproximadamente 85 megabytes. Cabe destacar que la tabla más pesada y compleja de manejar es la de mapas de tiempos, que equivale a la tabla de tiempos, pero con más cantidad de información. Aunque, al momento de redactar este informe, la cantidad de mapas no supera los 80, el peso de estos datos asciende a más de 150 megabytes. Algunos de estos documentos, en formato JSON han llegado a tener hasta 23.000 líneas de texto, lo que destaca la complejidad y la abundancia de información almacenada en esta tabla a la que le cuesta un tiempo cargar cada vez que se abre en la aplicación de escritorio de MongoDB.

El tamaño total de la colección de MongoDB relacionada con este trabajo actualmente es de 335 Megabytes sobre los 512 permitidos. Considerando el consumo de almacenamiento en el último mes y si se mantiene el mismo patrón de consumo de memoria, es probable que antes de que finalice junio de 2023 se agote el espacio de almacenamiento de la versión gratuita de MongoDB Atlas, pero antes de que ocurra eso se moverán los datos menos necesarios a otro almacenamiento en la nube.

Durante el desarrollo del trabajo, se han realizado numerosos despliegues en las plataformas cloud Deta y Vercel. En Deta, se han desplegado casi 40 versiones distintas en producción, pero se han subido a la nube aún más versiones que no han salido de la etapa de prueba y se han utilizado principalmente para *testing*. En cuanto a Vercel, el proceso de despliegue es más rápido y sencillo, ya que solo se requiere realizar un *git push*. Esto ha permitido realizar más de 90 despliegues de la aplicación en Vercel. Muchas de estas versiones tenían variaciones menores y se enfocaron en corregir pequeños problemas y mejorar detalles específicos. Es importante destacar que la mayoría de los despliegues realizados en Vercel y Deta siguen activos. Sin embargo, se debe tener en cuenta que muchas versiones de la vista no funcionan debido a los cambios en la API. Para evitar conflictos y garantizar un correcto funcionamiento, solo

una versión de Deta se mantiene activa en la recopilación de datos y el tratamiento de elementos de la base de datos.

Estos despliegues frecuentes son una muestra del enfoque iterativo y ágil del desarrollo del trabajo. Permiten una rápida iteración y actualización de la aplicación en respuesta a las necesidades y mejoras identificadas durante el proceso, además nos regala a los administradores una funcionalidad crítica en todo tipo de sistemas actualizables que utilicen la nube, la habilidad de hacer un *rollback* para volver a cualquiera de estas versiones antiguas en caso de que la más reciente tenga problemas. Esta funcionalidad ha sido aprovechada multitud de veces durante el trabajo. Este enfoque garantiza que el trabajo se mantenga en constante evolución y adaptación, mejorando continuamente para brindar una experiencia óptima a los usuarios.

4. Validación

4.1. Introducción

La validación es fundamental en la construcción de gemelos digitales. A través de este proceso, se verifica la calidad, precisión y coherencia del sistema replicado en comparación con el sistema original. Un gemelo digital validado puede realizar simulaciones cuyos resultados son útiles y coherentes con el comportamiento del sistema real.

En el caso específico de la validación del sistema construido, se llevaron a cabo diversas pruebas. Dos de ellas consistió en verificar la precisión de las predicciones de llegada y su concordancia con los datos reales de llegada de los autobuses. Además, se realizaron análisis comparativos del servicio de autobuses en días laborables y festivos, así como en condiciones climáticas secas y lluviosas. Estas pruebas permitieron evaluar el rendimiento y la consistencia del sistema en diferentes escenarios y condiciones, no solo asegurando que los resultados obtenidos fueran confiables y útiles si no demostrando el potencial que tiene una herramienta como esta para analizar y optimizar el sistema real.

4.2. Primera prueba de precisión en predicción de llegada

En esta sección se han realizado diversas pruebas utilizando diferentes autobuses y paradas, concretamente se ha utilizado un trayecto del autobús con código 733 del día 18 de mayo. Es importante destacar que las paradas seleccionadas en estas pruebas se

eligieron manualmente de forma semialeatoria, procurando espaciar las selecciones dejando un número mínimo de 4 o 5 paradas entre cada una. Estas paradas fueron elegidas antes de analizar los resultados para asegurar que esto no afectara el proceso de selección.

Cada minuto, se actualiza la tabla de tiempos, y en cada actualización se recalculan las predicciones para mejorar su precisión. Estas predicciones se muestran en el gráfico como una línea azul en el eje Y, que son modificadas a los 60 segundos de actualización. En rojo podemos observar que la llegada real aproximada a la parada, como ya sabemos, no se puede afirmar a ciencia cierta que la llegada haya sido exactamente a esa hora, pero se sabe que ha sido muy cercana a dicho momento.

El primer gráfico se utilizará como ejemplo para explicar su representación. En dicho gráfico, el eje Y muestra la predicción de llegada a la parada con el código 473, mientras que el eje X representa la secuencia temporal que ocurre al seleccionar un autobús en el Cliente. En él podemos observar que las predicciones fluctúan hacia la hora final de llegada, convergiendo cada vez más cerca de la estimación inicial. Además, se puede apreciar que la primera predicción se realizó aproximadamente media hora antes de la llegada real y falló por apenas cinco minutos.

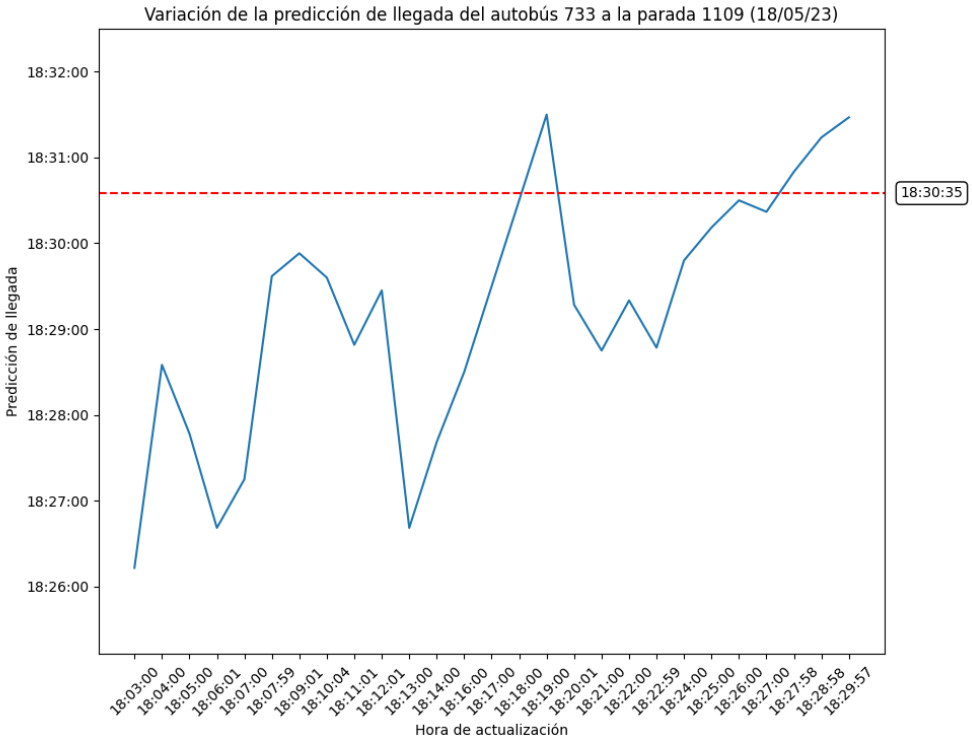


Gráfico 1. Autobús 733 llegando a la parada 1109.

En el segundo gráfico, se puede observar que las predicciones siempre fueron tardías en comparación con la hora real de llegada, hasta aproximadamente cuatro minutos antes, momento en el cual se produjo un cambio significativo en la predicción que permitió una aproximación mucho más precisa.

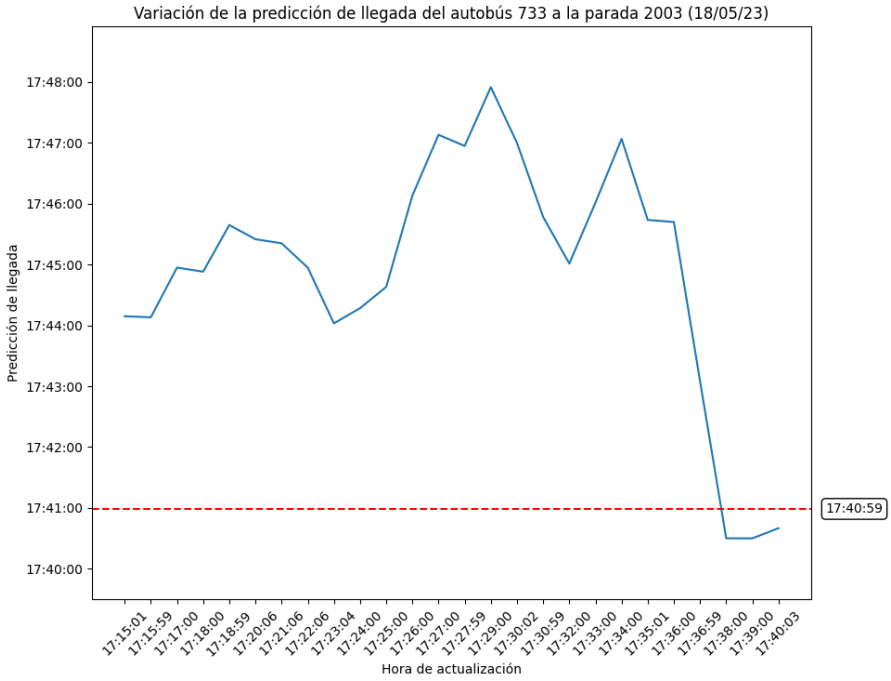


Gráfico 2. Autobús 733 llegando a la parada 2003.

En el tercer gráfico ocurre lo contrario, tenemos una serie de simulaciones que indicaban que se iba a llegar mucho antes de cuando se consiguió alcanzar la parada objetivo.

Gráfico 3. Autobús 733 llegando a la parada 2056.

En este caso particular, Gráfico 4, se han realizado menos muestras de lo habitual, ya que la parada seleccionada estaba cerca del autobús en el momento de la selección inicial. En la gráfica, podemos observar que, aunque al principio la predicción se encuentra relativamente alejada del valor objetivo, a medida que se realizan repeticiones de cálculos, converge gradualmente hacia el valor final. Esto puede deberse a que la parada anterior era la de final de sentido, así que es altamente probable que hubiese un cambio de conductor o un descanso que atrasase el autobús unos minutos más, normalmente alrededor de 10 minutos. Este caso demuestra lo importante que es realizar los cálculos repetidamente para poder observar estos fenómenos y refinar la precisión de la predicción.

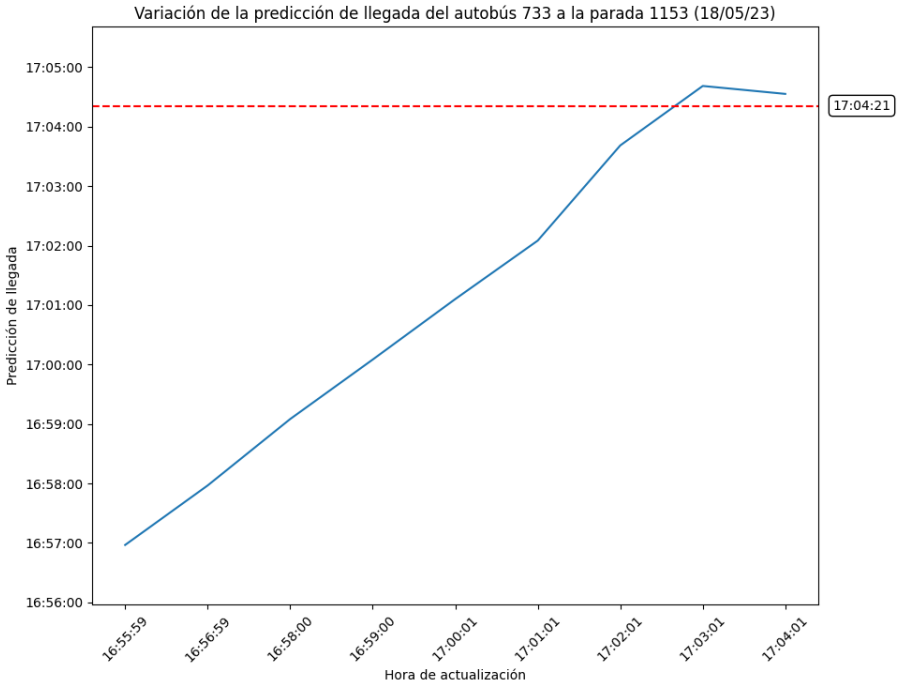
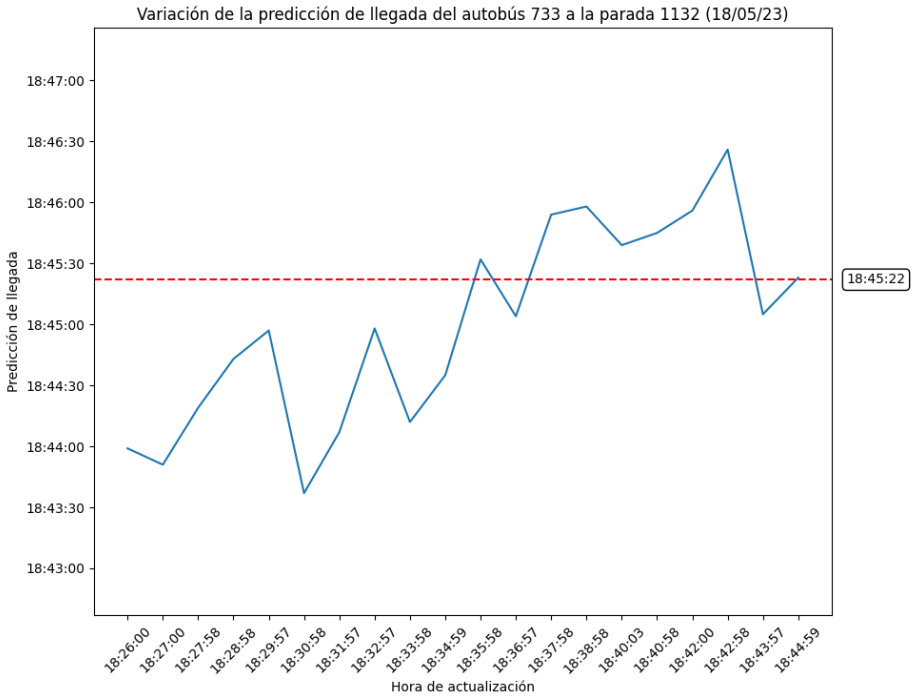


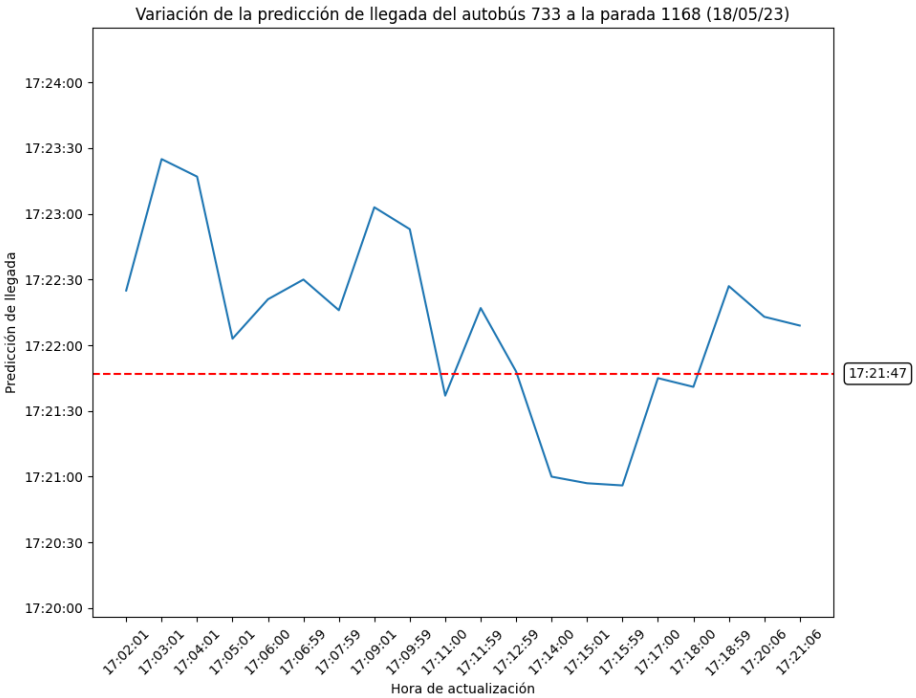
Gráfico 4. Autobús 733 llegando a la parada 1153.

En los últimos cuatro gráficos se puede observar que la predicción de llegada debe estar alrededor de la hora de llegada real, algunas veces se acierta en las primeras predicciones y se mantiene ahí, como en las gráficas 5 y 6, en cambio otras veces la simulación puede desviarse con el tiempo, como en la gráfica 7.



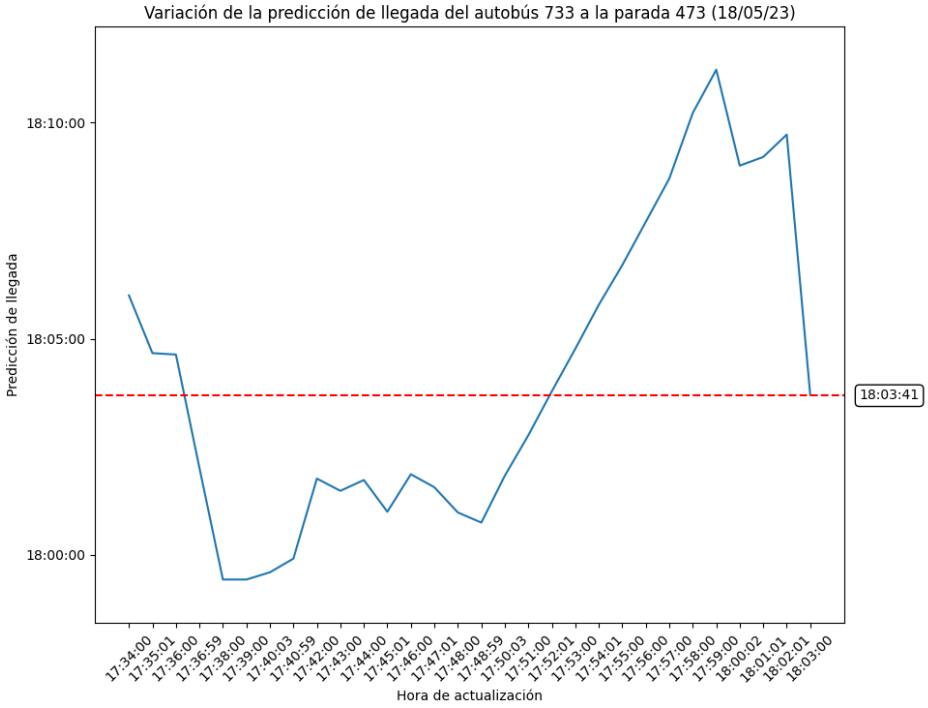
Gráficos 5. Autobús 733 llegando a la parada 1132

El cambio repentino de las gráficas 5 y 6 que ocurren en el último cuarto indica que se produjo una alteración del horario y el conductor del autobús decidió acelerar en el caso 5, y frenar en el 6, para no desviarse demasiado del tiempo previsto.



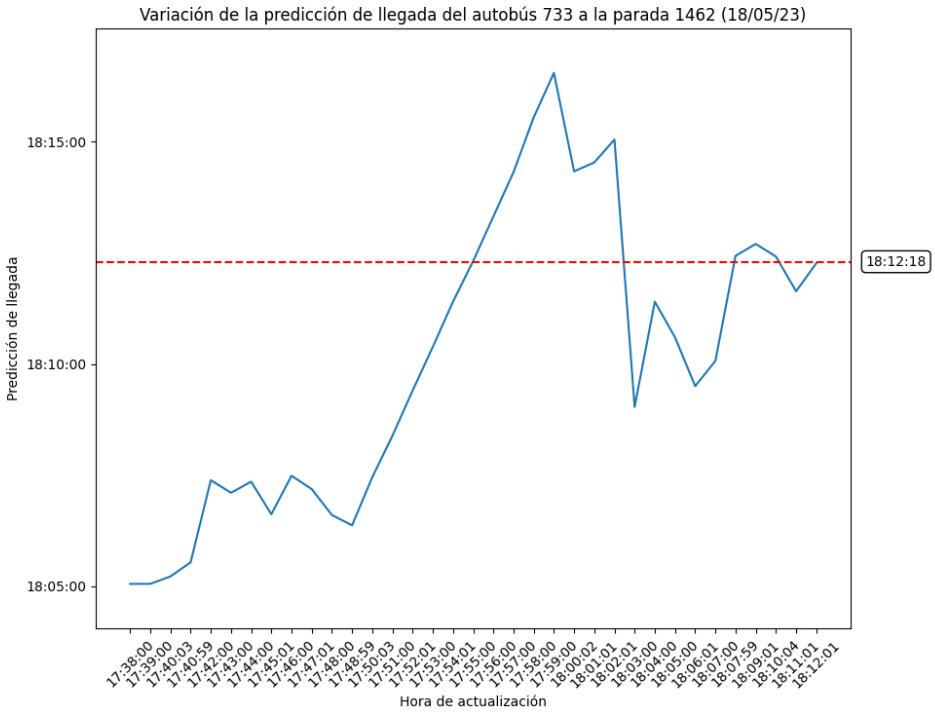
Gráficos 6. Autobús 733 llegando a la parada 1168

Las últimas dos gráficas muestran un patrón similar en su desviación: la gráfica 7 en el último tercio y la gráfica 8 en el tercer cuarto. Si nos fijamos con atención, podemos observar que ambos patrones tienen la misma forma y comienzan a la misma hora. Esta similitud se debe a que solo hay tres paradas entre ellas, a pesar de estar separadas por una distancia de más de 3 kilómetros. La gráfica 8 comienza separada de la llegada final, pero al mitad del dibujo la predicción empieza a fluctuar y acaba convergiendo en el resultado final.



Gráficos 7. Autobús 733 llegando a la parada 473

La anomalía que se presenta en estas gráficas puede explicarse de la misma manera que en el gráfico sobre la parada con el código 2056, que es el punto de inicio de la línea, ya que es donde se produce el retraso. Es común que los autobuses hagan una parada larga en este tipo de ubicaciones, lo cual afecta el tiempo de llegada previsto en las paradas posteriores.



Gráficos 8. Autobús 733 llegando a la parada 1462.

4.3. Segunda prueba de precisión en predicción de llegada

Es importante volver a destacar que una única prueba no es suficiente para validar por completo un sistema de predicciones estadístico como este gemelo digital. Realizar diferentes pruebas con días, autobuses y paradas distintas proporciona una visión más completa y precisa sobre el rendimiento del sistema completo. Esto permite identificar patrones y tendencias además de validar el propio sistema en diferentes circunstancias. Para especificar, esta prueba se ha realizado con autobús y día distinto, y no se ha utilizado ninguna de las paradas analizadas en el apartado anterior para intentar tener mayor variedad en los datos. Al igual que con la prueba anterior, se debe tener en cuenta que en las paradas donde se producen cambios de línea y descansos de los conductores, es probable que se produzcan grandes retrasos en las predicciones, que serán corregidos una vez se atraviese esa parada.

Esta prueba empieza por la parada con código 2055, que se ubica a la entrada de una autovía a Málaga con una gran cantidad de tráfico. Sin embargo, podemos observar que,

a menos de 10 minutos de llegar a la parada, la predicción empieza a fluctuar alrededor de la que sería la llegada real.

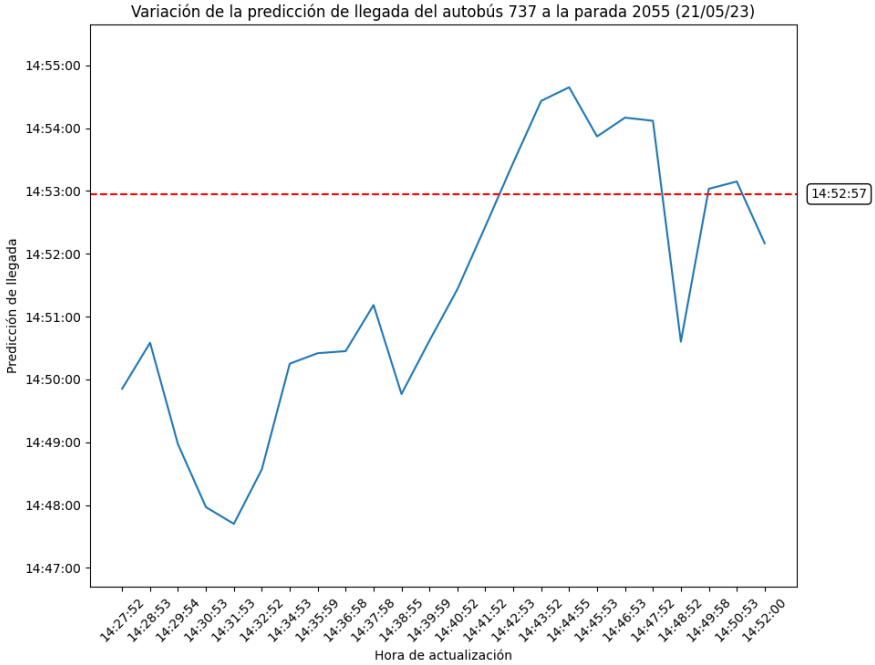


Gráfico 1. Autobús 737 llegando a la parada 2055.

Las dos siguientes gráficas, Gráficas 2 y 3, representan dos paradas cercanas al final del sentido de dirección al Palo. Se observa un patrón similar en ambas gráficas, que coincide con la hora en la que el autobús se acerca al puente que conduce a la Alameda Principal. Esto podría indicar la presencia de un atasco o la influencia de semáforos en el tiempo de llegada.

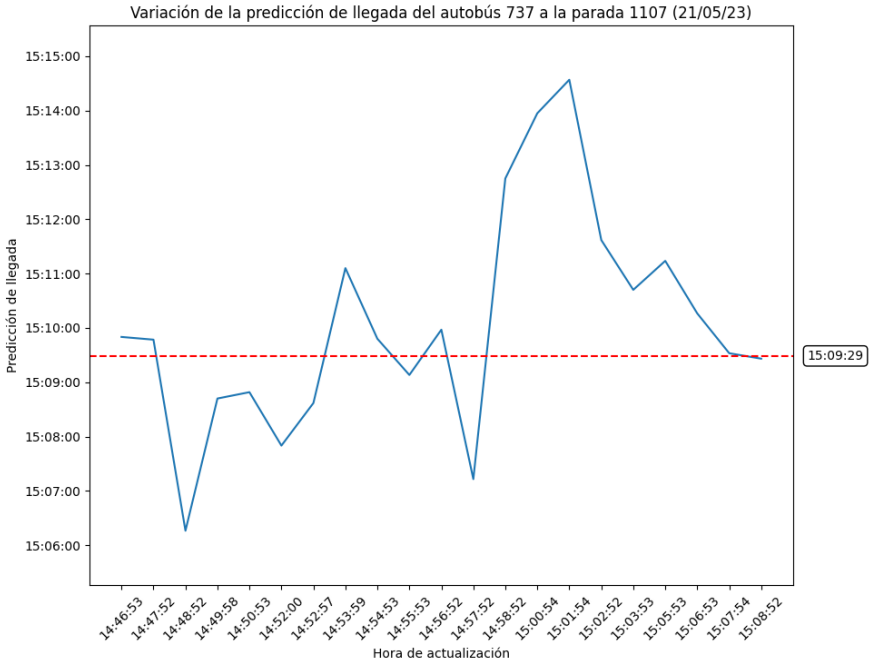


Gráfico 2. Autobús 737 llegando a la parada 1107.

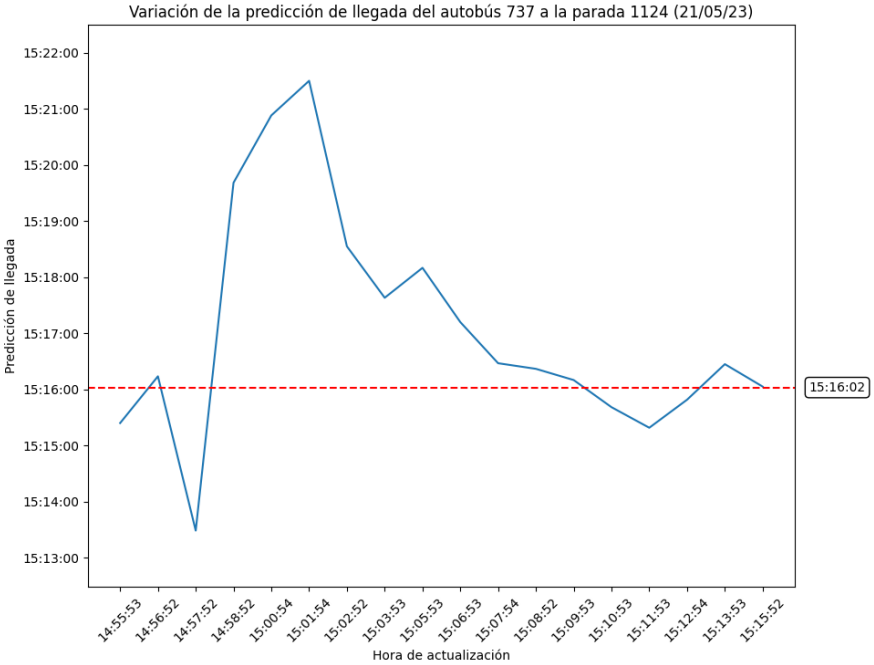


Gráfico 3. Autobús 737 llegando a la parada 1124.

La figura siguiente, Gráfico 4, muestra la llegada a la última parada antes del cambio de sentido con una precisión de predicción bastante precisa. Esto sugiere que no se han producido imprevistos significativos en las últimas paradas.

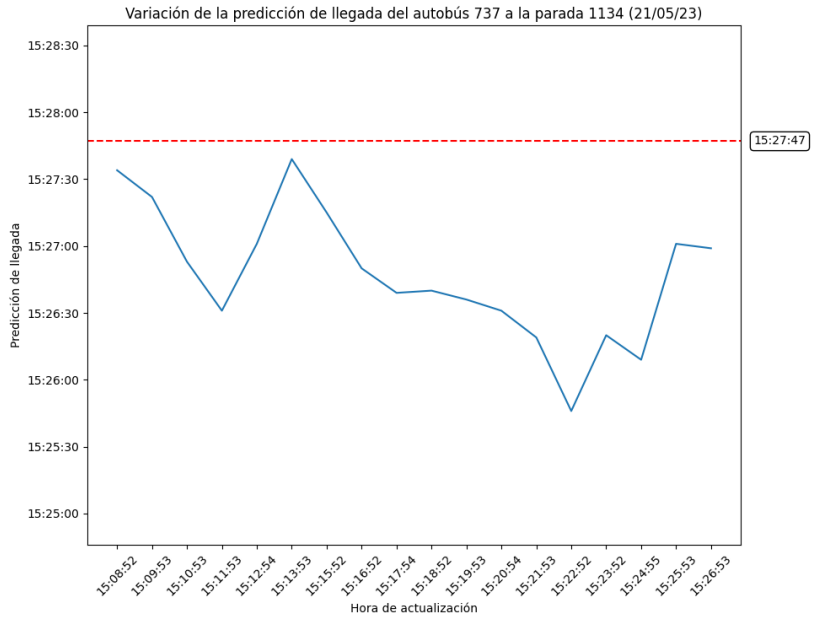


Gráfico 4. Autobús 737 llegando a la parada 1134.

En la gráfica 5 se puede observar la parada posterior al cambio de línea, donde se aprecia un periodo de tiempo en el que el autobús ha estado detenido, aproximadamente 10 minutos. Esto se determina al observar el momento en el que comienza a aumentar la predicción, indicando que el backend de la aplicación ha detectado la parada y el retraso en la llegada a la siguiente parada. Además, aproximadamente a finales de las 15:37, se observa que el autobús comienza a moverse nuevamente y la predicción se estabiliza.

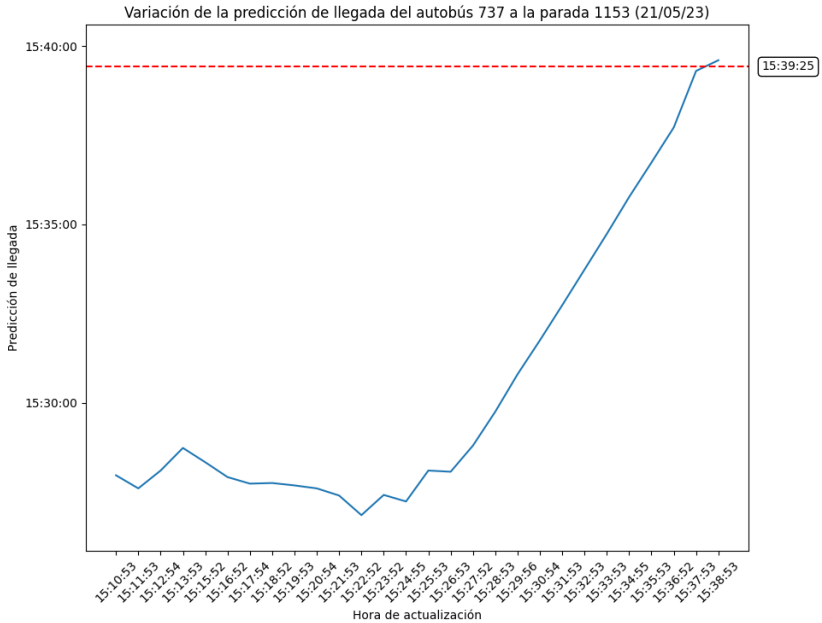


Gráfico 5. Autobús 737 llegando a la parada 1153.

En la gráfica 6 se pueden observar aún algunas repercusiones del cambio de línea, pero a partir de ese punto la precisión de las predicciones se ha estabilizado si mayor inconveniente.

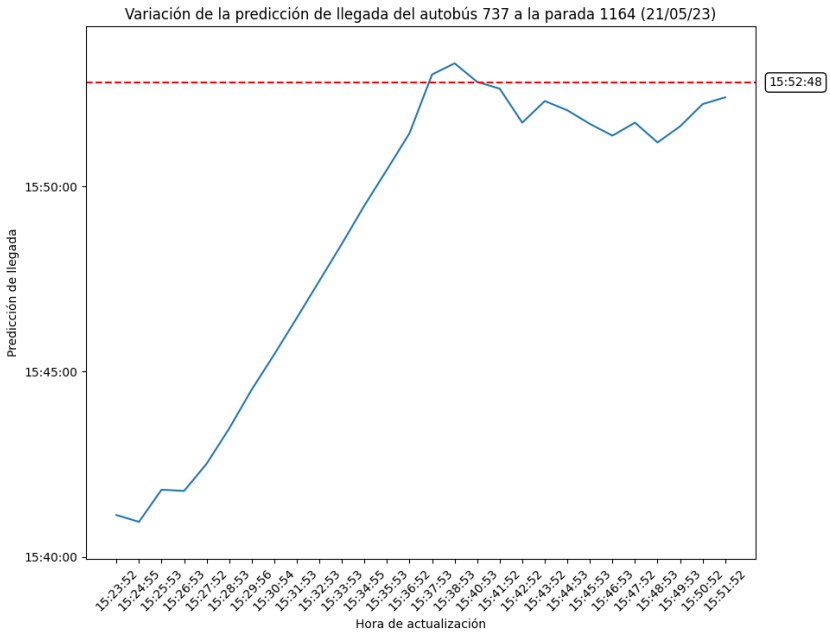


Gráfico 6. Autobús 737 llegando a la parada 1164.

En la gráfica 7, se puede observar un suceso destacable que ocurre a partir de las 15:50, donde se registra un retraso en la predicción que luego disminuye en los minutos siguientes y vuelve a aumentar nuevamente, acercándose a una predicción más acertada. Esta fluctuación podría indicar que, cuando el autobús se atrasó, el conductor aceleró para evitar llegar tarde y más tarde, tuvo que frenar un poco para no adelantarse a su hora.

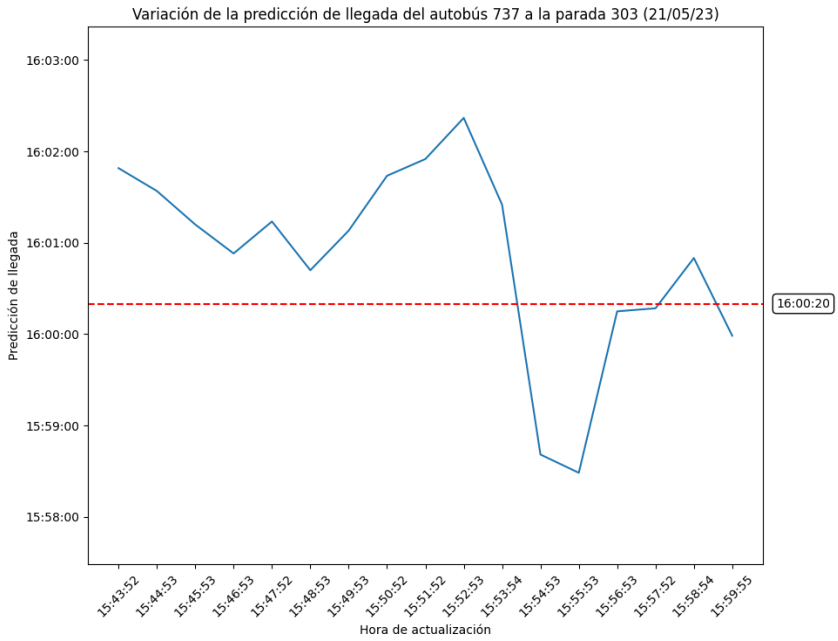


Gráfico 7. Autobús 737 llegando a la parada 303.

En la gráfica 8, se puede apreciar que las predicciones fluctúan alrededor de una hora aproximada de llegada, sin llegar a converger por completo. Aunque inicialmente se podría pensar que esta variación se debe a la zona cercana a la unión de la autovía y la entrada a Málaga, es importante destacar que los cambios en las predicciones son más antiguos y no se originan por estar en esta área en particular.

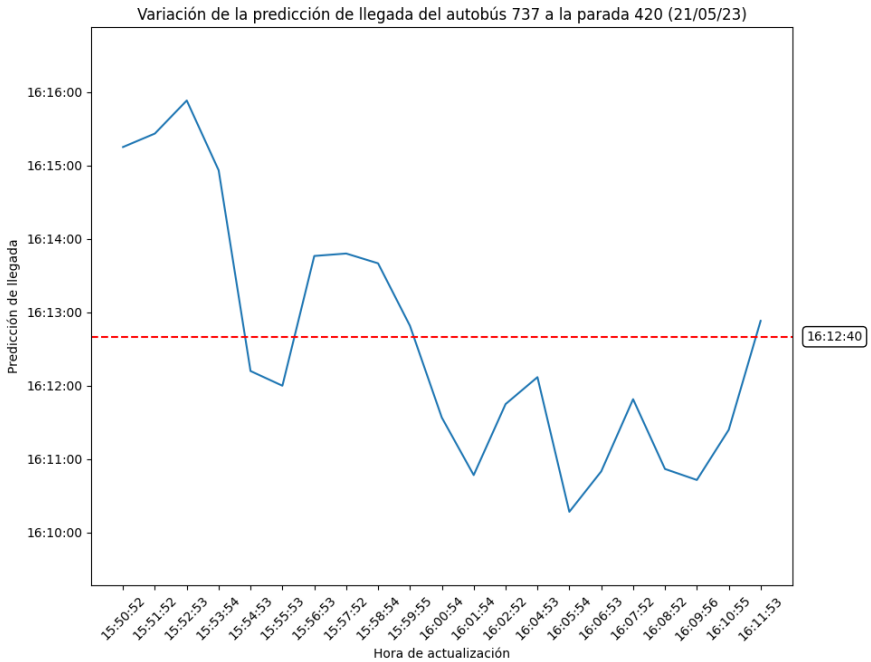


Gráfico 8. Autobús 737 llegando a la parada 420.

La parada con código 2011 es la última del recorrido, marca el final del recorrido antes del cambio de sentido en la siguiente parada y se pueden observar que las fluctuaciones anteriores aún están presentes. Sin embargo, es importante destacar que las predicciones comienzan a volver a converger después de pasar por la zona de la parada anteriormente analizada. En esta nueva etapa del recorrido, los motivos que causaban las fluctuaciones anteriores ya no tienen efecto, lo que resulta en una mayor precisión en las predicciones.

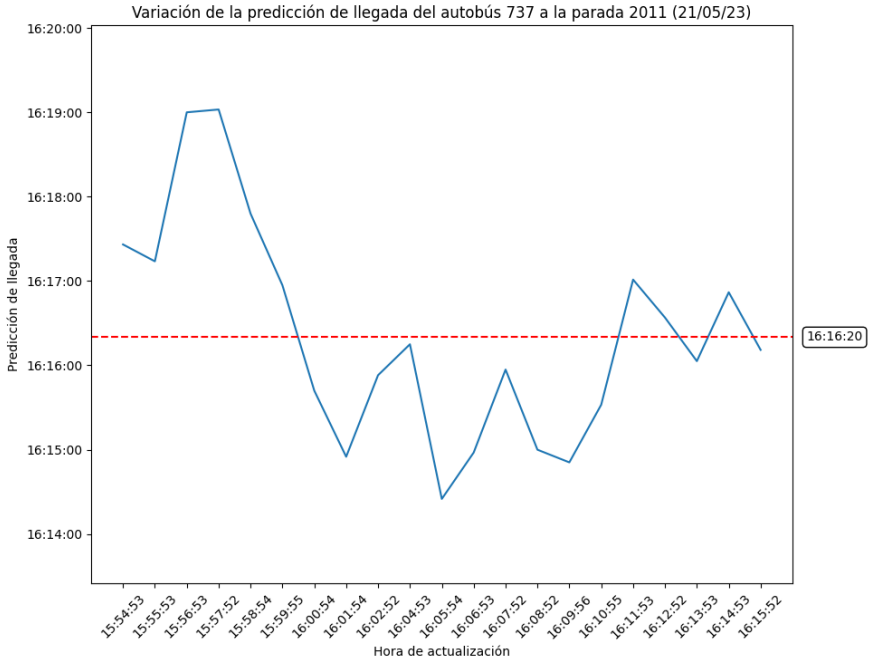


Gráfico 9. Autobús 737 llegando a la parada 2011.

4.4. Tercera prueba de precisión en predicción de llegada

En esta sección, se destacarán los resultados y patrones consistentes que han surgido a partir de las pruebas previamente realizadas. Gracias a la repetición de estas pruebas en diversos entornos y circunstancias, hemos observado resultados similares a los anteriores. Estos patrones recurrentes nos generan una comprensión más sólida de los fenómenos analizados y nos permiten confirmar nuestras conclusiones sobre su comportamiento.

En esta nueva prueba, se ha realizado un análisis utilizando un autobús y un día distintos a los previamente estudiados. Además, se han seleccionado paradas diferentes a las analizadas anteriormente. Como se ha comentado anteriormente, es importante tener en cuenta que en aquellas paradas donde se producen cambios de línea y descansos de los conductores, es probable que se generen retrasos significativos en las predicciones, los cuales serán corregidos una vez se supere dicha parada.

El análisis se inicia en la parada con código 476, ubicada a tres paradas desde el inicio de la línea. Aunque disponemos de pocas muestras, las dos últimas se acercan al objetivo debido a su proximidad en términos de fecha de llegada.

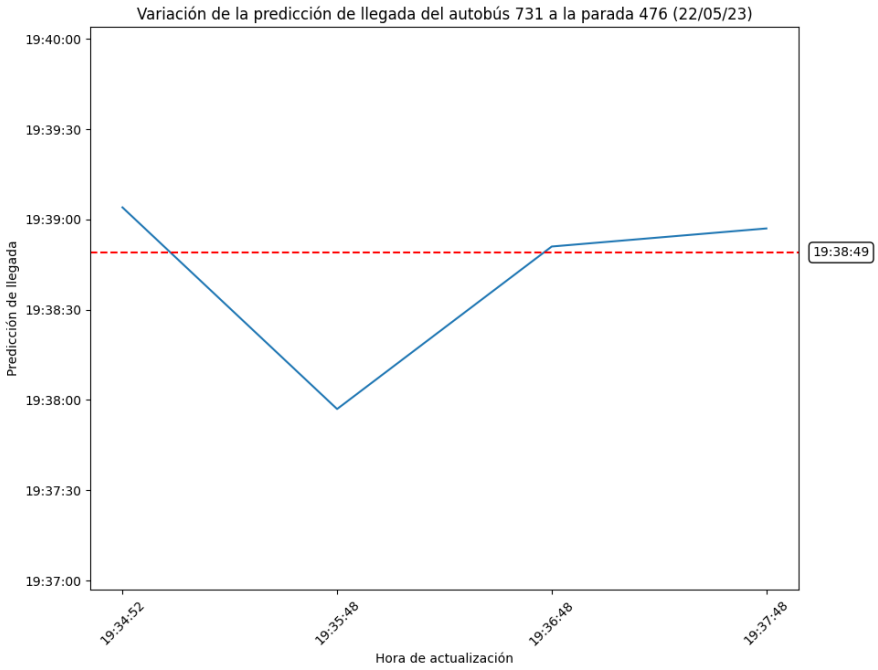


Gráfico 1. Autobús 731 llegando a la parada 476.

En esta segunda representación, se observa que los resultados, como en el caso anterior, cuentan con un número limitado de muestras. Sin embargo, aproximadamente 5 minutos antes de alcanzar el objetivo, la predicción se acerca considerablemente a la hora esperada. No obstante, posteriormente se produce una desviación, lo cual podría indicar que el conductor aceleró el vehículo para evitar un retraso significativo en el horario establecido.

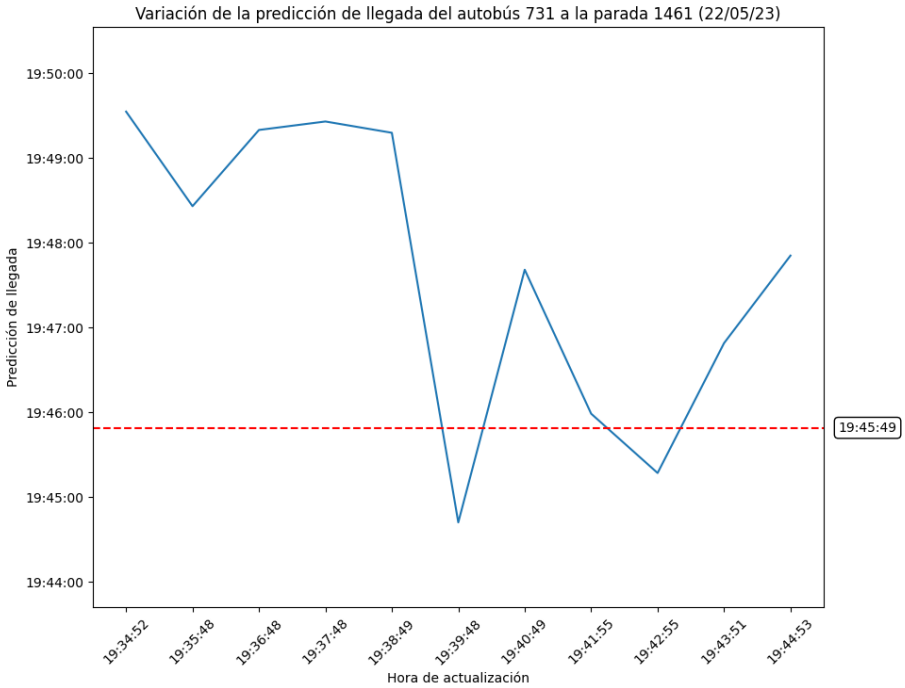


Gráfico 2. Autobús 731 llegando a la parada 1462.

En este caso particular, se puede observar que se produjo un cambio de conductor durante una de las paradas ubicadas en la Alameda Principal. Antes y después de este cambio, la predicción muestra una fluctuación alrededor de una hora específica. En la primera mitad de la gráfica, esta fluctuación ocurre en el rango de predicción entre las 20:00 y las 20:01, mientras que, en la parte final de la gráfica, los resultados varían alrededor del objetivo final.

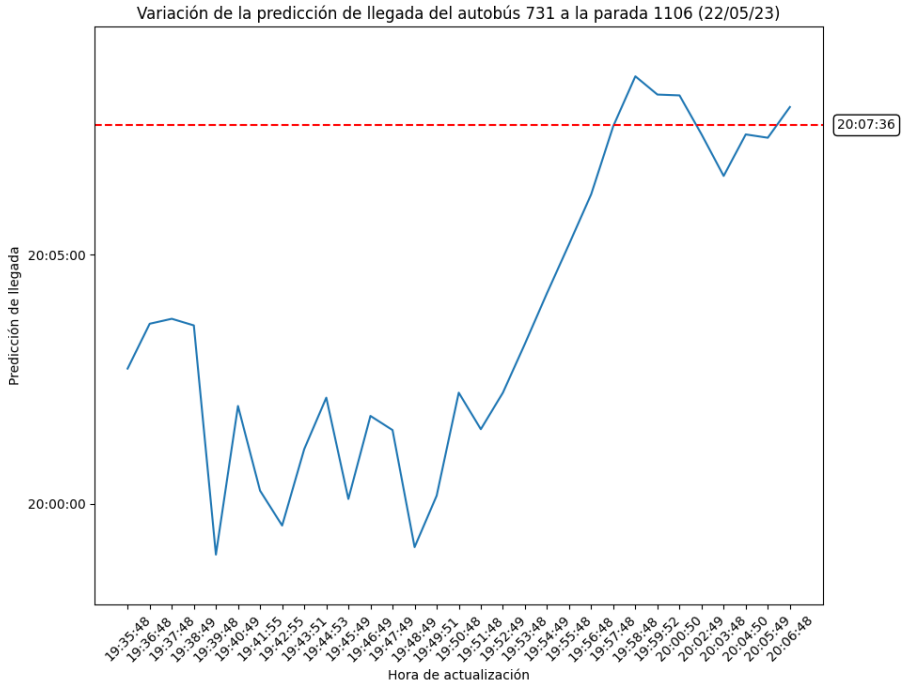


Gráfico 3. Autobús 731 llegando a la parada 1106.

En este otro caso podemos observar que la fluctuación de la predicción se sitúa alrededor de un minuto por debajo de la hora final, hasta que 5 minutos antes de llegar, el sistema empieza a recalibrarse y aumenta el techo de predicción para acercarse al resultado deseado.

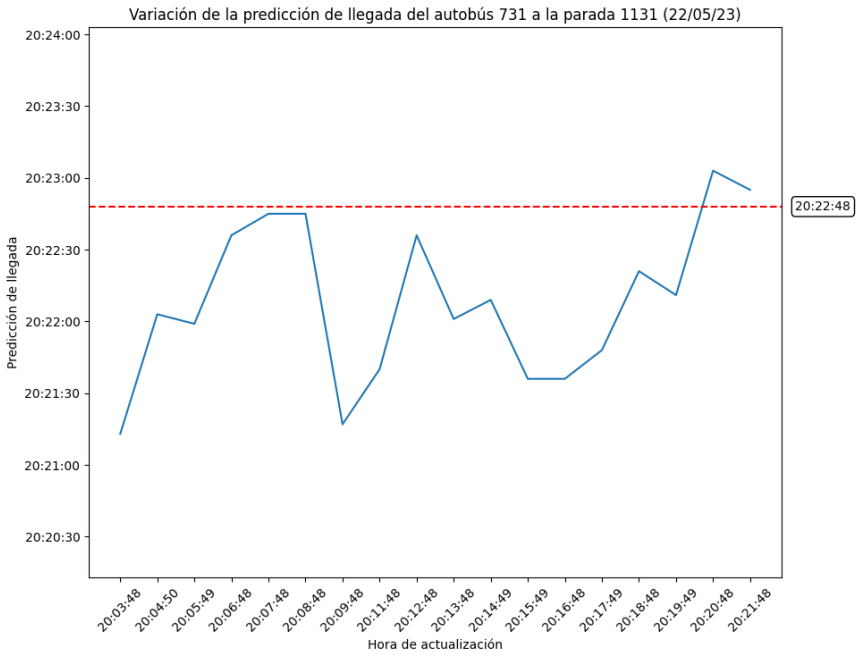


Gráfico 4. Autobús 731 llegando a la parada 1131.

Este es uno de los casos en los que el autobús está parado en la parada de cambio de sentido del Palo, en la Playa Virginia, y se afecta a todos los tiempos simulados posteriores. Una vez que el autobús retoma su recorrido podemos ver que la predicción empieza a variar, aunque acaba a aproximadamente un minuto del objetivo.

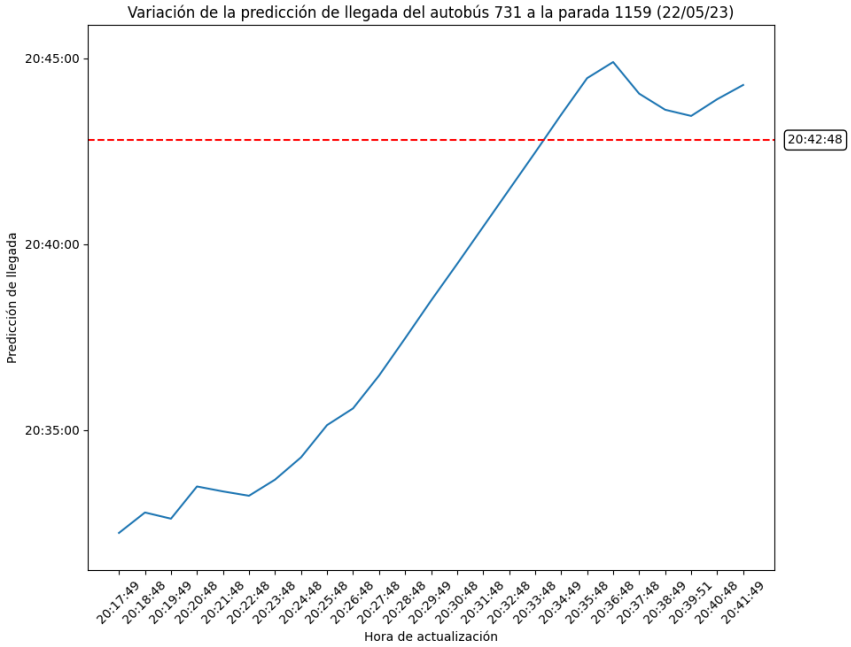


Gráfico 5. Autobús 731 llegando a la parada 1159.

En la gráfica 6 vemos aún las secuelas del cambio de sentido y cómo el autobús se había atrasado, pero ahora está acelerando, por eso la predicción se reduce constantemente hasta converger a menos de un minuto de la hora real.

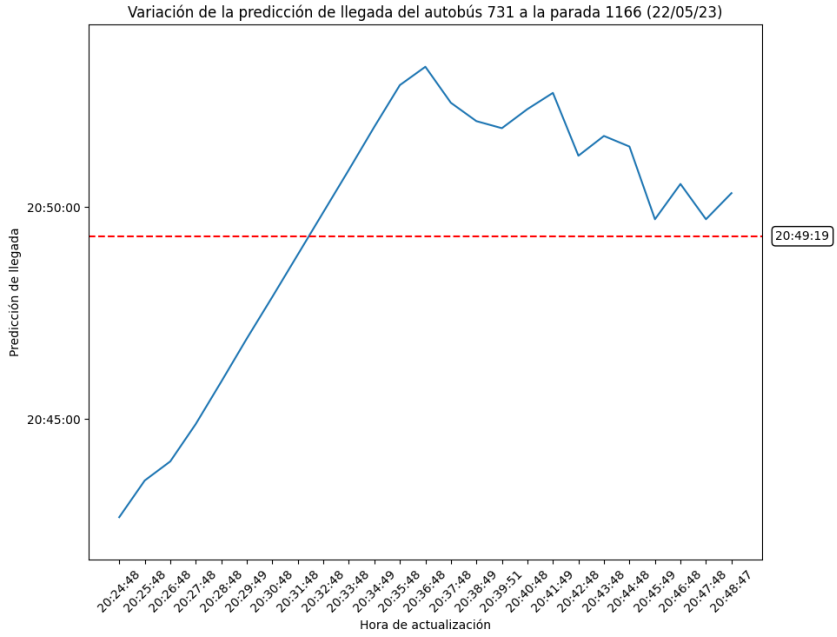


Gráfico 6. Autobús 731 llegando a la parada 1166.

Esta tabla de predicciones varía mucho más, pero, como en el caso anterior, se ve que la predicción tiene tendencia negativa, y al final tiene que volver a aumentarse para no fallar.

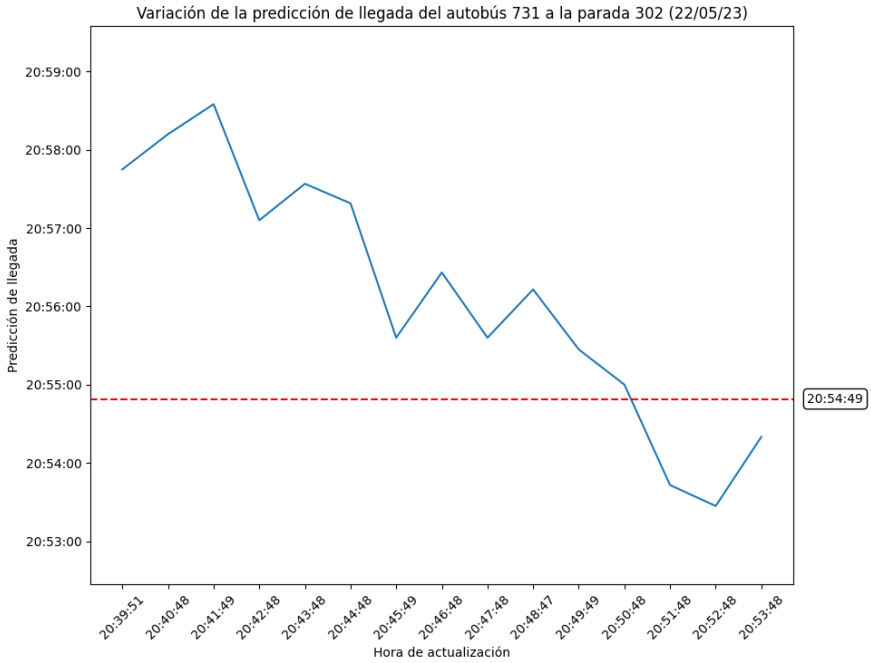


Gráfico 7. Autobús 731 llegando a la parada 302.

En esta gráfica podemos ver que el gemelo digital no siempre proporciona predicciones acertadas. Es posible que el tráfico que existe a las nueve de la noche entre el centro y

la salida de Málaga por la autovía tengan un impacto significativo. El resultado acaba fallando por cerca de dos minutos.

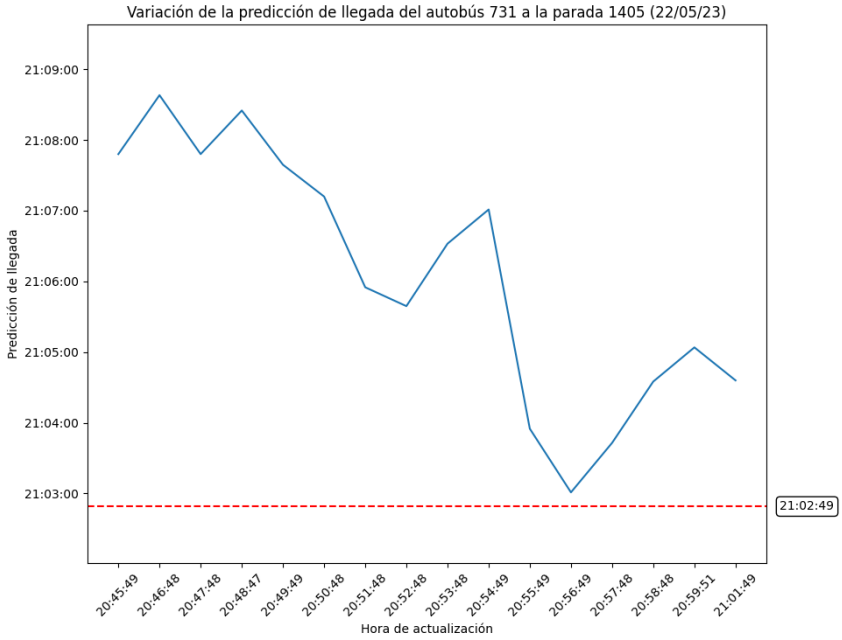


Gráfico 8. Autobús 731 llegando a la parada 1405.

En la última gráfica podemos ver los mismos patrones descendientes de la anterior, pero esta vez, cuando llega a la hora de la predicción converge en ella y nos da una predicción muy acertada.

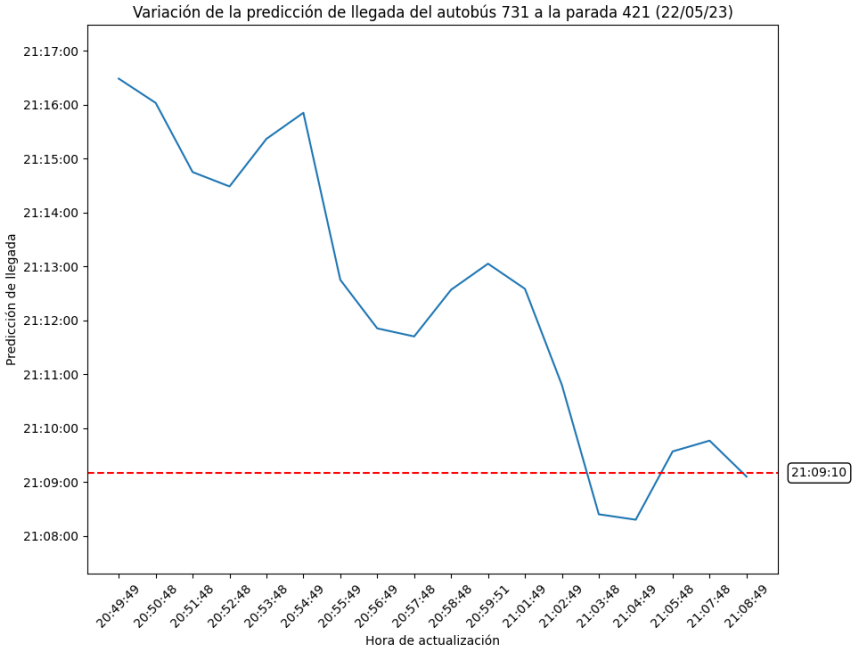


Gráfico 9. Autobús 731 llegando a la parada 421.

4.5. Variaciones por lluvia

El desarrollo de mejoras de precisión con respecto a este tema del tiempo se comentará en el apartado de mejoras, pero aprovechando que ha llovido en los últimos días de desarrollo de la memoria se ha decidido realizar pruebas para validar una hipótesis que teníamos desde el principio. Esta hipótesis es que la lluvia afecta al tiempo entre paradas, produciendo más atascos, accidentes y en general una peor conducción por la ciudad.

Para esta prueba se han tomado los datos del jueves 18 de mayo, de 14:00 a 15:00, una hora en la que llovió, y se han comparado con los datos medios de la misma franja horaria en el mismo día de la semana de las semanas anteriores, en las que no llovió. La primera gráfica presentada es la que relaciona directamente los tiempos medios entre paradas del caso seco, en rojo, con el mojado, en azul. Las líneas de la gráfica indican los valores en relación con el día seco, siendo este el 100%.

Como podemos observar, en la primera parada hay una gran variación entre los tiempos, esto puede deberse a que se tienen pocos datos de esa hora y se está tomando en cuenta que entre las primeras paradas no coincidió con el descanso que suele tomarse al ser un cambio de sentido, como ya se ha comentado anteriormente.

En el resto de la gráfica podemos ver que le línea azul, en lluvia, varía significativamente en algunas paradas. Un motivo que podría ser es que el servicio de autobuses está preparado para ello y tiene puestas medidas para evitar que el autobús se atrase, como un horario más holgado.

Por ejemplo, en el punto A sí podemos ver cambios grandes en las medias de tiempo, donde se consigue más de un 50% de tiempo extra entre las paradas de la Alameda Principal. Además en el centro de la gráfica tenemos una anomalía, este tiempo tan elevado probablemente signifique que el autobús estuvo parado en el cambio de sentido del Palo.

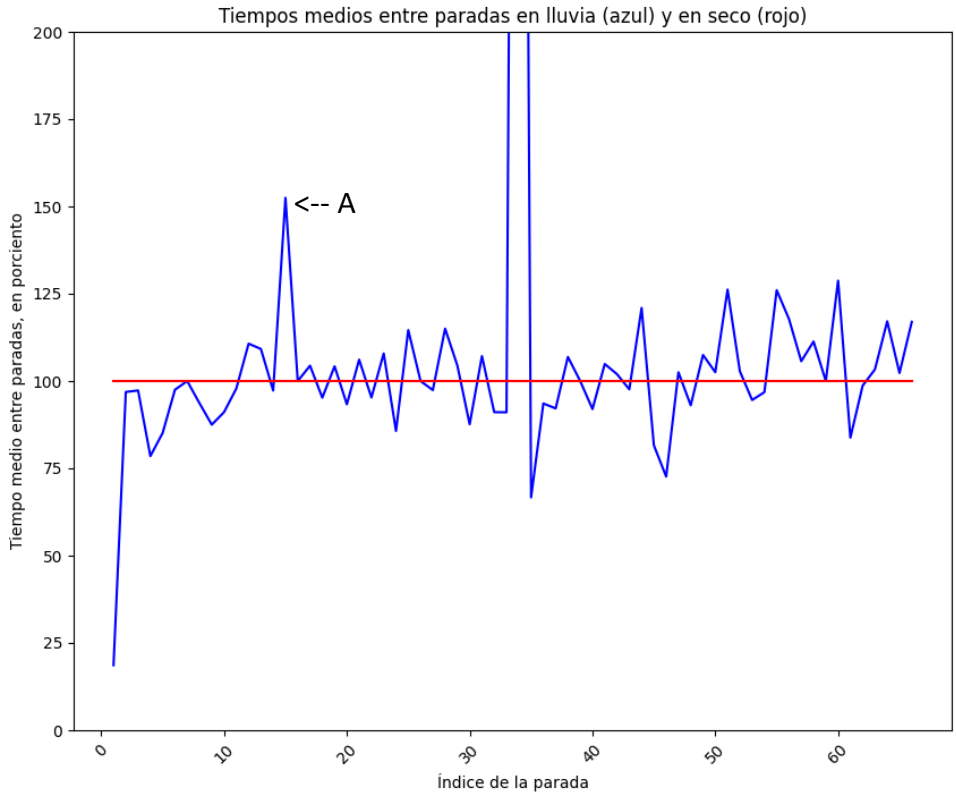


Gráfico 1. Medias de tiempos en seco y en lluvia, en porcentaje sobre el tiempo seco.

Aún no podemos afirmar que la lluvia altere el tráfico de autobuses, ya que, aunque las medias sufren una alteración, no es tan elevada como se esperaba. Sin embargo, podemos utilizar las desviaciones típicas como medida para evaluar la estabilidad de sus medias. Al observar el gráfico, podemos notar que el jueves 18 de mayo, entre las dos y las tres de la tarde, se ve claramente una estabilidad variable. En algunas paradas es más estable, es decir, tiene un valor menor, pero en otras supera por mucho la desviación de los días secos.

Por ejemplo, en la parada del centro de la tabla, y en las posteriores, se consigue una desviación típica mucho menora a las de los días secos. Esto significa una mayor estabilidad en las medias, aunque puede deberse a la cantidad de datos recogidos.

También tenemos que destacar el punto B, en la Avenida de Juan Sebastián Elcano, se consigue una desviación de más de un 180% del valor de un día seco. Otros puntos

destacables por su gran desviación son el punto C y D, corresponden a la parada de Cerrado Calderón y del tramo del Paseo de Sancha, respectivamente.

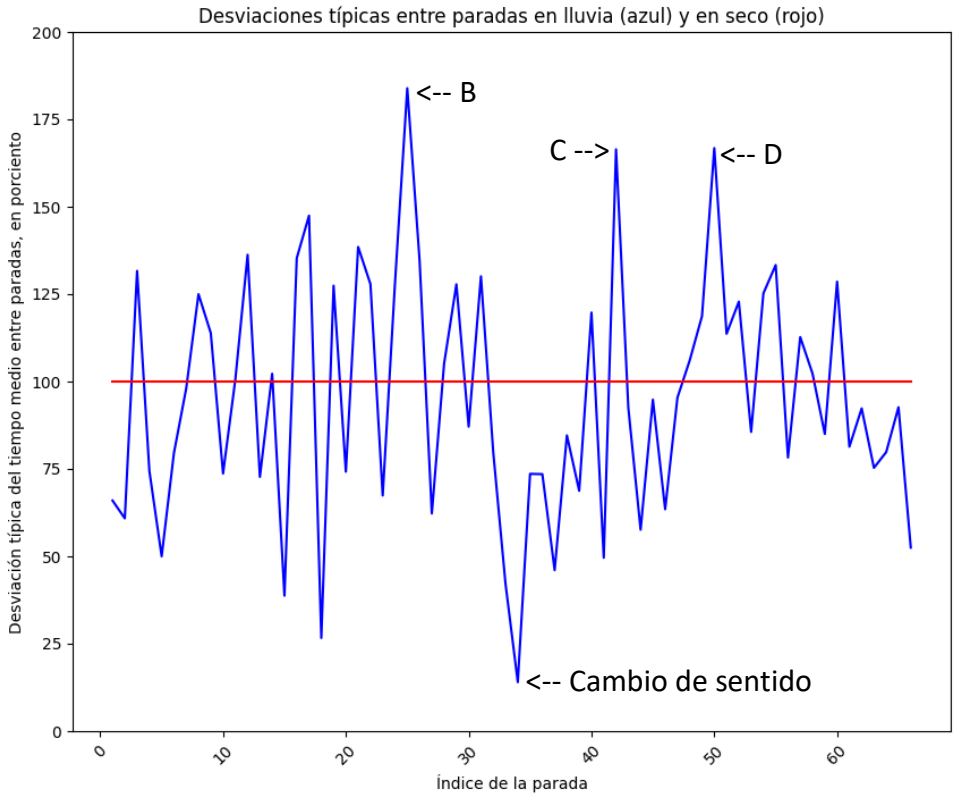


Gráfico 2. Desviaciones típicas de las medias de tiempos en seco y en lluvia, en porcentaje sobre el tiempo seco.

El tráfico puede experimentar variaciones significativas según la ubicación y las características de la zona. Es posible que no hayamos encontrado más variaciones significativas en los datos debido a que la congestión generada por el tráfico durante la lluvia no ha afectado tanto al recorrido del autobús en algunas zonas. Otra posible razón es que solo se ha realizado una prueba en un único día lluvioso. Para poder obtener conclusiones más sólidas, sería necesario realizar un estudio más amplio que compare múltiples días de lluvia con múltiples días secos comparando grandes cantidades de datos. Esto nos permitiría analizar con mayor precisión el impacto de las condiciones meteorológicas en el tráfico y evaluar si existen patrones consistentes en diferentes días de lluvia.

4.6. Variaciones por días laborables y festivos

En este apartado se ha realizado un análisis comparativo entre los días laborables y los días festivos de abril. Los días festivos en este caso son todos fines de semana, ya que el análisis se ha realizado sobre abril desde el 11 hasta el final de mes. Se ha evitado coger los días anteriores a este ya que fue Semana Santa y la línea 11 sufrió alteraciones en el recorrido que pasa por el centro de Málaga.

En el gráfico presentado, se muestra la media de los tiempos entre paradas para ambos tipos de días. La línea roja representa los días laborables, mientras que la línea verde representa los días festivos y fines de semana. Los valores están representados en porcentaje con respecto al día seco, teniendo este siempre un valor del 100%.

Una observación interesante es que, durante los días laborables, los tiempos de los días festivos son consistentemente menores a los tiempos de los días festivos. Esta diferencia puede atribuirse al mayor flujo de tráfico generado por las personas que se desplazan hacia y desde sus lugares de trabajo. Esta hipótesis es reforzada por las siguientes observaciones.

Las paradas con mayor variación entre festivos y laborables se encuentran en su mayoría llegando, estando o saliendo de la zona de la Universidad, ya que en días festivos no hay clases por lo que reduce significantemente el número de desplazamientos en la zona. Estas zonas son los puntos A y B de la gráfica. El otro tramo con una media significantemente menor a la laborable es el punto C, que corresponde a la Alameda Principal, y el punto D, que se refiere a la Avenida del Pintor Joaquín Sorolla.

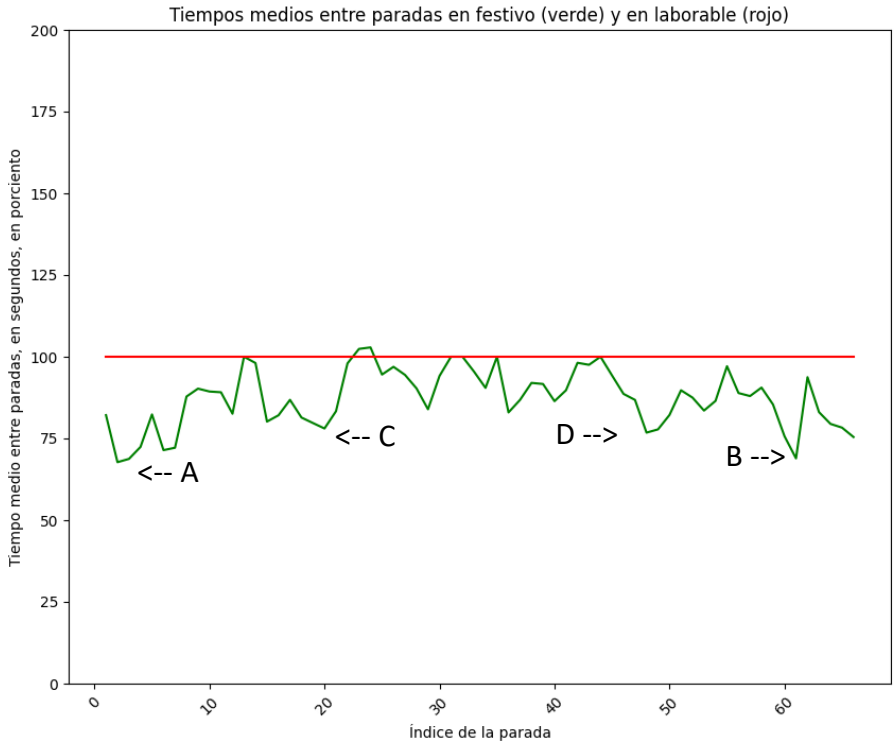


Gráfico 1. Medias de tiempos días festivos y laborables en porcentaje, sobre el tiempo laborable.

En la gráfica se puede apreciar que la parada de entrada a la ciudad muestra una menor variación en los tiempos de llegada durante los días festivos, lo que indica una mayor consistencia en los horarios. Sin embargo, las paradas ubicadas cerca o dentro del centro, especialmente la parada de la Plaza de Toros con índice 15 y la parada de Miramar con índice 20, presentan desviaciones significativas en los días festivos. Esto sugiere que puede haber factores específicos en esos lugares que afectan la puntualidad de los autobuses durante los días festivos.

Además, hay que destacar la parada de Miramar Bellavista en dirección el Palo, con índice 21, ya que tiene un horario menos estable los días laborables. También se distinguen variaciones mayor intensidad en días laborables en las paradas con índice 23 y 32, que corresponden a las de los Baños del Carmen y Padre Coloma.

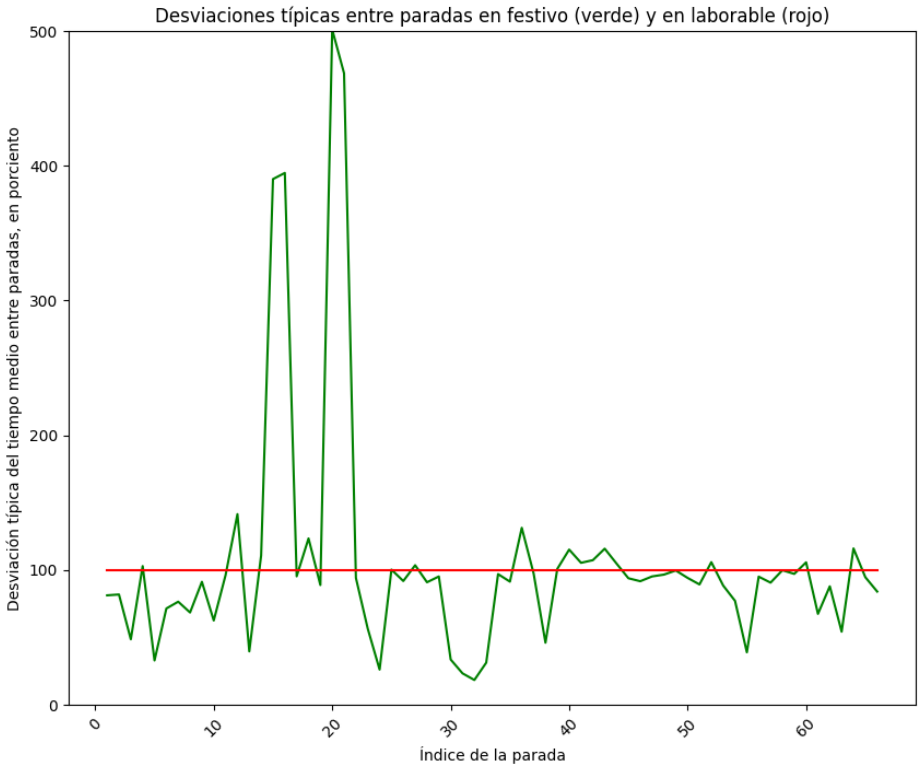


Gráfico 2. Desviaciones típicas de las medias de tiempos en días festivos y laborables, sobre el tiempo laborable.

Aunque existen otras paradas con variaciones en las desviaciones, estas diferencias no son tan significativas como las mencionadas anteriormente. Estos resultados sugieren que la influencia del flujo de personas que se dirigen al trabajo durante los días laborables afecta de manera general a los tiempos de llegada de los autobuses. Sería recomendable realizar un estudio sobre las horas punta durante los días laborables y el resto del día para evaluar el aumento promedio de la afluencia en las horas más concurridas. Identificar las horas punta y comprender cómo varía la cantidad de pasajeros a lo largo del día ayudaría a mejorar la eficiencia del servicio evitando aglomeraciones de usuarios.

4.7. Patrones en las pruebas

Basándonos en los análisis realizados, se pueden extraer ciertos patrones y conclusiones. Al examinar las gráficas de variaciones de medias y desviaciones típicas, se observa que los valores iniciales suelen ser elevados. Esto se debe a que, al inicio de la ruta, los autobuses tienden a detenerse durante períodos prolongados, lo que incrementa el tiempo entre la primera y segunda parada.

Se identifica un patrón particular en las paradas posteriores a un cambio de dirección. Cuando se produce un cambio de línea, los autobuses permanecen detenidos por un tiempo mayor al habitual, y el algoritmo de predicción no tiene en cuenta esta variable, requiriendo tiempo para recalibrarse. Durante este período de ajuste, las predicciones pueden retrasarse, lo cual se refleja en las variaciones de los tiempos entre paradas.

Se observa una tendencia en la cual las predicciones realizadas a 10 minutos o menos tienden a converger hacia los valores reales, especialmente cuando se encuentran a menos de 5 minutos. Por otro lado, las predicciones realizadas con más de 10 minutos de antelación muestran mayores diferencias con la hora de llegada final, generalmente presentando menor precisión.

Se pueden identificar patrones relacionados con el comportamiento de los conductores de los autobuses. En algunos casos, los conductores pueden tener que acelerar o frenar para ajustarse a su horario, generando un patrón con cambios repentinos en las predicciones.

Estos patrones y conclusiones adicionales proporcionan una mayor comprensión de las variaciones en los tiempos entre paradas y los factores que los afectan, como la hora del día, el tramo actual, las discrepancias en las predicciones y las condiciones meteorológicas. Si bien se requiere una investigación adicional, estas pruebas validan la eficacia del gemelo digital en la predicción y comprensión de los tiempos de llegada de los autobuses.

5. Conclusiones

5.1. Conclusiones y Líneas futuras

La implementación del gemelo digital en el servicio de autobuses urbanos de Málaga ha supuesto un avance significativo para mejorar la eficiencia y planificación del transporte público en la ciudad. A través de un modelo estadístico basado en datos muy limitados, se ha logrado optimizar el sistema tradicional de autobuses. A pesar de esto, no se han alcanzado todos los objetivos propuestos debido a las restricciones de tiempo y datos disponibles.

Para mejorar la precisión y completar el gemelo digital, se han identificado posibles mejoras y avances tanto en el sistema digital como en la experiencia del usuario. La implementación de estas mejoras permitiría maximizar el potencial del gemelo digital y acercarse aún más a resultados deseados.

1. Integración de información meteorológica. Asignar cada parada a un distrito o barrio y utilizar servicios como Aemet para obtener datos de meteorología en tiempo real. Esto permitiría analizar si está lloviendo o no en cada parada y mostrar esta información al usuario, lo cual sería útil para la planificación de viajes.
2. Considerar el tráfico en las calles. Utilizar herramientas proporcionadas por otras fuentes, como TomTom, para tener en cuenta el estado del tráfico en las calles. Esto podría mejorar la precisión de las estimaciones de llegada al tener en cuenta posibles congestiones de tráfico.
3. Detección de horarios de fin de servicio. Ampliar el gemelo digital para que reconozca cuándo un autobús está a punto de finalizar su horario y llegar a su parada

de destino final. Esto permitiría ajustar las predicciones y evitar mostrar resultados incorrectos cuando los autobuses se acercan al final de su recorrido.

4. Actualización del *backend* para considerar paradas de descanso o cambio de conductor. Implementar mejoras en el *backend* para reconocer paradas de descanso o cambios de conductor que suelen ocurrir en ciertas ubicaciones o momentos específicos del recorrido. Esto permitiría ajustar las predicciones futuras y considerar el tiempo adicional que se pierde durante estas paradas.
5. Identificación de problemas en el trayecto. Detectar situaciones anormales como averías, semáforos o paradas prolongadas que puedan retrasar el vehículo más de lo normal. Estos eventos podrían ser marcados como problemas y tratados de manera especial en el análisis. Por ejemplo, sumar el tiempo medio perdido en estas situaciones a las predicciones futuras y tener en cuenta paradas largas o cambios de conductor.
6. Mostrar predicciones en lugar de confirmaciones. Una mejora para la usabilidad del Cliente web podría ser mostrar las predicciones de todos los autobuses en lugar de solo aquellos cuya ubicación se conoce. Sin embargo, para que esta mejora tenga mejor usabilidad que la vista actual, habría que implementar otros avances para que la precisión sea óptima.
7. Adaptar la vista para mejorar la accesibilidad de personas con discapacidades visuales o motoras. Si logramos mejorar significativamente el gemelo digital y alcanzar los resultados deseados, esta mejora se volvería aún más relevante para acercar el servicio a personas desfavorecidas. Al hacer que la aplicación sea más accesible, se garantiza que todos los usuarios, independientemente de sus limitaciones, puedan beneficiarse de las funcionalidades y la información proporcionada por el gemelo digital.

Las mejoras al procesamiento de datos podrían incrementar la precisión y la calidad de las predicciones, además de proporcionar una experiencia más completa del gemelo digital del servicio de autobuses, pero para mejorar significativamente la precisión de la aplicación habría que recurrir a mejoras utilizando herramientas físicas. La instalación de sensores en los autobuses que proporcionen datos de posición en tiempo real sería

una opción viable para obtener información más precisa y actualizada sobre la ubicación. Esto permitiría mejorar significativamente la calidad de las predicciones acercándose más en tiempo a la actualidad y nos daría información real sobre cuándo se llega a cada parada.

Otra alternativa sería utilizar la ubicación GPS de los teléfonos móviles de los usuarios como una fuente de datos para ajustar los movimientos reales de los autobuses en el trazado. Esta solución aprovecharía la gran disponibilidad de dispositivos móviles para recopilar información, lo que permitiría obtener una imagen más precisa de la situación del servicio de autobuses. Esto haría posible analizar la duración real de las paradas saltándose completamente el problema de tener una única foto cada 60 segundos. Además, se podrían analizar datos adicionales, como el número de personas a bordo de cada autobús.

Aumentar la frecuencia de actualización de la información de los autobuses sería una solución más sencilla de implementar y podría mejorar significativamente la precisión del gemelo digital del servicio de autobuses de Málaga. Al actualizar la información con mayor frecuencia, por ejemplo, cada 15 segundos en lugar de cada minuto, se obtendría una imagen más actualizada y real del estado de los autobuses en la línea.

Sin embargo, la mayoría de estas soluciones traerían consecuencias indeseadas, como un aumento sustancial de la carga computacional y una reescritura y ampliación de parte del código, aunque esto último no sería tan complejo ya que se pensaron en estas mejoras durante el desarrollo.

En conclusión, implementar estas mejoras en el gemelo digital del servicio de autobuses de Málaga nos acercaría a la creación de un sistema más preciso y completo que refleje de manera fiel la realidad del sistema original. Estas mejoras nos permitirían analizar con mayor precisión y detalle el funcionamiento del servicio de autobuses, lo que a su vez nos brindaría la oportunidad de identificar áreas de mejora y optimización.

La implementación de estas mejoras en el gemelo digital del servicio de autobuses de Málaga no solo traería beneficios a los usuarios, sino que también brindaría valiosa información para la gestión y la toma de decisiones por parte de los responsables del transporte público. Al lograr una mayor precisión en el gemelo digital y aprovechar tecnologías avanzadas, como la integración de datos meteorológicos, la consideración

del tráfico y la detección de eventos inusuales mediante el internet de las cosas, el servicio de autobuses de Málaga tendría la oportunidad de alcanzar un nivel de calidad y eficiencia similar al de otras ciudades que cuentan con sistemas de transporte público más avanzados. Esto nos acerca más a la visión de las ciudades inteligentes del futuro, donde la tecnología puntera se utiliza para optimizar los servicios públicos y mejorar la vida de los ciudadanos.

Bibliografía

Digital Twin Consortium

«Glossary of Digital Twins»

Casey Mullen

(2021) «Glossary of Digital Twins and Digital Twin technology»

Olivier Bloch, Jason Frankel

(2021) «An open source library that makes Digital Twins development easy»

Petra Hurtado, Alexsandra Gomez

(2021) «Smart City Digital Twins Are a New Tool for Scenario Planning»

Samantha Bresnahan

(2023) «Cities are being cloned in the virtual world. Here's what that means for the future»

John Kosowatz

(2021) «Smart Cities Look for Digital Twins»

Aaron Parrott, Lane Warshaw

(2017) «Industry 4.0 and the digital twin»

Coursera

(2023) «What Is a Digital Twin? Definition, Types, and Uses»

Christos Pylianidis, Val Snow, Hiske Overweg, Sjoukje Osinga, John Kean, Ioannis N. Athanasiadis

(2021) «Simulation-assisted machine learning for operational digital twins

Dr. Michael Grieves

(2014) «Digital Twin: Manufacturing Excellence through Virtual Factory Replication»

Cheryl Ajluni

(2022) «Why AI and digital twins could be the keys to a sustainable future»

B Danette Allen

(2021) «Digital Twins and Living Models at NASA»

Bernard Marr
(2017) «What is Digital Twin technology - And why is it so important?»

Eclipse Foundation
«Eclipse Ditto»

Thomas Jäckle
(2022) «Eclipse Ditto uses Digital Twins to enable IoT interoperability»
(2018) «Example demonstrating connectivity to an MQTT broker»

David Schwilk
(2021) «Ditto example with Azure»

MongoDB
«Documentation»
«MongoDB Atlas. The multi-cloud developer data platform»
«Explaining BSON»

W3Schools
«What is JSON? »

Sebastián Ramírez
«FastAPI»

Reed Barger
(2021) «Is React a Library or a Framework? Here's Why it Matters»

Meta Open Source
«React»

Facebook Open Source
«Presentando JSX»

Material UI SAS
«Material UI»

Paul Le Cam
«React Leaflet»

OpenStreetMap
«OpenStreetMap proporciona datos de mapas para miles de sitios web, aplicaciones móviles y dispositivos de hardware»

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA