Simulación de movilidad urbana mediante sistemas multi-agente

Trabajo de Fin de Grado

Ingeniería Informática

Julio de 2022

Blanca Mellado Pinto

María Belén Pérez Lancho

Sara Rodríguez González

Alfonso González Briones

AUTORIZACIÓN

La Dra. María Belén Pérez Lancho, la Dra. Sara Rodríguez González y el Dr. Alfonso González Briones, profesores del Departamento de Informática y Automática de la Universidad de Salamanca

HACEN CONSTAR

que el presente documento titulado "*Simulación de movilidad urbana mediante sistemas multi-agente*" ha sido realizado por Blanca Mellado Pinto, con DNI 70909868V, y constituye la memoria del proyecto realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de la Universidad de Salamanca.

Para que así conste a todos los efectos oportunos.

Salamanca, 7 de julio de 2022

PEREZ LANCHO MARIA BELEN - 07858727H 2022.07.07 17:40:15 +02'00'

RODRIGUEZ GONZALEZ SARA - 70864126E Firmado digitalmente por RODRIGUEZ GONZALEZ SARA - 70864126E Fecha: 2022.07.07 21:52:44 +02'00'

GONZALEZ BRIONES ALFONSO - 52415611Z Firmado digitalmente por GONZALEZ BRIONES ALFONSO - 52415611Z Fecha: 2022.07.07 17:52:58 +02'00'

AGRADECIMIENTOS

A mis tutores, Belén, Sara y Alfonso por sus buenos consejos, su excepcional apoyo y por dedicarme su tiempo.

A mi familia, amigos y seres queridos, especialmente a mis padres, a mi hermana, a Łukasz y a Clara por ser un apoyo moral en las dificultades.

A Salenbici por facilitarme los datos utilizados en este trabajo.

A todos los profesores y personas que me han proporcionado los conocimientos necesarios para realizar este trabajo.

RESUMEN

En este trabajo se presenta un sistema de balanceado para un servicio de préstamo de bicicletas, basado en la cooperación del usuario a cambio de recompensas. Este se ha implementado como un sistema multi-agente, utilizando el *framework* JADE. Las estaciones son representadas por agentes, y un agente servidor gestiona las consultas y préstamos de los usuarios. Una de las ventajas de esta implementación es la posibilidad de añadir o eliminar dinámicamente estaciones durante su funcionamiento. Otra ventaja es que permite distribuir en distintas máquinas los agentes.

El sistema es también capaz de predecir diariamente la demanda de las estaciones, utilizando métodos de *Machine Learning*, y de utilizar esta información en el balanceado del sistema. Como parte del proyecto se han estudiado distintos modelos de predicción.

Se ha utilizado el simulador de movilidad urbana SUMO para analizar los efectos del uso del sistema propuesto en el balanceado del servicio de préstamos de la ciudad de Salamanca, “Salenbici”. Se han llevado a cabo distintas simulaciones variando el nivel de colaboración de los usuarios. Para estas simulaciones se han utilizado datos históricos reales del uso de este servicio.

Palabras clave: SUMO, simulación, sistema multi-agente, JADE, BSS, BRP, *Machine Learning*, movilidad urbana, *smart cities*

ABSTRACT

On this work, we propose a balancing system for a bike sharing service, based on the cooperation of the user in exchange for rewards. This system is implemented as a multiagent system, using the JADE framework. The stations are represented by agents and a server agent manages the consultations and loans of the users. An advantage of this implementation is that it allows to add or remove dynamically stations on run time. Another advantage is that it allows to distribute the system in different machines.

The system also predicts the expected demand of the station for the day, using Machine Learning methods. This information is used in the balancing of the system. As part of the project, different models were studied for this task.

The urban mobility simulator SUMO was used to test the effects of the use of the proposed system on the balancing of the bicycle loan service of the city of Salamanca, “Salenbici”. Different simulations were run with varying levels of collaboration from the users. For this simulations, real historical data from the use of the service was used.

Keywords: SUMO, simulation, multi-agent system, JADE, BSS, BRP, Machine Learning, urban mobility, smart cities

TABLA DE CONTENIDOS

TABLA DE FIGURAS

TABLA DE TABLAS

LISTADO DE ABREVIATURAS

BRP: *Bike sharing Rebalancing Problem*

MAS: *Multi-Agent System*

ML: *Machine Learning*

BSS: *Bike Sharing System*

BRP: *Bike Sharing Rebalancing Problem*

DRT: *Demand Responsive Transport*

V2V: *Vehicle 2 Vehicle*

P2V: *Person 2 Vehicle*

AOP: *Agent Oriented Programming*

OOP: *Object Oriented Programming*

BDI: *Belief Desire Intention*

REST: *REpresentational State Transfer*FIPA: *Foundation for Intelligent Physical Agents*

ACL: *Agent Communication Language*

AP: *Agent Platform*

MTS: *Message Transport System*

AMS: *Agent Management Service*

DF: *Directory Facilitator*

AID: *Agent Identifier*

HAP: *Home Address Platform*

SUMO: *Simulation of Urban Mobility*

TraCI: *Traffic Control Interface*

XML: *eXtensive Markup Language*

kNN: *k Nearest Neighbors*

SVM: *Support Vector Machine*

SVC: *Support Vector Classification*

SVR: *Support Vector Regression*

API: *Application Programming Interface*

JSON: *JavaScript Object Notation*

OSM: *Open Street Maps*

JADE: *Java Agent DEvelopment framework*

HTTP: *HyperText Transfer Protocol*WGSI: *Web Server Gateway Interface*

AEMET: *Agencia Estatal de METeorología*

IDE: *Integrated Development Environment*

CSV: *Comma-Separated Values*

URI: *Uniform Resource Identifier*

POI: *Point of Interest*

2 INTRODUCCIÓN

El siguiente documento constituye la memoria del trabajo de fin de grado del grado de Ingeniería Informática en la Universidad de Salamanca.

La idea de este proyecto nace de la unión de varios intereses y experiencias durante mi etapa universitaria.

Por un lado, el interés por este tema se inició con la beca de colaboración realizada durante el curso 2020-2021, en la cual analicé diferentes simuladores de tráfico, entre ellos SUMO. Mis tutoras en esta beca de colaboración fueron María Belén Pérez Lancho y Sara Rodríguez González.

Por otro lado, en el curso 2021-22 he realizado prácticas en el grupo de investigación BISITE en el área de movilidad urbana y *smart cities*. Estas prácticas fueron dirigidas por Alfonso González Briones y Guillermo Hernández González.

Los conocimientos adquiridos en estas dos experiencias motivaron la decisión de realizar este trabajo final de grado en el campo de la movilidad urbana y la simulación del tráfico.

Alfonso González Briones propuso la idea en la que se basa este proyecto, desarrollar un sistema de balanceado de estaciones mediante la colaboración del usuario y la predicción de la demanda mediante métodos de *Machine Learning*. Se decidió complementar el desarrollo de esta idea con el uso de un simulador de tráfico como herramienta para evaluar el sistema.

Por último, decidí implementar el balanceado de las estaciones como un sistema multi-agente debido a mi interés personal en esta área, y a sus posibilidades y adecuación para el caso de uso concreto.

También quiero agradecer al Servicio de préstamos de bicicletas del Ayuntamiento de Salamanca “Salenbici” por proporcionarme los datos necesarios para la realización de este trabajo.

2.1 Contexto

Los sistemas de préstamo de bicicletas (BBS) son una forma alternativa de transporte en áreas urbanas que recientemente goza de gran popularidad, especialmente en las grandes ciudades. Los usuarios tienen acceso a un conjunto de bicicletas distribuidas por la ciudad. Las pueden tomar prestadas en cualquier localización disponible y devolverlas en otra. En los BBS basados en estaciones, las bicicletas se encuentran distribuidas en un grupo de localizaciones fijas, denominadas estaciones.

Muchas ciudades están implementando este tipo de transporte compartido debido a sus beneficios para la salud de los ciudadanos, así como para el medio ambiente y para reducir los niveles de contaminación en los centros urbanos. Son además una alternativa barata para realizar desplazamientos cortos. Otros beneficios que se están

explorando son, por ejemplo, su efecto en la resiliencia en las comunicaciones del transporte público.

2.2 Problema

A pesar de todas las ventajas citadas, un factor que disuade a los ciudadanos de utilizar estos sistemas es la preocupación por la disponibilidad de bicicletas en las estaciones que desean utilizar.

Especialmente aquellos usuarios que utilizan este servicio regularmente y en desplazamientos diarios concretos (por ejemplo, entre casa y el trabajo o centro de estudios) necesitan que la falta de bicicletas o de espacios para dejarlas no suponga un inconveniente o un retraso en su jornada.

El balanceado de las estaciones es la distribución equilibrada de bicicletas entre las estaciones para prevenir la acumulación o la escasez, y las medidas tomadas para corregir estas. Normalmente, uno o varios empleados desplazan manualmente las bicicletas. La planificación eficiente de la ubicación de las estaciones y la distribución de las bicicletas según la demanda prevista ayudan también a mantener este equilibrio.

2.3 Solución propuesta

En este trabajo se explora un enfoque diferente al problema de balanceado de un BSS mediante un sistema que utiliza los propios viajes de los usuarios para reubicar dinámicamente las bicicletas entre las diferentes estaciones.

En la solución propuesta, el usuario obtiene una recompensa por escoger utilizar una estación sugerida por el sistema, en lugar de la más cercana. La estación que se propone como alternativa se escoge teniendo en cuenta la prioridad de la estación dentro del sistema (por la falta o exceso de bicicletas) y la distancia adicional que deberá ser recorrida por el usuario.

En el momento de planificar un desplazamiento el usuario consultará mediante una aplicación el sistema. La aplicación le ofrece información sobre las estaciones más cercanas a su origen y destino previstos y la disponibilidad de bicicletas y de espacios en estas. Adicionalmente, el sistema sugiere al usuario una estación cuyo uso puede ayudar a balancear el sistema. El usuario puede entonces aceptar la propuesta y, si se lleva a cabo, recibe la recompensa.

Se espera que los usuarios más flexibles o aquellos que utilizan las bicicletas de manera recreativa estén más dispuestos a utilizar este sistema.

2.4 Implementación de la solución

El objetivo principal de este trabajo es el diseño e implementación de este sistema, haciendo menos hincapié en el método provisto al usuario para utilizar este (aplicación, página web…).

La propuesta hace uso de un sistema multi-agente. Cada estación está representada por un agente que almacena la información sobre su estado. Las estaciones utilizan mensajes para comunicar este estado.

El usuario se comunica con este sistema mediante un servletREST. Un agente servidor dentro del sistema multi-agente maneja las peticiones y la comunicación con el usuario y con las estaciones.

Por otro lado, cada estación predice la demanda esperada para el día a partir de datos como el día de la semana, mes, las condiciones climatológicas… mediante técnicas de *machine learning*.

Para evaluar el funcionamiento del sistema se utilizará una herramienta de simulación de tráfico, en concreto, el simulador de tráfico microscópico multimodal SUMO (*Simulation of Urban MObility*). En primer lugar, se simulará el funcionamiento del préstamo de bicicletas con datos reales como caso base. Después se realizarán varias simulaciones utilizando el sistema de balanceado propuesto, cada una de ellas con un diferente grado de colaboración de los usuarios. Los resultados se analizarán en base a dos parámetros, la distancia adicional recorrida por el usuario respecto al caso base y la mejora en la distribución de las bicicletas en las estaciones.

3 TRABAJOS RELACIONADOS

En este trabajo final de grado se aborda el problema del balanceado de un sistema de préstamo de bicicletas, denominado *Bike sharing Rebalancing Problem* (BRP). Consultando la literatura disponible se encuentran numerosos ejemplos de soluciones propuestas para este problema, todas ellas enfocando el problema desde diferentes perspectivas y empleando métodos variados.

El método más empleado para mantener el equilibrio en este tipo de sistemas es reubicar, usualmente durante la noche o al final de la jornada, las bicicletas entre las estaciones. Este trabajo puede ser llevado a cabo por operarios del BSS o mediante el uso de camiones, especialmente en ciudades grandes con un número elevado de estaciones y de bicicletas. La reubicación eficiente y la planificación de las rutas que deben seguir los camiones para optimizar el proceso constituyen en sí mismos un problema adicional. En el trabajo (Gaspero, Rendl, & Urli, 2013), los autores proponen un sistema que utiliza CP (C*onstraint Programming*) y LNS (L*arge Neighborhood Search*) para planificar eficientemente dichas rutas. Otro ejemplo es (Li, y otros, 2021), que utiliza el aprendizaje reforzado con el mismo propósito.

Otra forma de afrontar el problema del balanceado de estaciones es la planificación previa de las ubicaciones de las estaciones o del número de bicicletas necesarias en cada estación, con el objetivo de cumplir con la demanda de los usuarios, y reducir la necesidad de trasladar bicicletas entre las estaciones.

La propuesta de (Liu, et al., 2015) considera los patrones de movilidad de los ciudadanos, puntos de interés de la ciudad y la estructura de la red de transportes. Estas características se utilizan para encontrar las ubicaciones óptimas de las estaciones mediante redes neuronales artificiales y algoritmos genéticos.

Existen también sistemas que utilizan datos históricos de los desplazamientos de los usuarios, junto con el estado actual del sistema, para predecir las tendencias del servicio. Esto es lo que propone el siguiente trabajo (Wang, He, Zhang, Liu, & H. Son, 2019), en el que se presenta eShare, una herramienta que utiliza datos del pasado y presente de un BSS para inferir el uso futuro.

Por último, existen varios ejemplos de soluciones que utilizan a los usuarios de diversas maneras para mantener el balanceado de un BSS. Algunas de ellas simplemente informan al usuario cuando no hay disponibilidad de bicicletas o espacios libres en la estación, o los redirigen a una estación cercana, como en el caso de (Aeschbach, Zhang, Georghiou, & Lygeros, 2015).

Otras, siguiendo un método similar al utilizado en este proyecto, incentivan a los usuarios a colaborar en el balanceado del sistema, como (Singla, Santoni, Bartók, Meenen, & Krause, 2015).

Resulta de gran interés el siguiente trabajo (Fernández, y otros, 2019), en el cual se proponen y comparan distintas estrategias de balanceado, escalables a BSS de grandes ciudades, algunas de las cuales utilizan incentivos para el usuario. Las

distintas estrategias propuestas son evaluadas utilizando datos del BSS de Madrid, BiciMAD, en el simulador Bike 3S, una herramienta que permite experimentar con BSS y evaluar distintas planificaciones.

En cuanto al uso de sistemas multi-agente en el balanceado, se ha encontrado por ejemplo el siguiente trabajo (Luo, Du, Klemmer, Zhu, & Wen, 2022). En este caso un controlador jerárquico genera planes, que se evalúan en paralelo utilizando un entorno multi-simulación. Los resultados de estas simulaciones se utilizan para mejorar el controlador mediante aprendizaje de refuerzo profundo.

4 OBJETIVOS

Una vez realizada la presentación del problema y de algunas de las soluciones que otros autores han propuesto para abordarlo, pasamos a detallar los objetivos principales del presente trabajo, enumerando los objetivos funcionales, técnicos, personales y medioambientales que nos hemos planteado.

4.1 Objetivos funcionales

- Gestión del balanceado de un sistema de préstamo de bicicletas basado en estaciones: El objetivo principal del sistema es equilibrar del número de bicicletas y espacios disponibles en cada estación, para intentar que siempre haya bicicletas disponibles en la estación de origen escogida por el usuario, y espacios libres para realizar la devolución de la bicicleta en la estación de destino.

- Almacenamiento de datos que representen el estado actual del sistema: El sistema debe almacenar la información pertinente sobre las estaciones, así como el número de bicicletas disponibles en cada estación y actualizarla apropiadamente.

- Información a los usuarios: Los usuarios podrán hacer uso del sistema antes de iniciar su desplazamiento a la estación para obtener información sobre la estación más cercana y el número de bicicletas disponibles. Similarmente, el sistema informará de la estación más próxima al destino y el número de espacios libres.

- Recomendación de estaciones alternativas: El sistema también deberá recomendar al usuario las estaciones cercanas que ayuden a balancear el BSS.

- Cálculo de las estaciones sugeridas al usuario: Se debe crear un algoritmo que, en base a la distancia y a las necesidades de las estaciones, seleccione aquellas que constituyan alternativas realistas para proponer al usuario que está consultando la aplicación.

- Predicción de la demanda de cada una de las estaciones: Mediante algoritmos de *Machine Learning* se intentará predecir diariamente la demanda de cada estación, es decir, los préstamos y devoluciones de bicicletas que se prevén para esa estación en el día. Esta predicción será utilizada por el sistema recomendador para balancear las estaciones.

- Gestión del préstamo y devolución de bicicletas: Se debe gestionar también el préstamo y la devolución de bicicletas en las estaciones.

- Simulación del funcionamiento del sistema: Mediante una herramienta de simulación de tráfico, se probará con datos reales el sistema para observar su funcionamiento y comprobar la mejora en el balanceado de las bicicletas.

4.2 Objetivos técnicos

- Adaptabilidad a cambios: Posibilidad de adaptarse a los cambios en la configuración de las estaciones sin detener el funcionamiento del sistema.

- Escalabilidad: Creación de un sistema que permita reflejar futuras ampliaciones del servicio, como pueden ser la introducción de nuevas estaciones o el crecimiento del número de usuarios.

- Modularidad: Para lograr los objetivos anteriores de adaptabilidad y escalabilidad, se debe lograr la división del sistema en componentes que pueden ser reemplazados o modificados sin afectar al resto.

- Sistema distribuido: Los distintos componentes del sistema se deben poder distribuir en varias máquinas, en caso de ser necesario.

- Concurrencia: Para asegurar la correcta gestión durante la utilización simultánea por varios usuarios, el sistema debe ser capaz de aceptar varias peticiones a la vez sin que se produzca un error debido a ello. Además, se debe evitar que la gestión concurrente de las peticiones incremente excesivamente los tiempos de respuesta.

- Sistema inteligente: El sistema debe ser capaz de aprender de los datos generados por su uso.

- Seguridad: Se debe proteger la información confidencial de los usuarios, tanto en el desarrollo del sistema como durante su uso.

- Aplicación a un caso de uso con datos reales: Para comprobar el correcto funcionamiento del sistema, se utilizarán datos reales de uso del BSS.

- Documentación: Se proporcionará una documentación clara y exhaustiva del trabajo realizado, que incluya la fase de investigación sobre el campo en el que se enmarca el proyecto, las herramientas utilizadas y las razones de su elección, el proceso de desarrollo del sistema, la descripción detallada de los aspectos técnicos del mismo, los resultados obtenidos y análisis de estos, y el manual de uso del sistema planteado.

4.3 Objetivos personales

- Conocer los proyectos e ideas que se están desarrollando actualmente para mejorar la calidad de vida de los ciudadanos, y el funcionamiento de las ciudades, especialmente en el área de la movilidad urbana y los BSS.

- Descubrir nuevas herramientas y adquirir habilidades en el uso, tanto de estas, como de otras conocidas previamente.

- Adquirir experiencia en el diseño y programación de sistemas multi-agente, un campo de interés personal.

- Realizar un proyecto desde su fase de conceptualización y análisis de requisitos, hasta el diseño, implementación y pruebas, todo ello empleando una metodología de ingeniería del software apropiada.

- Aplicar en un proyecto real los conocimientos adquiridos en la carrera de Ingeniería Informática, los cuales hasta este momento se han estudiado de forma aislada. Este proyecto de fin de grado permite ver el rol de estos conocimientos en la visión global de un sistema.

- Por último, adquirir la capacidad de gestionar el tiempo y los imprevistos propios de un proyecto más amplio, y adquirir habilidades personales que en un futuro ayuden a manejar las dificultades.

4.4 Objetivos sociales y medioambientales

Desde una perspectiva más amplia, el fin de este proyecto es proponer y desarrollar una idea que ayude a promover el uso de la bicicleta como transporte urbano, en este caso, mediante la mejora de un servicio disponible en la ciudad. Por tanto, en la realización de este trabajo se tienen en mente los beneficios, tanto sociales, como para el medioambiente y la salud de los ciudadanos, que este tipo de iniciativas defienden.

- Sociales: El usuario puede obtener ventajas y recompensas por involucrarse y participar activamente en el buen funcionamiento del sistema. Se promueve el uso del préstamo de bicicletas y la cooperación ciudadana en la mejora de los servicios de la ciudad.

- Salud de los usuarios: Caminar, utilizar la bicicleta, o los patines como medio para desplazarse por la ciudad frente a otros como el coche o el autobús constituye una forma de realizar ejercicio en el día a día, con los consecuentes efectos beneficiosos en la salud.

- Medioambientales: Fomentar el uso de la bicicleta también contribuye a reducir la contaminación de los núcleos urbanos, ya que se evita el uso de medios de transporte más contaminantes.

5 MARCO TECNOLÓGICO

5.1 *Smart cities*

5.1.1 Definición

El concepto de *smart city* o ciudad inteligente nace como consecuencia de dos cambios sociales experimentados en las últimas décadas: el crecimiento de las ciudades y de la población urbana frente a la rural y la evolución y adopción de las tecnologías digitales para mejorar diferentes aspectos del funcionamiento de dichas ciudades.

Por un lado, la población urbana global ha aumentado de un 30% en 1950 a un 56% en 2020 (75% en Europa). En 2050 se espera que el porcentaje de población que reside en áreas urbanas en el mundo sea del 68% (Gu, Andreev, & Dupre, 2021).

Por otro lado, las tecnologías relacionadas con el ámbito de las *smart cities* han tenido una gran evolución y han despertado el interés de la ciudadanía en los últimos años. Algunas de las más mencionadas en la literatura son la inteligencia artificial, el IoT (Internet de las cosas), las redes 5G, las tecnologías RFID y SDN (redes definidas por sensores), el *big data*, el *cloud computing* (computación en la nube) o el control en tiempo real del tráfico (P. Kasznar, y otros, 2021).

Otras tecnologías incipientes como el *blockchain*, la visión artificial, el *fog computing* o el *edge computing* serán necesarias para afrontar los futuros desafíos (Sánchez-Corcuera, y otros, 2019).

Aunque la integración de la tecnología en las ciudades para satisfacer las necesidades de la creciente población urbana explica la aparición de las *smart cities*, el término es amplio y en ocasiones difuso. Por ello se han propuesto distintas definiciones y criterios por los cuales una ciudad puede ser considerada inteligente. Algunas de ellas desvían el foco de la tecnología y lo sitúan en las personas. Es esencial que el ciudadano sea el centro de las iniciativas y éstas constituyan realmente una mejora en su vida, trabajo, ocio y forma de relacionarse con el ámbito urbano.

Las definiciones propuestas mencionan frecuentemente la optimización de los recursos, la mejora de las infraestructuras, el crecimiento económico y el uso de las ICT (tecnologías de la información y la comunicación), así como valores más abstractos como el progreso, la creatividad, la humanidad o el aprendizaje (Lara, Costa, Furlani, & Yigitcanla, 2016).

Además, durante la última década la sostenibilidad se ha convertido en un factor clave en la planificación de las ciudades, debido a la preocupación y concienciación sobre el impacto de las actividades humanas en el medio ambiente. El ideal de ciudad inteligente tiene en cuenta la integración social, la reducción de la contaminación y de los desechos y la sostenibilidad energética y ecológica (Trindade, y otros, 2017).

5.1.2 Ámbitos

La movilidad inteligente es solo uno de los ocho dominios propuestos para clasificar los ámbitos de aplicación de las *smart cities* (Bellini, Nesi, & Pantaleo, 2022):

- Organización inteligente (participación ciudadana, *eGovernment*…)
- Infraestructuras y vivienda inteligente (hogares, educación, turismo)
- Movilidad y transporte inteligentes
- Economía inteligente (*eCommerce*, comercio y servicios *peer-to-peer*…)
- Industria y producción inteligente
- Energía inteligente (energía sostenible, *smart grids*, iluminación inteligente)
- Medioambiente inteligente (monitorización de la calidad del aire, monitorización del tiempo meteorológico, gestión de los residuos…)
- Salud inteligente (hospitales inteligentes, telemedicina, *eHealth*…)

5.1.3 Movilidad urbana inteligente

La movilidad urbana inteligente agrupa un conjunto de iniciativas y avances de la tecnología cuyo propósito es cubrir las necesidades de movilidad de los ciudadanos y limitar los efectos nocivos del transporte privado. Implica descubrir nuevas e innovadoras formas de concebir el transporte de personas y mercancías más allá de los modelos tradicionales.

A continuación, se muestran algunos ejemplos, extraídos de (Butler, Yigitcanlar, & Paz, 2020):

- Sistemas de transporte inteligentes: Señalización adaptativa, control de tráfico, conexión entre vehículos (V2V, P2V), monitorización de flotas de transporte, sistemas de información de plazas de aparcamiento en tiempo real, semáforos inteligentes, sistemas de apoyo a la conducción…
- Conducción automática: Vehículos autónomos, vehículos sin conductor, conducción autónoma, drones…
- Alternativas a los carburantes: combustibles alternativos, vehículos híbridos, vehículos eléctricos…
- Sistemas de movilidad compartida: *carpooling*, compartición de vehículos *peer-to-peer, ridehailing*, *ridesharing*, *ridesourcing…*
- Sistemas DRT *(Demand responsive transport*)

- Sistemas de movilidad integrados: Transporte intermodal, sistemas integrados de pago, sistemas de movilidad interconectados…

Se han identificado los siguientes objetivos para evaluar los efectos de las distintas iniciativas relacionadas con la movilidad urbana:

- Aumentar la seguridad de los transportes disponibles para los ciudadanos
- Reducir la congestión del tráfico y los tiempos innecesarios de espera
- Reducir el consumo energético, por ejemplo, de carburantes
- Reducir el impacto negativo del transporte en el entorno, tanto directamente como indirectamente (cambio climático, efectos en el suelo, efectos para la salud de los ciudadanos…)
- Mejorar la accesibilidad de los ciudadanos al transporte

5.1.4 Sistemas de movilidad compartida

Un servicio de compartición de vehículos pone a disposición de varios usuarios un mismo vehículo o flota de vehículos. Estos pueden ser privados, pertenecer a un negocio, o tratarse de un servicio municipal. Este tipo de sistemas permiten ahorrar costes a los usuarios y reducir el impacto del uso de transportes privados. Existen diferentes modelos de servicio (Butler, Yigitcanlar, & Paz, 2020):

- Compartición de vehículos privados: Préstamo temporal de un vehículo, puede ser *peer-to-peer* o B2C (*business to consumer*).
- *Ridesharing*: Emparejamiento de personas con trayectos similares para compartir vehículo. Los usuarios pueden compartir los gastos del viaje o turnarse en la conducción.
- *Carpooling*: Caso específico de *ridesharing* en el cual los usuarios comparten periódicamente o durante un trayecto su vehículo privado, generalmente un coche.
- *Ridesourcing*: Servicio similar a los taxis, con algunas diferencias en cuanto al tipo de licencia y flexibilidad de las tarifas, que no están reguladas. Generalmente, en el *ridesourcing*, el transporte siempre se reserva y el precio se indica antes del viaje. Se pone en contacto a los potenciales pasajeros con los conductores a través de aplicaciones.
- Sistemas de transporte alternativos o basados en demanda: Ofrecen una alternativa al transporte público, con rutas flexibles o dependientes de la demanda y menor capacidad. Un ejemplo es el *paratransit,* destinado a grupos específicos que requieren una mayor accesibilidad del vehículo o una ruta individualizada.

- Servicios de préstamo de transporte: Un ejemplo de estos son los *bike sharing systems* o BSS. Se pone a disposición del usuario un conjunto de vehículos (*scooters*, patines, bicicletas…) que este puede tomar prestados y devolver en diferentes localizaciones. Estos servicios pueden ser municipales o proporcionados por una compañía privada.

5.2 Simulación de tráfico

Una simulación es una representación dinámica de un aspecto del mundo real realizada mediante un ordenador. Esta representación puede tener como propósito predecir el comportamiento futuro de este, observar el efecto de una modificación en alguno de sus parámetros o ayudar a su visualización.

El tráfico es un elemento dinámico con unas características que hacen difícil su análisis, entre las que destacan:

- El número de participantes en el sistema es elevado.
- Los participantes tienen objetivos distintos, muchas veces opuestos entre ellos o contrarios al funcionamiento óptimo del sistema.
- Hay muchas variables difíciles de controlar, por ejemplo, las condiciones meteorológicas o el comportamiento de los conductores.
- Existe un gran número de interacciones ocurriendo simultáneamente.

Es por ello por lo que las simulaciones son una herramienta frecuente en el campo de la ingeniería del transporte, ya que permiten, por ejemplo, observar el funcionamiento de una planificación o sistema desarrollado en el tráfico, antes de su implementación (Pursula, 1999).

5.2.1 Tipos de simulaciones

Uno de los criterios para clasificar las simulaciones es su granularidad:

- Macroscópica: Modela flujos de vehículos o patrones en el tráfico
- Mesoscópica: Describe las rutas de cada vehículo de manera individual
- Microscópica: Introduce un mayor grado de precisión, modelando aspectos de los vehículos como los cambios de carriles o incorporación a carreteras, la interacción entre los vehículos, la respuesta a incidentes…

Las simulaciones microscópicas requieren más recursos computacionales, ya que el número de parámetros a modelar es mayor, y los cálculos pueden resultar más costosos para el procesador (Dorokhin, Artemov, Likhachev, Novikov1, & Starkov, 2020).

Las simulaciones se pueden clasificar también según su propósito, o según el tipo de vehículos que modelan. Una simulación intermodal incluye trayectos en los que se utilizan distintos vehículos o formas de transporte.

5.2.2 Componentes de la simulación

Los componentes principales de una simulación de tráfico son:

- Red: Define la infraestructura (calles, carreteras, intersecciones…) por la cual circularán los vehículos. Esta infraestructura puede estar definida simplemente por un conjunto de nodos y aristas, o incluir información más detallada como los carriles disponibles, conexiones entre carriles en las intersecciones, velocidades máximas y mínimas, vehículos permitidos, pendiente…

- Demanda (tráfico): Sobre esa infraestructura se describen trayectos de vehículos o flujos, dependiendo de la granularidad de la simulación. Estos pueden estar definidos por su punto de partida y de llegada, o por los ejes que se recorren, por ejemplo.

- Elementos adicionales: La simulación puede incorporar también elementos adicionales, como puntos de interés, configuraciones de los semáforos, presencia de señales de tráfico, etc.

5.2.3 Algoritmos para el cálculo de rutas

Algunos simuladores de tráfico permiten que el usuario defina un trayecto indicando el punto de partida y de llegada, calculando el simulador la ruta entre ambos. Para calcular la ruta, se utilizan algoritmos de búsqueda del camino más corto en grafos, ya que la red puede ser considerada como un grafo dirigido formado por nodos y conexiones entre ellos, con distintos pesos. Estos pesos pueden calcularse a partir de parámetros como la longitud de la arista, la velocidad permitida, o el tráfico observado en ella en el momento del cálculo.

Dos de los algoritmos utilizados para el cálculo de rutas son el algoritmo de Dijkstra y el algoritmo A*.

Algunas herramientas de simulación permiten también redirigir la ruta del vehículo dinámicamente dependiendo de las condiciones del tráfico o de la infraestructura.

5.3 Sistemas multi-agente

5.3.1 Definición

El concepto de programación orientada a agentes (AOP) surge en 1993 como una especialización de la programación orientada a objetos (OOP). Mientras que en la programación orientada a objetos el objeto es el elemento fundamental, que

encapsula datos y funcionalidad, en la programación orientada a agentes este elemento es el agente.

Un agente se define como una entidad presente en un contexto frente al que puede reaccionar, siendo capaz de actuar de forma racional, tomar decisiones independientemente, y de relacionarse y colaborar con otros agentes.

Un sistema multi-agente es el compuesto por varios agentes, su entorno y las relaciones entre estos.

Los sistemas multi-agentes resultan muy apropiados por sus características para ser utilizados en aplicaciones que requieran cálculos u operaciones distribuidos o concurrentes, o cuando se requiere que distintos componentes puedan comunicarse entre ellos. También son útiles en aplicaciones que utilizan mensajes distribuidos a través de una red y toman decisiones en base a estos.

Existen diversos lenguajes y *frameworks* especializados en la programación orientada agentes. Estos lenguajes implementan diversos modelos o estándares definidos para este tipo de programación (Cardoso & Ferrando, 2021).

Unos de los modelos más conocido es el modelo BDI (*Belief Desire Intention*). En él, los agentes presentan tres "actitudes mentales": creencias (información sobre el entorno), deseos (estados que el agente desea conseguir) e intenciones (secuencias de acciones planificadas para conseguir esos estados). Estas tres "actitudes mentales" del agente se expresan mediante una lógica formal multimodal (Rao & Georgeff).

El agente ejecuta un algoritmo con varios pasos de manera cíclica. En primer lugar, el agente actualiza sus creencias a partir de la información del entorno. Posteriormente elabora y selecciona sus deseos a partir de sus creencias actuales. A partir de las creencia y deseos se actualizan las intenciones disponibles y, mediante una función de selección, se escoge la próxima acción del agente (Cardoso & Ferrando, 2021).

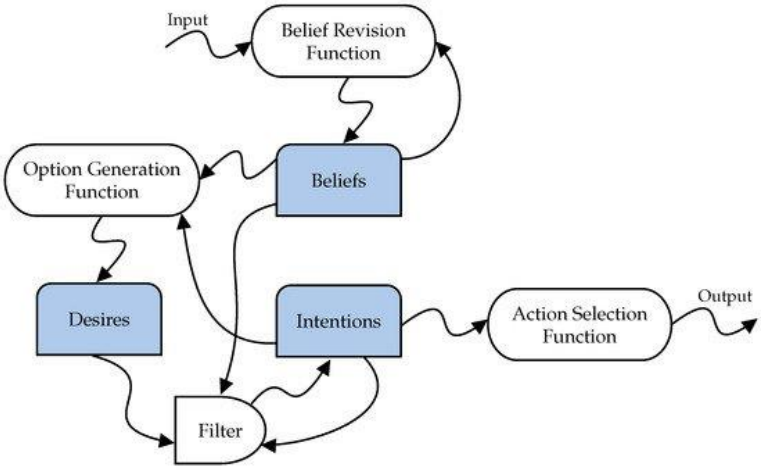

Figura 1. Esquema del ciclo del modelo BDI (Cardoso & Ferrando, 2021)

5.3.2 Estándares FIPA

Otra de las especificaciones más importantes implementadas por *frameworks* para la programación orientada a agentes son las especificaciones FIPA. Estas reciben su nombre de la organización FIPA *(Foundation for Intelligente Pysichal Agents*), que las promueve y regula, y que forma parte de la IEEE (*Institute of Electrical and Electronics Engineers*). En 2002 se estandarizó un conjunto de 25 de estas especificaciones. El objetivo de este proceso de estandarización es lograr la interoperabilidad, tanto entre diferentes implementaciones de estas especificaciones, como con otras tecnologías (Foundation for Intelligent Physical Agents, 2005).

FIPA (FIPA TC Agent Management, 2004) describe los servicios necesarios para la gestión de los agentes como componentes lógicos, sin indicar de qué forma se deben implementar estos. Estos componentes son:

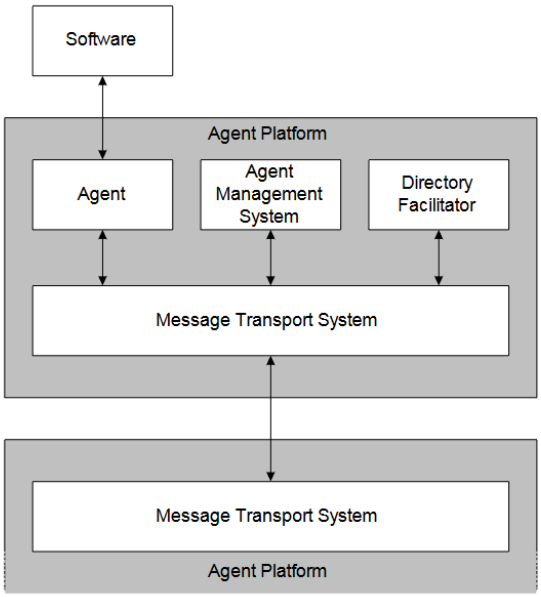

Figura 2: Servicios definidos por FIPA para la gestión de los agentes (FIPA TC Agent Management, 2004)

- Plataforma AP (Agent Platform): Infraestructura física en la que se despliegan los agentes. Está compuesta por la máquina o máquinas, sistema operativo, el software accesible por los agentes y los componentes FIPA descritos en esta sección. La plataforma puede estar distribuida en varias máquinas. FIPA describe el modo de comunicación entre plataformas, pero dentro de una misma plataforma se pueden utilizar las comunicaciones adicionales que se deseen.

- Sistema de gestión de agentes o AMS (*Agent Management System*): Servicio donde se registran los agentes para obtener un identificador en el sistema. Supervisa el acceso y uso de la plataforma.

- Directorio o DF (*Directory Facilitator*): Opcional e implementado como un servicio. Permite a los agentes registrar servicios y encontrar servicios ofrecidos por otros agentes.

- *Sistema de mensajería o MTS (Message Transport System)*: Gestiona la comunicación entre agentes de diferentes plataformas

- Agente: Cada uno de los procesos autónomos con capacidad de comunicación de la plataforma. Ofrece uno o más servicios en un modelo de ejecución integrado.

Cada agente está identificado por un AID (*Agent IDentifier*). Este está compuesto por un nombre, una lista de direcciones en las cuales recibe los mensajes y una lista de direcciones de resolución de nombres de servicios. El nombre es un identificador global único del agente se obtiene de la concatenación del nombre local con la dirección de la plataforma donde reside o HAP (*Home Address Platform*).

Los agentes tienen un ciclo de vida similar a una máquina de estados finitos. Pueden estar inicializados, activos, suspendidos o en espera. También pueden estar en tránsito, ya que los agentes tienen movilidad y pueden ser desplazados a otras plataformas o máquinas a través de protocolos de transporte (FIPA TC Agent Management, 2004).

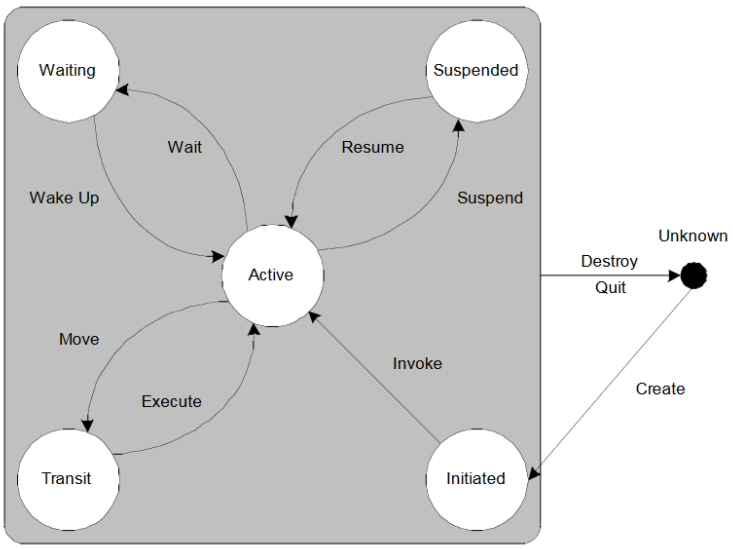

Figura 3. Esquema de estados de un agente (FIPA TC Agent Management, 2004)

5.3.3 Lenguaje ACL

El lenguaje ACL (*Agente Communication Language*) es un estándar definido por FIPA para la comunicación entre agentes mediante el intercambio de mensajes. Estos mensajes pueden formar parte de conversaciones. Este estándar (FIPA TC Communication, 2002) describe el formato de los mensajes, que deben implementar como mínimo los siguientes campos:

Campo	Categoría	Descripción	Obligatorio
Performativa*	Tipo de acto comunicativo	Especifica el acto comunicativo del mensaje	Sí
Emisor	Participante en la conversación	Identificador del agente que envía el mensaje	Recomendado
Receptor	Participante en la conversación	Identificador del agente al que va destinado el mensaje	Recomendado
Responder a	Participante en la conversación	Indica el agente al que se deben enviar los siguientes mensajes de la conversación	No
Contenido	Contenido	Contenido del mensaje o el objeto de la acción	Recomendado
Lenguaje	Descripción del contenido	Describe el lenguaje en el que se ha expresado el contenido del mensaje.	No
Codificación	Descripción del contenido	Indica la codificación utilizada en el contenido del mensaje.	No
Ontología	Descripción del contenido	Junto con el lenguaje permite interpretar el contenido del mensaje.	No
Protocolo	Control de la conversación	Describe el protocolo empleado por el agente emisor del mensaje. Si no es nulo, se considera que el mensaje pertenece a una conversación, y debe seguir unas reglas específicas.	No
Id de la conversación	Control de la conversación	Se utiliza para identificar a que conversación pertenece el mensaje y distinguir entre conversaciones concurrentes, o restaurar una conversación histórica.	No
Responder con	Control de la conversación	Expresión que debe ser usada por el agente receptor para identificar el mensaje.	No
En respuesta a	Control de la conversación	Expresión que referencia un mensaje anterior al cual el mensaje responde. Los dos últimos campos se utilizan como elemento de diferenciación.	No

Responder antes de	Control de la conversación	Indica la fecha o instante antes del cual el emisor desea recibir una respuesta.	No

Tabla 1: Campos de los mensajes ACL

* Performativas: FIPA define un conjunto de macros para especificar cuál es la intención del mensaje, o acto comunicativo. Algunas de estas performativas son: *REQUEST, PROPOSE, INFORM, REQUEST_WHEN, INFORM_IF ACCEPT_PROPOSAL, REJECT_PROPOSAL, CONFIRM, CANCEL, FAILURE*… (FIPA TC Communication, 2002) La semántica de cada acto comunicativo se ha formalizado utilizando la lógica modal de los modelos BDI, ya que esta permite expresar distintos grados de confianza o condiciones (Poslad, 2007).

FIPA también describe en varias de sus especificaciones las formas estandarizadas de codificar los mensajes para su distribución. Cada especificación de representación de mensajes ACL (XML, cadenas de texto…) indica con precisión cómo debe ser la sintaxis (FIPA TC Communication, 2002).

Existen también especificaciones de protocolos que definen el patrón de actos comunicativos (performativas) requeridos para desarrollar una interacción concreta. Algunos ejemplos son el protocolo de suscripción o el protocolo *Contract Net* (FIPA TC Communication, 2002).

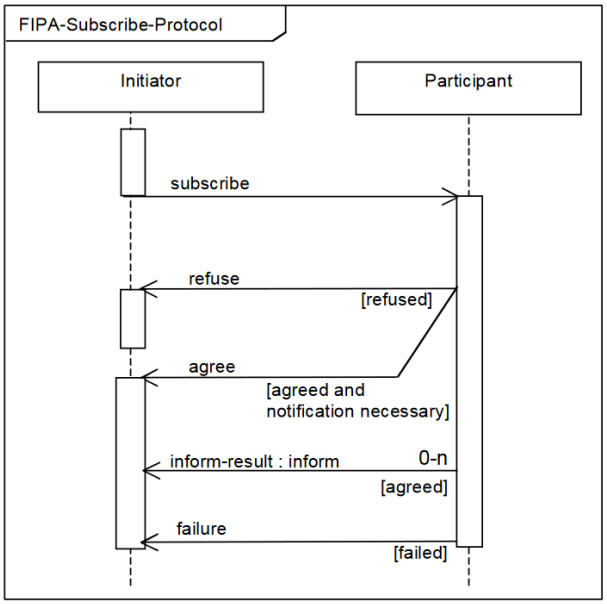

Figura 4: Protocolo de suscripción (FIPA TC Communication, 2002)

5.4 Machine learning

El aprendizaje automático o *Machine Learning* forma parte de un área más amplia denominada inteligencia artificial, que pretende dotar de inteligencia a los ordenadores o máquinas. Entre sus objetivos están la replicación de las capacidades humanas y la extensión de estas capacidades. En concreto se refiere a la detección automática de

patrones en los datos. En las últimas décadas se ha convertido en una herramienta esencial para extraer información de grandes conjuntos de datos, o para tareas donde la complejidad de los patrones es elevada.

La finalidad del *Machine Learning* es dotar a los ordenadores de la capacidad de aprender a realizar tareas mediante el entrenamiento, del mismo modo que el ser humano aprende mediante la experiencia.

En el caso del aprendizaje automático, la experiencia proviene de un conjunto de datos, con los cuales el algoritmo es entrenado para realizar una tarea, produciendo un modelo que representa el conocimiento adquirido sobre la realización de dicha tarea. Esta consiste en muchos casos en la predicción de una variable a partir de otras. El modelo producido se puede utilizar posteriormente sobre datos nuevos.

La utilidad de estos modelos se puede evaluar mediante el uso de casos de prueba extraídos del conjunto de datos y no utilizados en el entrenamiento.

Existe una distinción importante entre aprendizaje supervisado y no supervisado. En el supervisado se proporcionan en el entrenamiento las variables que el modelo debe predecir (etiquetas), y que no estarán incluidos tanto en los casos de prueba como en la aplicación real del modelo. El objetivo del modelo es por tanto aprender a predecir estas etiquetas.

En el aprendizaje no supervisado, las variables a predecir no se proporcionan en el entrenamiento, y el objetivo es obtener el esquema subyacente a los datos, por ejemplo, la división de los datos en diferentes conjuntos o *clusters*.

Dentro del aprendizaje supervisado existen dos categorías:

- Clasificación: La variable a predecir es discreta. Los datos se clasifican en varias categorías y el objetivo es asignar la categoría correcta a cada dato del conjunto de pruebas.

- Regresión: La variable a predecir es continua. El objetivo es obtener el valor más cercano al valor real de la variable.

Para los propósitos de este proyecto, nos centraremos en los modelos de regresión. En los siguientes apartados se explicarán los métodos utilizados para evaluar estos modelos y los distintos algoritmos de aprendizaje que se han utilizado en este proyecto.

5.4.1 Evaluación del modelo

A diferencia de los modelos de clasificación, donde la predicción es o no correcta en función de si se ha acertado la categoría de los datos, en los modelos de regresión evaluar la corrección de la predicción es una tarea más compleja.

Por ejemplo, si el valor correcto es 3, no es lo mismo predecir 5 que 3,1. En el primer caso el error es de 2 puntos mientras que en el segundo es de 0,1 puntos.

Consideraremos mejor un modelo que se equivoque en una décima, que uno que se equivoque en varias unidades.

Por tanto, para evaluar el modelo se debe utilizar un método que penalice la discrepancia entre el valor real y el calculado.

Uno de los métodos más utilizados es *R-squared*. Este método compara cuánto mejor es el modelo que hemos entrenado respecto a tomar simplemente la media de los valores como predicción.

En primer lugar, se calcula la diferencia de cada valor real con la media y se eleva al cuadrado para evitar que valores positivos y negativos se cancelen entre sí. Se suman todos los valores. Denominaremos al resultado obtenido variación de la media.

A continuación, se calcula la diferencia de cada valor real con la predicción realizada (denominado residual) y se eleva también al cuadrado. Se suman todos los valores. El resultado obtenido es la variación del modelo.

Por último, se utiliza la siguiente fórmula:

$$R^2 = \frac{Var(media) - Var(modelo)}{Var(media)}$$

R^2 es un valor entre 0 y 1. Cuánto más se aproxima a 1, mejor es el modelo respecto a la media.

El resultado de R^2 nos proporciona información sobre lo bien que se ajusta el modelo a los datos, métrica denominada sesgo. Sin embargo, también es conveniente considerar la varianza del modelo, es decir, la precisión en las predicciones cuando se utilizan otros datos. Un modelo con un sesgo alto puede presentar un problema denominado *overfitting.* Este problema se produce cuando el modelo es demasiado específico para los datos sobre los que se ha entrenado.

5.4.2 Regresión lineal y polinómica

El objetivo de los modelos de regresión (no confundir con regresión como categoría de modelos donde la variable a predecir es continua) es encontrar una ecuación que describa los datos. El caso más simple es la regresión lineal, donde la ecuación es lineal.

La regresión lineal simple encuentra la relación entre dos variables, *y* es la variable independiente y *x* la variable dependiente, mediante una ecuación lineal:

$$y = b_0 + b_1 x$$

La regresión lineal multivariable incluye varias variables independientes.

El entrenamiento consiste en la modificación de los coeficientes de la ecuación para que el error sea el menor posible. Si visualizamos la ecuación como una línea el caso ideal sería que todos los puntos se encontrasen dentro de la línea.

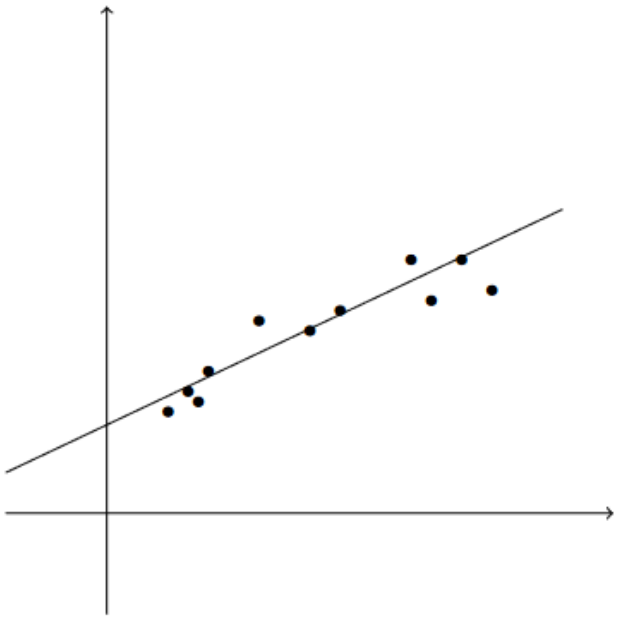

Figura 5: Regresión lineal (Shalev-Shwartz & Ben-David, 2014)

Para ajustar la línea a los datos se utiliza el método *Least Squares*, que intenta minimizar el error conjunto. Para ello se calcula el error de cada predicción y se eleva al cuadrado para evitar que los puntos positivos y negativos se cancelen entre sí. Después se obtiene la media de estos valores. Se escogerán los coeficientes que minimicen este valor.

$$LS\ (h) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

La regresión polinómica es similar a la regresión lineal, pero en lugar de una ecuación lineal se busca una ecuación polinómica que describa los datos.

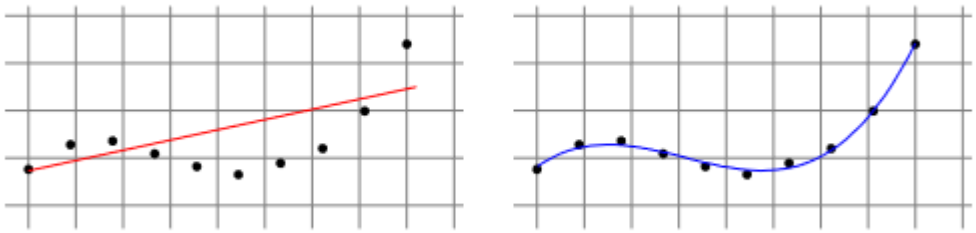

Figura 6: Regresión polinómica (Shalev-Shwartz & Ben-David, 2014)

5.4.3 K Nearest Neighbors

K Nearest Neighbors o kNN es un algoritmo que se basa en el almacenamiento de todos los casos entrenados y la predicción de los nuevos casos a partir de la similitud con estos.

Este método se basa en la creación de un espacio n-dimensional donde cada dimensión representa un atributo, y cada observación está representada por un punto. Cuando se realiza una predicción, se recrea el punto que la nueva observación ocuparía en este espacio y se calcula la distancia entre este y el resto de los puntos.

A continuación, se escogen los puntos vecinos más cercanos a la observación. El número de puntos escogidos depende del parámetro *k.* La predicción es la clase mayoritaria entre los puntos escogidos en el caso de los modelos de clasificación. En los modelos de regresión es la media, ponderada con la distancia o no.

En los modelos de clasificación es conveniente que *k* sea un valor impar para que no se produzcan empates.

La ventaja de este método es que el entrenamiento es poco costoso, ya que solo requiere añadir los puntos al espacio. Sin embargo, la realización de las predicciones requiere de un mayor número de cálculos.

5.4.4 Support Vector Machine

Aunque este algoritmo se utiliza de manera más frecuente para modelos de clasificación (SVC o *Support Vector Classification*), también se puede utilizar para entrenar modelos de regresión (SVR o *Support Vector Regression*).

Este método se asemeja al método anterior en la representación de los datos en un espacio n-dimensional. Sin embargo, el objetivo de este método es encontrar un hiperplano que separe el espacio, clasificando los puntos de manera distintiva. En el caso del SVR, se utiliza el hiperplano que contiene más puntos como ajuste de los datos, en vez de como separador.

Los puntos más cercanos al hiperplano se denominan *Support Vectors,* y dan nombre al algoritmo. Estos puntos tienen una gran influencia en la posición y orientación del hiperplano.

5.4.5 Metodos híbridos y ensemble: Adaboost

Para mejorar los resultados de los modelos, frecuentemente se utilizan métodos híbridos o *ensemble*. Estos métodos se benefician del uso de varios modelos. En el caso de los métodos híbridos se utiliza una combinación de modelos, por ejemplo, se realiza la selección de atributos mediante un modelo y estos se entrenan mediante otro.

En el caso de los métodos *ensemble* se entrenan varios modelos y se combinan sus resultados. Por ejemplo, se puede realizar una votación entre los distintos resultados. Los modelos utilizados pueden ser del mismo tipo o de distintos tipos y pueden ser entrenados con los mismos datos o con datos diferentes.

Adaboost es un método *ensemble* cuyo nombre proviene de *Adaptative Boosting.* Los métodos que realizan *Boosting* se caracterizan por combinar un gran número de modelos débiles, cuyo rendimiento es ligeramente mejor que realizar predicciones al azar, para obtener resultados precisos. La parte de adaptativo proviene de la capacidad de Adaboost de asignar mayor peso en el entrenamiento a las observaciones más difíciles de predecir.

5.5 Sistemas distribuidos

Un sistema distribuido es un conjunto de ordenadores que trabajan de forma conjunta o se relacionan entre sí. Para ello son necesarios protocolos y definiciones arquitectónicas que permitan establecer una comunicación efectiva. En este apartado se definen algunos conceptos para la comprensión de las tecnologías utilizadas para la cooperación de los distintos componentes del sistema, que pueden ser distribuidos en diferentes máquinas.

5.5.1 APIS

Una API (RedHat, s.f.) (Application Programming Interface) es un conjunto de protocolos y definiciones utilizados para integrar una aplicación con otros sistemas. A veces se define como un contrato entre el proveedor y el usuario, que define el formato y contenido de las peticiones y las respuestas

5.5.2 REST

REST (RedHat, s.f.) (REpresentational State Transfer) es un conjunto de especificaciones arquitectónicas para la creación de APIS, que se pueden implementar de diversas formas.

- En la arquitectura REST el cliente utiliza peticiones (*requests*) sobre recursos. Un recurso es una información almacenada en el lado del servidor. Al recibir la petición, el servidor envía una representación del estado actual del recurso (Representation). Esta representación puede tener diferentes formatos: texto plano, XML, JSON… El cliente también puede actualizar o crear un recurso en el servidor.

- Las peticiones deben comunicar toda la información necesaria, en el servidor no se guarda la información del cliente entre peticiones (Stateless)

- Las representaciones obtenidas son cacheables

- Se utiliza términos HTTP para identificar las acciones sobre el recurso (GET, POST, PUT, DELETE…)

- Las peticiones se realizan mediante URIs y *headers.* Una URI es una secuencia de caracteres que identifica el recurso. Las URIs pueden contener parámetros

- La interfaz entre el servidor y el cliente debe estar bien definida, y la información intercambiada debe ser entendible por sí misma.

- La respuesta al cliente puede contener hipervínculos para acceder a la información.

Una API que implementa todas las especificaciones arquitectonicas de REST se considera RESTful.

5.5.3 Estándar JSON

JSON (*JavaScript Object Notation*) es un formato ligero de intercambio de datos, fácil de entender y escribir por humanos.

Este formato se basa en una subespecificación del *JavaScript Programming Language Standard* de diciembre de 1999. A pesar de contener JavaScript en sus iniciales, JSON es independiente del lenguaje.

JSON se construye mediante estructuras que resultan familiares si se conocen lenguajes como Python o C.

Un objeto en JSON es una colección de pares nombre/valor, similar a un diccionario o a una tabla *hash* o *array* asociativo. Se describe entre paréntesis, separando los nombres de los valores mediante dos puntos. Los distintos pares se separan mediante comas.

6 METODOLOGÍA

Al iniciar el proyecto se observó que las características de este no se ajustaban a las del desarrollo de una aplicación tradicional:

- Diferentes componentes muy distintos entre sí: El sistema planteado unifica varias tecnologías distintas con procesos de desarrollo muy distintos.

- Alto componente de investigación.

- Ámbito del proyecto: El ámbito del proyecto es la realización del sistema de recomendación y su evaluación. No se han realizado por tanto interfaces gráficas para el usuario, y no se han incluido algunas finalidades que estarían presentes si se continuase con el desarrollo del sistema propuesto para su implementación y utilización en la realidad.

Estas características motivaron la decisión de rechazar el uso de metodologías rígidas y orientadas a objetos que se habían valorado en un principio, como el Proceso Racional Unificado (RUP), y se decidió utilizar metodologías diferentes adaptadas a cada componente.

Los componentes en los que se puede dividir el proyecto son:

- Sistema multi-agente para el balanceado del sistema

- Predicción de la demanda mediante *Machine Learning.*

- Simulación del uso del sistema mediante SUMO

En el caso del sistema de balanceado, se ha decidido utilizar una metodología orientada al desarrollo de sistemas multi-agente. Se valoró en uso de diferentes metodologías, por ejemplo, MESSAGE (Kearney, y otros, 2001). Finalmente se decidió utilizar Gaia, ya que, al estar enfocado a modelar los roles y la comunicación entre los agentes, se adaptaba muy bien a este caso.

En el caso de la predicción de la demanda se ha utilizado la metodología propuesta por Microsoft para el entrenamiento de modelos de *Machine Learning* (Amershi, y otros, 2019).

En el caso de la simulación de uso del sistema, en el plan del proyecto se ha tratado como una prueba del funcionamiento del sistema, aunque la interfaz y los algoritmos se han desarrollado durante la fase de diseño.

Por último, se ha dado gran importancia a la conexión entre los distintos componentes y a la arquitectura del sistema en todas las fases del proyecto. Esta se ha representado en los diagramas de arquitectura.

6.1 Gaia: Metodología orientada a agentes

Gaia (Wooldridge, Jennings, & Kinny, 2000) es una metodología para el desarrollo de sistemas orientados a agentes. Es una metodología general, aplicable a diversos tipos de sistemas multi-agente. Modela el sistema tanto a nivel macroscópico (relaciones entre los agentes y con el entorno) como microscópico (servicios y funcionalidades de cada agente). La visión de los sistemas multi-agente de Gaia es la de una organización consistente por varios roles.

Se ha escogido esta metodología ya que la visión del sistema que se pretende desarrollar encaja muy bien con la visión de Gaia de los sistemas multi-agente.

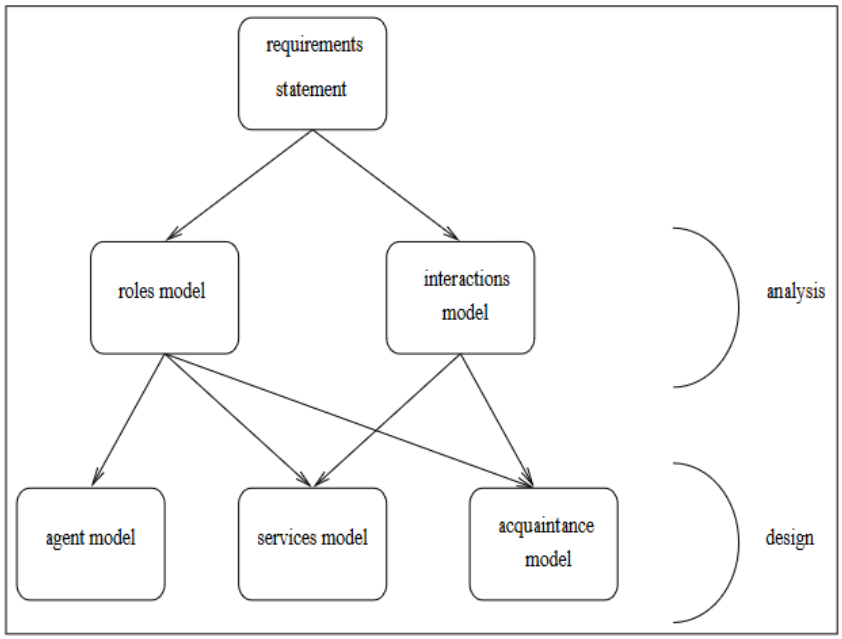

Figura 7: Documentos de la metodología Gaia (Wooldridge, Jennings, & Kinny, 2000)

6.1.1 Captura de requisitos

La metodología Gaia define una fase de captura de requisitos, pero no la metodología a utilizar e indica que se escoja la opción que se valore mejor para esta. Se ha decidido utilizar la Metodología para la Elicitación de Requisitos de Sistemas Software de Durán y Bernárdez (Durán Toro & Bernárdez Jiménez, 2000). Además, en esta fase se han incluido los requisitos del proyecto completo para tener un documento de referencia de los objetivos a cumplir por el sistema.

6.1.2 Análisis

El objetivo de la fase de análisis es entender el sistema y su estructura. En este caso se pretende capturar la organización del sistema, entendida como una colección de roles con ciertas relaciones entre sí.

En la fase de análisis se generan dos documentos:

- Modelo de roles: Identifica los roles principales presentes en el sistema. Estos roles son descripciones abstractas de las funcionalidades esperadas por una entidad. Los roles tienen responsabilidades y permisos. Las responsabilidades pueden ser a su vez condiciones del sistema que mantener o acciones que se deben realizar.

- Modelo de interacciones: Identifica las relaciones entre los distintos roles mediante protocolos

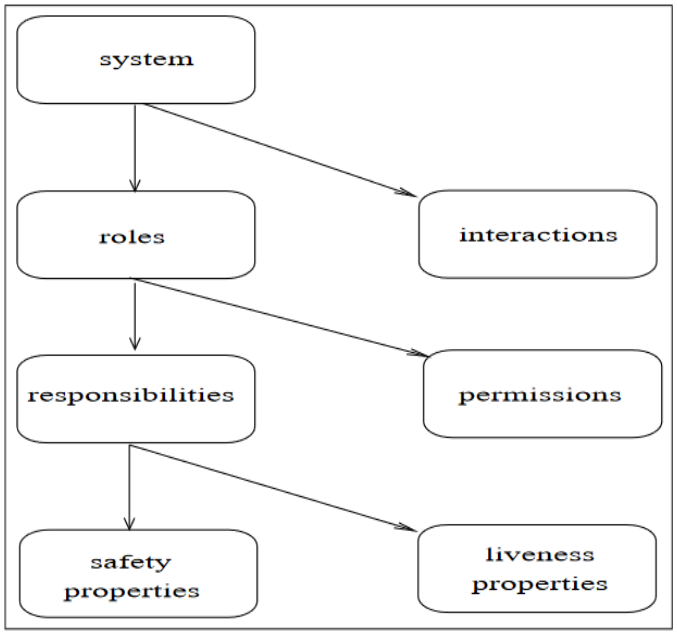

Figura 8: Análisis según la metodología Gaia (Wooldridge, Jennings, & Kinny, 2000)

6.1.3 Diseño

El objetivo de la fase de diseño es transformar los modelos de análisis a un nivel bajo de abstracción para poder implementarlos. En concreto, Gaia modela cómo una sociedad de agentes coopera para conseguir los objetivos globales del sistema y las funciones de cada individuo para lograr estos objetivos.

En la fase de diseño se generan tres modelos:

- Modelo de agentes: Identifica los tipos de agentes y el número de instancias de estos.

- Modelo de servicios: Identifica los servicios principales que se requieren para realizar las funciones del agente.

- Modelo de conocidos: Documenta las líneas de comunicación entre los distintos agentes.

6.2 Metodología para la predicción de la demanda

Para la predicción de la demanda se ha utilizado la metodología descrita por Microsoft en (Amershi, y otros, 2019).

Esta es una metodología experimental basada en los datos. Mientras que en el desarrollo de aplicaciones tradicionales el código es el elemento principal, en el desarrollo de modelos de aprendizaje los datos adquieren toda la importancia.

Otro aspecto que se tiene en cuenta es la dificultad para mantener límites claros entre distintos módulos o componentes, ya que los diferentes procesos necesarios para el entrenamiento de modelos pueden ser complejos y estar entrelazados entre ellos.

La metodología consta de varias fases, algunas de ellas orientadas a los datos y otras al entrenamiento del modelo.

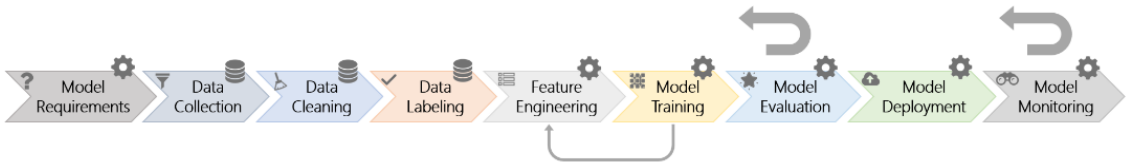

Figura 9: Fases de la metodología propuesta por Microsoft (Amershi, y otros, 2019)

La metodología identifica las siguientes fases:

- Definición de requisitos del modelo: Se analiza la tarea a resolver y se decide qué modelos son más apropiados.
- Recolección de datos: Se buscan colecciones de datos apropiadas ya disponibles o se crean.
- Limpieza de los datos: Se eliminan los errores e inexactitudes que puedan contener los datos.
- Etiquetado de los datos: Esta fase se realiza principalmente en el aprendizaje supervisado. Consiste en asignar las etiquetas apropiadas a los datos.
- *Feature Engineering:* Se extraen y seleccionan los atributos que se utilizarán en el entrenamiento.
- Entrenamiento de los modelos: Se entrenan los modelos seleccionados con los atributos escogidos para los datos preparados.
- Evaluación de los modelos: Se utilizan métricas para evaluar los modelos entrenados sobre conjuntos de datos de prueba.
- Despliegue de los modelos: Se despliega el modelo en la máquina o aplicación para la cual ha sido diseñado.

- Monitorización de los modelos: Una vez desplegado el modelo, este se monitoriza de forma continua para detectar y solucionar posibles errores.

El proceso es secuencial, pero existen dos bucles principales de realimentación. Tras la evaluación de los modelos o como parte de la monitorización del modelo se puede volver a cualquiera de las fases, para mejorar los resultados. Existe también un bucle de realimentación menor en la fase de entrenamiento del modelo a la fase de *Feature Engineering.*

El proceso de entrenamiento de los modelos de predicción siguiendo estas fases se refleja en el apartado de desarrollo de la solución, exceptuando la fase de etiquetado y limpieza de los datos, que no ha sido necesaria, ya que los datos no contenían errores al ser generados de manera automática y ya estaban etiquetados.

6.3 Planificación y fases del proyecto

La planificación general del proyecto se ha dividido en 8 fases, siguiendo un esquema similar a la metodología de cascada. Dentro de esta planificación se han integrado las distintas fases de las metodologías utilizadas, permitiendo la realización de revisiones o iteraciones de acuerdo con las necesidades del proyecto y las distintas metodologías.

- Investigación: La primera fase del proyecto ha consistido en la investigación sobre el problema a tratar y las herramientas disponibles, sistemas externos que se podían utilizar y trabajos relacionados.

- Captura de requisitos: Se ha realizado la elicitación de todos los requisitos del sistema.

- Análisis: Se ha realizado la fase de análisis de la metodología Gaia, y se han analizado los requisitos del modelo de predicción de la demanda, decidiendo qué modelos se iban a utilizar y con qué características.

- Diseño: Se ha realizado la fase de diseño de la metodología Gaia, y el diseño de la interfaz y algoritmos de la simulación.

- Implementación: Se han implementado e integrado los distintos componentes del sistema.

- Pruebas: En la fase de pruebas se ha comprobado el funcionamiento individual de los distintos componentes y la buena integración entre ellos. Se han realizado también las simulaciones y se han obtenido los resultados.

- Despliegue: Se ha preparado el sistema para su despliegue y entrega.

- Documentación: Se ha realizado la documentación a entregar junto con el proyecto.

El proyecto se ha realizado en un plazo de 8 meses.

7 HERRAMIENTAS

En este apartado se detallan las herramientas escogidas para la realización de este trabajo y las razones de su elección. En muchos casos se ha optado por utilizar las herramientas más populares entre las disponibles por dos motivos. En primer lugar, porque, al ser utilizadas por más desarrolladores, reciben más *feedback* de errores, volviéndose más robustas. En segundo lugar, y por los mismos motivos, porque hay sobre ellas más documentación disponible. También se ha preferido utilizar herramientas gratuitas y *open source*.

7.1 Simulación: Herramientas del paquete SUMO (v1.13.01)

Para la simulación del funcionamiento del sistema se ha utilizado el paquete de herramientas SUMO (Eclipse Foundation, s.f.). Este paquete obtiene su nombre de su herramienta principal, el simulador de movilidad urbana. Junto a él se incluyen distintas herramientas, librerías y programas que aportan distintas funcionalidades para el tratamiento de los mapas, modelado de la demanda o conexión con distintos sistemas, por ejemplo.

En la conceptualización del proyecto se optó por utilizar estas herramientas, ya que, como parte de la beca de colaboración realizada en el curso 2019-2020, tuve la oportunidad de analizar diferentes simuladores. Esta experiencia me permitió determinar las siguientes ventajas de SUMO para este proyecto concreto:

- Popularidad frente a otros simuladores. SUMO es uno de los simuladores más utilizados, junto con Transsim y Vissim. Este último es un sistema comercial, mientras que SUMO es gratuito, *open source*. y además ofrece compatibilidad con Vissim.

- Propósito general: Las herramientas no tienen un propósito especifico, por lo que se pueden adaptar a nuestro ejemplo con mayor facilidad.

- Microscópico y multimodal: SUMO permite establecer el comportamiento de cada vehículo de la simulación individualmente y proporciona distintos tipos de vehículos, incluso personalizados. La herramienta también permite modelar bicicletas, peatones y personas utilizando distintos medios de transporte, incluso transporte público.

- Rendimiento con grandes redes: SUMO permite realizar simulaciones en mapas con un elevado número de nodos y conexiones.

- Herramientas adicionales: Las herramientas adicionales que incluye SUMO pueden ser de utilidad en la creación de la simulación. Por ejemplo, incluye una herramienta para descargar y convertir mapas de OSM (*Open Street Maps)*.

- SUMO permite conectarse a la simulación desde nuestro propio código y modificarla dinámicamente. Este factor ha motivado en gran medida el optar por esta herramienta, ya que en nuestra simulación las rutas dependen de si los usuarios aceptan las propuestas del sistema, y estas propuestas dependen a su vez de los trayectos anteriores de otros usuarios.

Las herramientas incluidas en el paquete SUMO que se han utilizado para preparar la simulación o para ejecutarla son las siguientes:

7.1.1 SUMO

Simulador de tráfico microscópico multimodal. Se ha utilizado para simular los trayectos de los usuarios en bicicleta o a pie, obteniendo los resultados de distancia recorrida. Recibe como entrada una configuración donde se especifican el mapa o red, la demanda de tráfico y los elementos adicionales, como la configuración de los semáforos o los puntos de interés del mapa. El cálculo de las rutas se realiza mediante el algoritmo de Dijkstra. Se puede ejecutar la simulación mediante comandos de terminal o utilizando la interfaz gráfica, SumoGui.

7.1.2 TraCI (Traffic Control Interface)

Biblioteca proporcionada por SUMO para conectarse a la simulación desde nuestro propio código siguiendo un modelo cliente-servidor. Existen diferentes implementaciones para distintos lenguajes de programación, pero se ha optado por utilizar la de Python, debido a que se ha valorado que la documentación disponible era de mejor calidad.

7.1.3 Otras herramientas

- SumoGui: Interfaz gráfica de sumo. Se ha utilizado para visualizar la simulación. Permite entre otras opciones ajustar la velocidad de la simulación, además de ejecutarla o pararla. También permite cambiar algunos ajustes de visualización

- OSMWizard: Herramienta web que permite descargar mapas de *Open Street Maps* y convertirlas en configuraciones de SUMO. Se ha utilizado para descargar los mapas de la ciudad de Salamanca con el propósito de utilizarlos en la simulación.

- Netedit: Herramienta gráfica que permite modificar o ajustar mapas. Se ha utilizado para comprobar de manera visual los errores existentes en los mapas descargados.

- Netconvert: Herramienta de terminal que permite realizar cambios en la configuración del mapa o en algunos parámetros. Esta herramienta permite también solucionar algunos errores detectados en el mapa a través de comandos.

- Netgenerate: Herramienta que permite generar mapas simples o aleatorios a partir de distintos parámetros introducidos. Se ha utilizado para generar mapas sencillos con los que trabajar en la simulación.

- Netcheck: Herramienta que permite comprobar la existencia de ejes desconectados en el mapa. Se ha utilizado para analizar los mapas obtenidos de OSM.

7.2 Sistema de balanceado

Tras un análisis de los *frameworks* disponibles para sistemas multi-agente, escogí utilizar JADE al estar más familiarizada con su uso. Al estar este implementado en Java, el resto de las herramientas utilizadas para este componente se escogieron en base a su disponibilidad para Java, entre otros criterios.

7.2.1 *Framework* JADE (v4.5.0)

El *framework* JADE (JADE, s.f.) para el desarrollo de sistemas multi-agente está formado por un *middle-ware* basado en las especificaciones FIPA y un conjunto de herramientas gráficas que ayudan en las fases de desarrollo y depuración. Por ejemplo, una de estas herramientas permite visualizar los agentes presentes en el sistema y añadir, eliminar, suspender o realizar diversas acciones sobre ellos. Adicionalmente, permite observar las comunicaciones entre los agentes de forma gráfica, mediante la introducción de un agente *sniffer*.

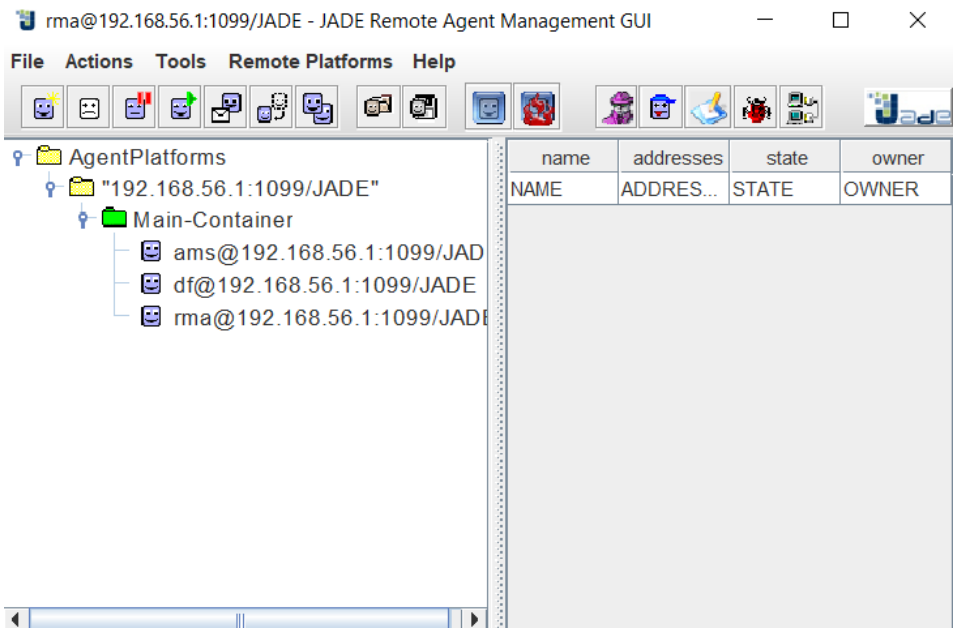

Figura 10: Monitorización de los agentes mediante una herramienta gráfica de JADE

Los agentes se despliegan en una plataforma formada por varios contenedores que se pueden distribuir en varias máquinas. Los agentes pueden ser creados, eliminados o desplazados entre contenedores durante la ejecución. JADE proporciona clases que

se pueden extender para crear agentes con distintos comportamientos que se comunican entre ellos mediante el lenguaje ACL.

7.2.2 Servlet REST

Para la comunicación entre el usuario y el sistema de balanceado se han utilizado las siguientes tecnologías:

- Jersey (v2.36): *Framework* para desarrollar servicios web RESTful basados en la especificación JAX-RS, que propone una serie de indicadores e interfaces para definir los recursos expuestos (Eclipse Jersey, s.f.). Se decidió utilizar Jersey ya que es *open source,* y se había utilizado previamente en la carrera en la asignatura de Sistemas Distribuidos.

- Apache Tomcat (v9.0.58): Implementación de *Jakarta Servlet,* que se utiliza como contenedor de *servlets* (The Apache Software Foundation, s.f.). Se ha utilizado para alojar el sistema de balanceado. Se ha escogido por ser *open source*, y por su compatibilidad con Jersey.

7.3 Predicción de la demanda

7.3.1 Librerías ML

- Pandas: Biblioteca para la manipulación y análisis de datos (Pandas, s.f.). Es una biblioteca ampliamente utilizada en el campo del aprendizaje automático. Resulta especialmente conveniente la estructura de datos DataFrame que ofrece. Se ha utilizado para la limpieza y análisis de los datos, tanto para la predicción de la demanda como para la simulación.

- Matplotlib: Biblioteca para crear gráficos y visualizaciones (Matplotlib, s.f.). Se ha utilizado para crear y retocar visualmente la mayor parte de los gráficos del proyecto.

- Seaborn: Biblioteca basada en Matplotlib que permite crear visualizaciones más atractivas visualmente (Waskom, s.f.). Se ha utilizado para crear alguno de los gráficos más complejos.

- Scikit-Learn: Biblioteca para *machine learning* (Scikit-learn, s.f.) que implementa los modelos que se han utilizado para la predicción de la demanda.

- Yellowbrick: Extiende la biblioteca Scikit-Learn para facilitar la selección del modelo más apropiado. Utiliza visualizadores que encapsulan el modelo de Scikit-Learn y utilizan gráficos generados mediante Matplotlib para mostrar las características o la eficiencia de este (Yellowbrick, s.f.). Se ha utilizado para evaluar los modelos.

7.3.2 Servlet REST

- Request: Biblioteca que permite realizar peticiones HTTP de manera sencilla (Python Software Foundation, s.f.). Se ha utilizado para obtener información de la API AEMET OpenData

- Flask: Framework minimalista basado en la especificación WSGI para crear aplicaciones web rápidamente (Pallets, s.f.). Se ha utilizado junto a su extensión Flask-RESTful para crear el servlet REST de respuesta a las peticiones de demanda. Se ha escogido esta herramienta debido a su fácil aprendizaje y a su sencillez para crear recursos.

7.4 Lenguajes de programación

7.4.1 Java

Java (Oracle, s.f.) es un lenguaje de alto nivel orientado a objetos que ofrece una gran compatibilidad para desarrollar aplicaciones y ejecutarlas en máquinas con distintas especificaciones, ya que el código se compila para ser ejecutado por una máquina virtual (*Java Virtual Machine*) que actúa como *middleware*. Esta es una de las razones de su popularidad. Fue desarrollado por Sun Microsystems en 1995 y es todavía en la actualidad uno de los lenguajes más utilizados.

La elección de este lenguaje de programación para este proyecto estuvo condicionada por la elección de JADE como *framework* para la implementación del sistema multi-agente.

7.4.2 Python (v3.9)

Python (Python Software Foundation, s.f.) es un lenguaje de alto nivel e interpretado. Es un lenguaje orientado a objetos, aunque también permite la programación procedural y funcional. Una de sus características es que es débilmente tipado, lo que le confiere dinamismo. También proporciona al programador estructuras de datos de muy alto nivel y versátiles. Como desventaja, su ejecución es más lenta que la de otros lenguajes compilados. Entre sus aplicaciones está el prototipado rápido de aplicaciones, ya que es fácil de aprender y utilizar. También es un lenguaje muy utilizado para realizar *scripts* debido a que, al ser un lenguaje interpretado, el código se puede ejecutar línea por línea.

Se decició utilizar este lenguaje para la predicción de la demanda debido a que es un lenguaje muy utilizado en el campo del *Machine Learning,* debido a la facilidad para trabajar con sus estructuras de datos ya mencionadas y a que existen para él bibliotecas muy potentes en esta área. Además, se pueden incorporar nuevos módulos muy fácilmente utilizando el gestor de paquetes pip. También se consideró la facilidad para implementar de manera rápida un servicio REST mediante Flask.

7.4.3 R

Durante las primeras etapas de la realización del proyecto, se utilizó el lenguaje R (R Foundation, s.f.) para la limpieza y visualización de los datos. Una de las ventajas de R es su facilidad para trabajar con tablas. Finalmente se decidió descartar el uso de este lenguaje, para mejorar la mantenibilidad del sistema y mantener su simplicidad.

7.5 Lenguajes de marcado

7.5.1 XML

XML es un lenguaje de marcado que permite definir etiquetas propias para describir e intercambiar información sobre un elemento.

SUMO utiliza este lenguaje para describir las configuraciones de las simulaciones, las redes que forman los mapas, las rutas y la demanda del tráfico… y prácticamente todos los elementos que participan en la simulación.

Se ha utilizado este lenguaje para definir la simulación, tanto los vehículos como los POI (puntos de interés) de las estaciones, además de la configuración general y los ajustes de visualización.

7.6 Entornos de desarrollo

7.6.1 Eclipse (v4.17.0)

Eclipse (Eclipse Foundation, s.f.) es un entorno de desarrollo principalmente orientado a Java, aunque actualmente soporta otros lenguajes de programación. Esta desarrollado por la Eclipse Foundation, cuyo objetivo es promover la colaboración en la creación de software *open source.* Ofrece las funcionalidades básicas de un entorno para programación y permite añadir una gran cantidad de módulos que ofrecen otras añadidas, como es el caso de los módulos para programación de aplicaciones web dinámicas. La interfaz permite también cambiar de perspectiva para adaptarse al tipo de desarrollo que se está realizando.

Este entorno se ha escogido por su compatibilidad con el proyecto y las herramientas utilizadas, por ejemplo, Jersey, que también ha sido desarrollado por la Eclipse Foundation.

7.6.2 PyCharm Community Edition (v2021.3.2)

PyCharm (JetBrains, s.f.) es un entorno de desarrollo inteligente para Python creado por JetBrains, una empresa especializada en la creación de IDEs para diferentes lenguajes de programación. Entre las ventajas valoradas para su elección destacan el

buen rendimiento y funcionamiento de sus herramientas para autocompletar, sugerir y detectar posibles errores en el código.

7.6.3 Sublime Text

Sublime Text (Sublime, s.f.) es un editor de textos especializado en la edición de código. Se ha utilizado principalmente para la creación y modificación de archivos xml y csv. Las razones para escoger este editor han sido su interfaz, que resulta muy cómoda, y los múltiples funcionalidades y atajos que ofrece para editar texto con rapidez, por ejemplo, mediante el uso de multicursores, que permiten editar varias líneas a la vez.

7.6.4 RStudio (v2022.02.0)

RStudio (RStudio, s.f.) es un entorno de desarrollo para R. Tanto el lenguaje R como este entorno están especializados en el tratamiento de grandes conjuntos de datos. La ventaja de este entorno es que su interfaz se encuentra dividida y se puede visualizar el código, las estructuras de datos y los gráficos elaborados simultáneamente. También permite la ejecución de los *scripts* línea por línea y seleccionar una parte del código y ejecutarlo, lo que resulta muy cómodo.

7.7 Datos

7.7.1 API AEMET OpenData

Los datos meteorológicos utilizados en este proyecto han sido obtenidos de la API de la AEMET (AEMET, s.f.). Esta es la Agencia Estatal de Meteorología de España, y cuenta con estaciones automáticas que cubren la geografía española. El objetivo de esta API es la difusión y reutilización de la información recabada por la Agencia.

Se ha decidido utilizar esta API por el carácter oficial de los datos ofrecidos, la facilidad para obtener una clave de uso y por ofrecer un gran número de recursos con una documentación muy clara e interactiva.

7.7.2 Salenbici

Se han utilizado datos históricos de uso del servicio de préstamo de bicicletas de Salenbici (Ayuntamiento de Salamanca, s.f.), promovido por el Ayuntamiento de Salamanca. Este servicio y los datos proporcionados se describirán en secciones posteriores del documento.

7.7.3 Open Street Maps

Open Street Maps (Open Street Maps, s.f.) es un proyecto colaborativo para crear una base de datos de información geográfica en forma de mapas que cubra todo el

planeta. La información es editada, actualizada y añadida por un gran número de colaboradores.

Los mapas elaborados incluyen información tanto geográfica como sobre el trazado de las carreteras y calles, fronteras entre países, comercios, puntos de interés…

Mediante la herramienta OSMWizard se ha obtenido información de esta fuente para la creación de algunos de los mapas que se valoró utilizar para la simulación. Finalmente, no ha sido posible utilizar estos mapas ya que contaban con un gran número de calles desconectadas del resto y, al ser importados en SUMO, eso impedía crear las rutas. En un futuro, SUMO añadirá la funcionalidad para corregir estos errores, lo que permitirá utilizar esta fuente.

7.8 Otras herramientas

7.8.1 Diagrams.net

Diagrams.net (antes draw.io) (JGraph LTD, s.f.) es una página web que permite crear diagramas, ofreciendo distintas plantillas. Existe también una versión de escritorio. Esta herramienta se ha utilizado para realizar todos los diagramas de la documentación.

7.8.2 Microsoft Office Word

Esta herramienta del paquete Office se ha utilizado para redactar la documentación del proyecto. Se ha escogido por sus amplias funcionalidades para la redacción y maquetado de documentos, por ejemplo, para la gestión de las referencias y la bibliografía.

7.8.3 Google Maps y Google Street View

Estas dos herramientas (Google, s.f.) se han utilizado para obtener información sobre la localización geográfica de las estaciones y el aspecto de estas.

8 DESCRIPCIÓN DE LA SOLUCIÓN

8.1 Análisis del caso de uso

8.1.1 Análisis del servicio Salenbici

Para el desarrollo de la solución se ha tomado como caso de uso concreto el servicio de préstamo de bicicletas de Salamanca, Salenbici. Este servicio depende del ayuntamiento de Salamanca y está operativo desde el año 2010 hasta la actualidad.

En primer lugar, se ha realizado un análisis de las características del servicio de préstamo. A continuación, se muestra una tabla que recoge las más relevantes.

Tipo de servicio	BSS basado en estaciones
Tipo de bicicletas	Tradicionales y eléctricas
Suscripción	Suscripción anual, 365 días desde el alta con un coste de 13€
	Gratis para abonados del transporte público municipal
Alta del usuario en el servicio	Puntos de alta y página web
Identificación en el sistema	Mediante un número de tarjeta de usuario y clave
Utilización del sistema	Las estaciones disponen de una terminal para realizar el préstamo o devolución de la bicicleta. El proceso lo realiza el usuario sin supervisión. Los candados se abren y cierran automáticamente. También se puede hacer uso de una aplicación, o enviar un SMS con los datos de la bicicleta.
Limitaciones de uso	1 hora por préstamo
	De 7:00 a 23:00 h de lunes a viernes y de 10:00 a 23:00 h los sábados, domingos y festivos
	Sin limitación de préstamos diarios
	Penalizaciones por incumplimiento de las condiciones
Número de estaciones	44
Capacidad de las estaciones	Más de 178 candados en total
	Las estaciones disponen de postes con dos candados y un candado extra en la terminal.
	Cada estación dispone de un número de candados distinto, dependiendo de la demanda prevista de la estación u otros motivos.
	Entre 9 y 21 candados en cada estación

Número de bicicletas	Desconocido
Ámbito geográfico	40 estaciones distribuidas por Salamanca capital
	2 estaciones en Santa Marta (población cercana)
	1 estación en Tejares (población cercana)
	1 estación en Vistahermosa (población cercana)
Distribución de las estaciones	Cubriendo todos los barrios y zonas de Salamanca
	La ubicación de las estaciones se ha escogido estratégicamente: conexión con trenes y buses, integración con carriles bici, ubicaciones en parques, plazas, zonas concurridas.
Sistema actual de balanceado	Desconocido

Tabla 2: Análisis del servicio Salenbici

En la siguiente imagen, obtenida de la página web de Salenbici, se puede observar la distribución de las estaciones.

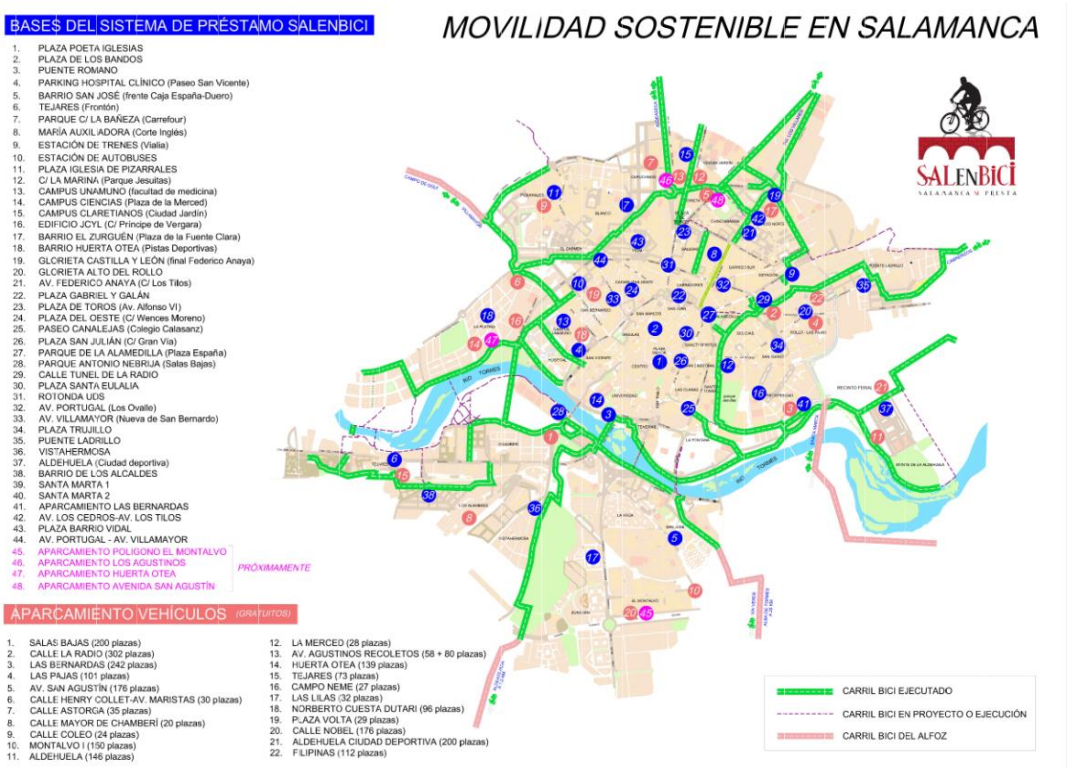

Figura 11: Mapa de distribución de estaciones de bicicletas de Salenbici

También podemos observar en la siguiente imagen la apariencia usual de las estaciones y las bicicletas del servicio. Cada uno de los postes tiene capacidad para dos bicicletas.

Figura 12: Ejemplo de estación de bicicletas (Redacción Noticias Salamanca, 2021).

A partir de estos datos se han obtenido las siguientes conclusiones:

- Utilización del sistema: El límite de tiempo de una hora por préstamo indica que la función del servicio es el préstamo de bicicletas para desplazamientos cortos. Es posible que los usuarios utilicen el servicio para desplazarse al trabajo, al centro de estudios o a una actividad de ocio.

- La distribución de las ubicaciones de las bicicletas propiciaría también su uso para desplazarse de un barrio residencial al centro de la ciudad.

- Se ha observado que Salenbici amplia frecuentemente (por lo menos una vez al año) el número de estaciones y de bicicletas. Esto indica que el servicio y el número de usuarios está en crecimiento. Además, estas ampliaciones sirven para aumentar la densidad de estaciones o para dar servicio a poblaciones cercanas. La densidad creciente de estaciones reduce la distancia entre ellas, ofreciendo al usuario más opciones para realizar los préstamos y devoluciones en caso de no encontrar disponibilidad en una estación. Esta característica es beneficiosa también para la implantación de un sistema de balanceado basado en la colaboración del usuario.

- Tipo de recompensa: Debido al bajo coste del servicio (13€ por un año de uso) y al método de pago, no tiene sentido ofrecer un descuento sobre el precio como incentivo al usuario por el uso del sistema de balanceado. Se descarta por tanto el incentivo económico, y se propone otro tipo de recompensa basada en puntos. Cuando el usuario acepte las sugerencias de estaciones del sistema recibirá un número determinado de puntos canjeables por recompensas, como pueden ser descuentos en otro tipo de actividades o regalos. Como se ha observado que existe una relación previa entre el servicio Salenbici y el transporte público municipal (buses urbanos), se propone la

posibilidad de canjear puntos por viajes gratuitos en los buses urbanos como ejemplo. Esta medida adicionalmente incentivaría el uso del transporte público.

8.1.2 Análisis de los datos proporcionados

Los datos de uso del sistema proporcionados por Salenbici son los siguientes:

Descripción de los datos	Datos de uso del servicio por los usuarios. Recoge los préstamos realizados, la fecha y hora de realización del préstamo y la devolución, y las estaciones involucradas (origen y destino).
Formato de los datos	6 archivos CSV (datos divididos por año y semestre)
Ámbito temporal recogido	2017-2020
Número de estaciones representadas	34 estaciones
Confidencialidad de los datos	Anónimos (número de tarjeta del usuario)

Tabla 3: Datos proporcionados por Salenbici

Utilizando las librerías Pandas y Matplotlib se ha realizado un análisis general de los datos, obteniendo los siguientes gráficos representativos del uso del servicio:

- Demanda por estaciones: Número de préstamos y devoluciones totales realizados en cada estación. Se puede observar que las cifras son desiguales. También hay diferencias entre el número de préstamos y devoluciones. En algunas estaciones esta diferencia es mínima, por ejemplo, en las ubicaciones de la estación de trenes y de autobuses. En cambio, en otras la diferencia es significativa, por ejemplo, Salas Bajas y la plaza del poeta Iglesias reciben más devoluciones que préstamos, mientras que en el parque del Carrefour o la Marina se realizan más préstamos que devoluciones.

  Al observar el gráfico, hay que tener en cuenta que algunas estaciones llevan menos tiempo en servicio. Por ejemplo, la estación de Santa Marta (la más reciente adición de las estaciones representadas en los datos) se puso en funcionamiento en el segundo semestre de 2020.

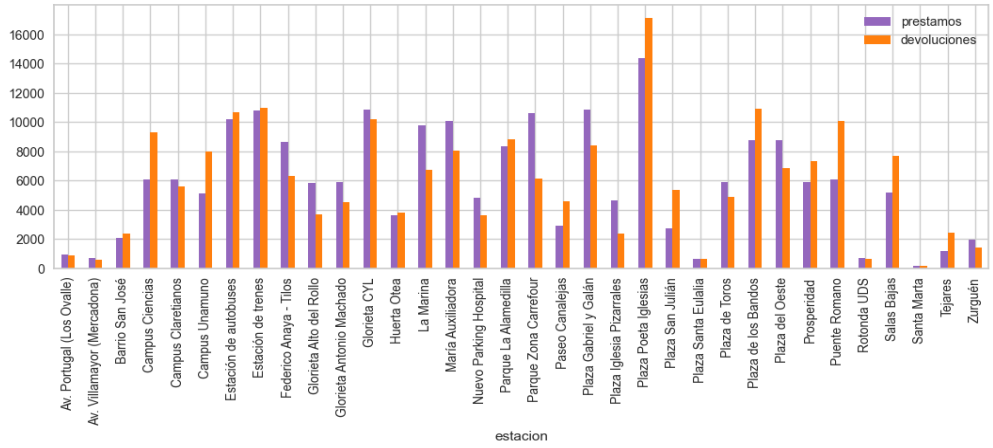

Figura 13: Demanda por estaciones

- Demanda temporal del servicio: El siguiente gráfico muestra la utilización del servicio a lo largo del tiempo. Se puede observar que existen picos de uso y temporadas más inactivas. Los meses de invierno registran cifras menores, por ejemplo. El uso del servicio también fluctúa dependiendo del día de la semana.

  Es muy significativo el impacto de la pandemia del Covid-19 en el gráfico. Durante los meses de confinamiento (marzo y abril de 2020) el servicio no pudo ser utilizado por los ciudadanos, por lo que apenas se observa actividad. El levantamiento de las restricciones en los meses posteriores coincide con la reanudación del servicio que se observa en el gráfico, aunque no se alcanzan los picos de uso de los años anteriores.

  Habrá que tener en cuenta el impacto de esta circunstancia excepcional a la hora de predecir la demanda, ya que los datos correspondientes a este período no reflejan el uso habitual del servicio.

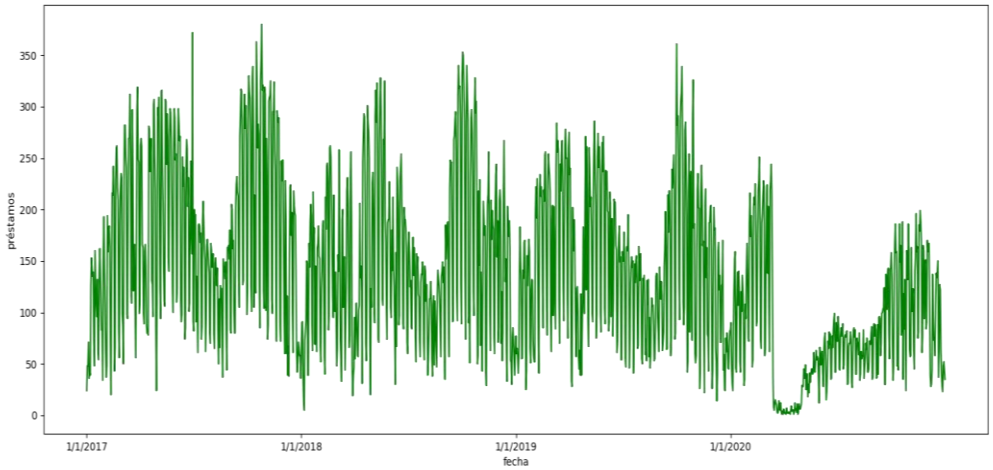

Figura 14: Demanda temporal del servicio en los últimos años

A continuación, se ha estudiado el número de préstamos realizados por cada usuario, la existencia de rutas más concurridas que otras y la combinación de ambas, es decir, usuarios realizando una ruta concreta habitualmente. Se pretende averiguar si existen usuarios o rutas que tengan un gran impacto en los datos y descubrir si existen patrones en la utilización del servicio.

Para ello, se han agrupado los datos atendiendo a cada uno de estos factores (usuario, ruta, ambos) y se han obtenido valores estadísticos y el gráfico de distribución.

También se han ordenado los datos por número creciente de préstamos asociados, se ha computado la suma cumulativa de los préstamos y se ha transformado a porcentajes para la visualización de los datos.

- Usuarios: Entre los años 2017 y 2020 han realizado préstamos de bicicletas 2471 usuarios. La media de préstamos ha sido de 81 por usuario y el 25% de los usuarios han utilizado el servicio menos de 7 veces en los 4 años.

Valores estadísticos préstamos por usuario	
media	81
std	167
min	1
25%	7
50%	24
75%	78
max	1822

Tabla 4: Valores estadísticos de los préstamos por usuario

La distribución del número de préstamos por usuario es la siguiente:

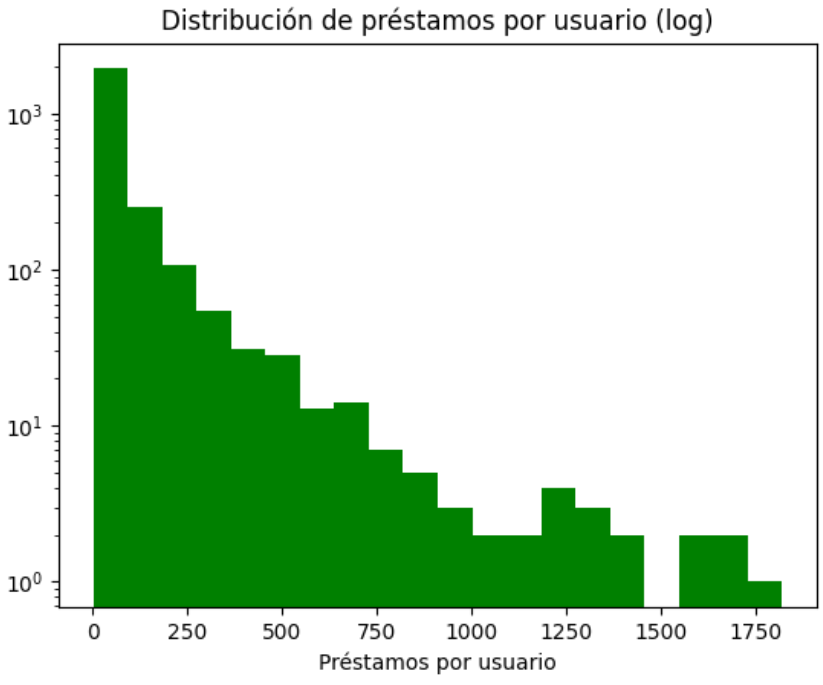

Figura 15: Gráfico de distribución de préstamos por usuario

En el siguiente gráfico se observa que un 20% de los usuarios es responsable de más del 70% de los préstamos del sistema.

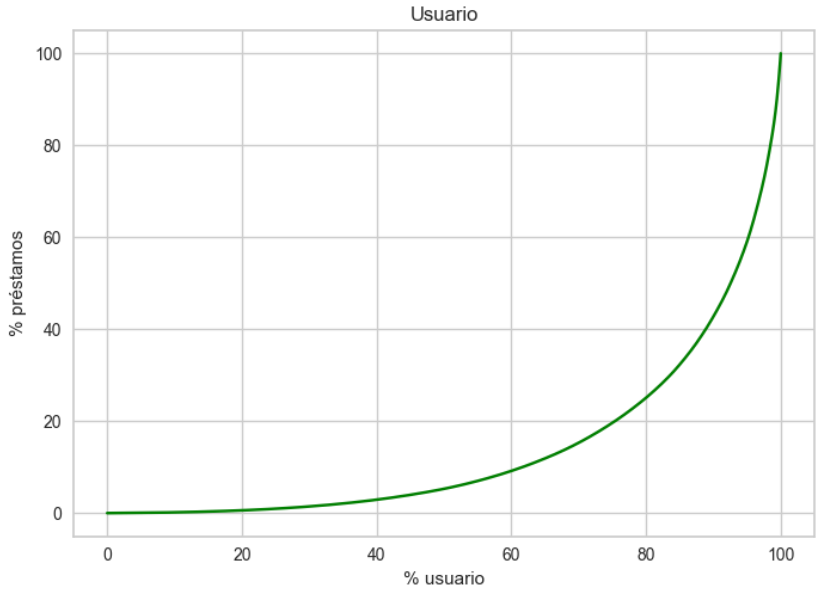

Figura 16: Relación entre usuarios y préstamos

Se puede concluir que la mayor parte de los usuarios registrados en el sistema realizan un uso ocasional del servicio. Los usuarios habituales, al contrario, tienen un gran impacto en los datos.

- Rutas: Considerando que una ruta sea un trayecto con la misma estación de origen y destino, se han realizado un total de 1112 rutas distintas. El total de rutas posibles entre las 34 estaciones es 1156 (34^2), por tanto 44 de las posibles rutas no han sido utilizadas nunca en los 4 años representados.

Valores estadísticos préstamos por ruta	
media	181
std	246
min	1
25%	22
50%	90
75%	240
max	1965

Tabla 5: Valores estadísticos de préstamos por ruta

La distribución del número de préstamos por ruta es la siguiente:

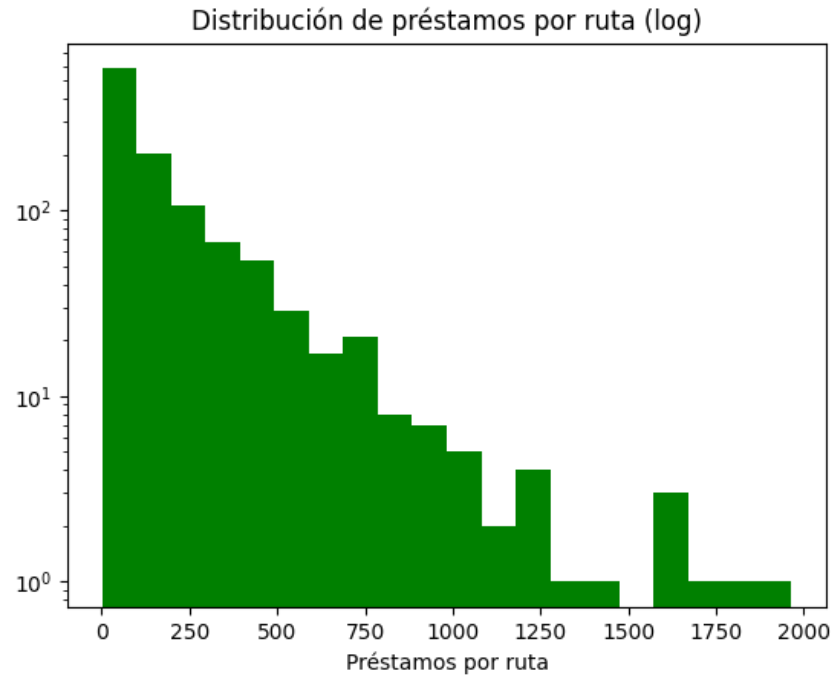

Figura 17: Gráfico de distribución de préstamos por ruta

El siguiente gráfico muestra que el 20% de las rutas han acumulado un 60% de los préstamos. Por tanto, podemos concluir que hay rutas mucho más concurridas que otras.

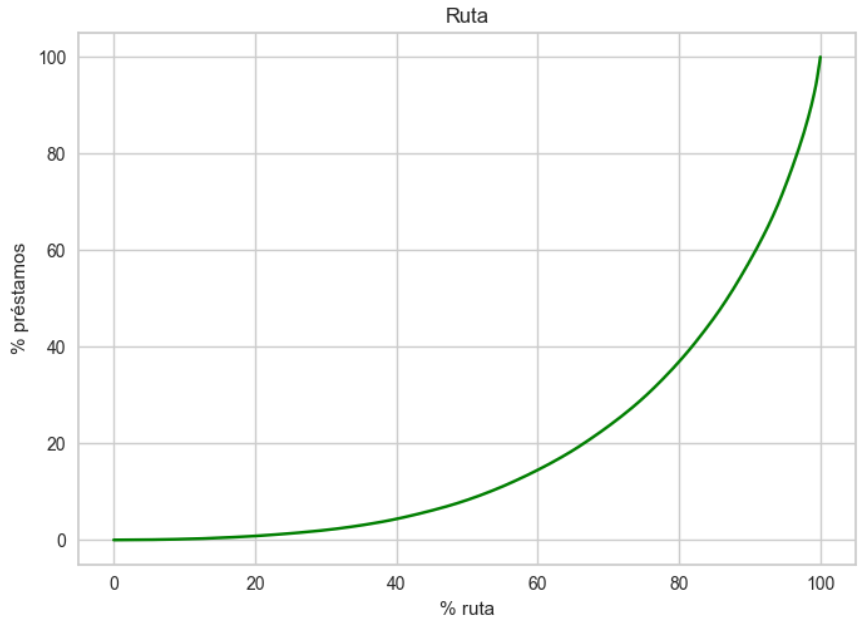

Figura 18: Relación entre rutas y préstamos

- Ruta y usuario: Existen 44197 combinaciones de rutas y usuarios en los datos. Los valores estadísticos que podemos encontrar a continuación nos indican

también que la mayoría de los usuarios utilizan una vez una ruta, es posible que esto sea debido a un uso no habitual del sistema, es decir, se utiliza el servicio puntualmente y entre estaciones diferentes.

Valores estadísticos préstamos por ruta y usuario	
media	4.5
std	16.65
min	1
25%	1
50%	1
75%	3
max	646

Tabla 6: Valores estadísticos de los préstamos por ruta y usuario

En el gráfico de distribución se observa también el elevado número de apariciones individuales de ruta y usuario (la escala es logarítmica).

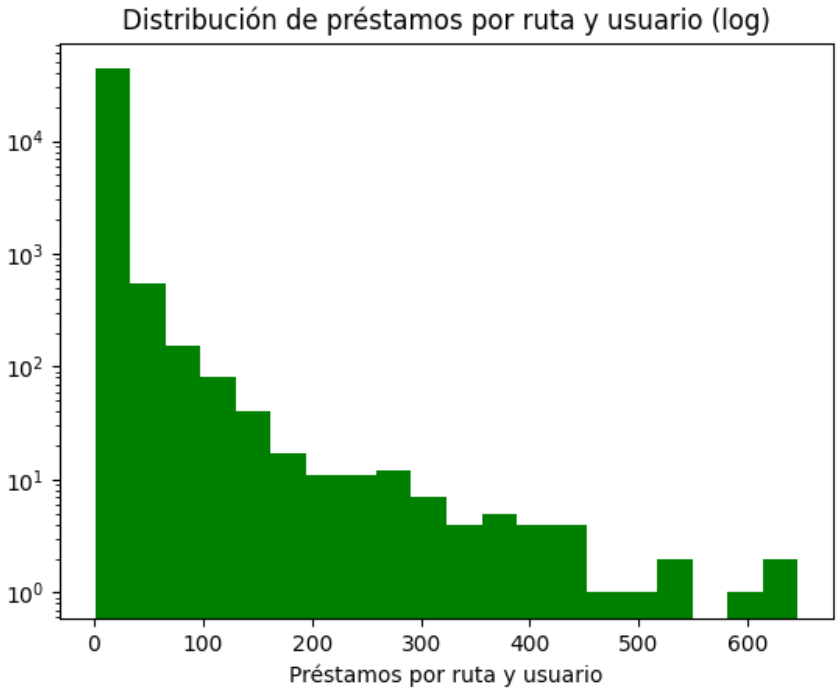

Figura 19: Gráfico de distribución de préstamos por ruta y usuario

En el siguiente gráfico se observa que más de un 50% de las combinaciones de usuario y ruta son únicas (crecimiento lineal). Como en los dos casos anteriores, un número muy pequeño de combinaciones (20%) representa más del 80% de las apariciones.

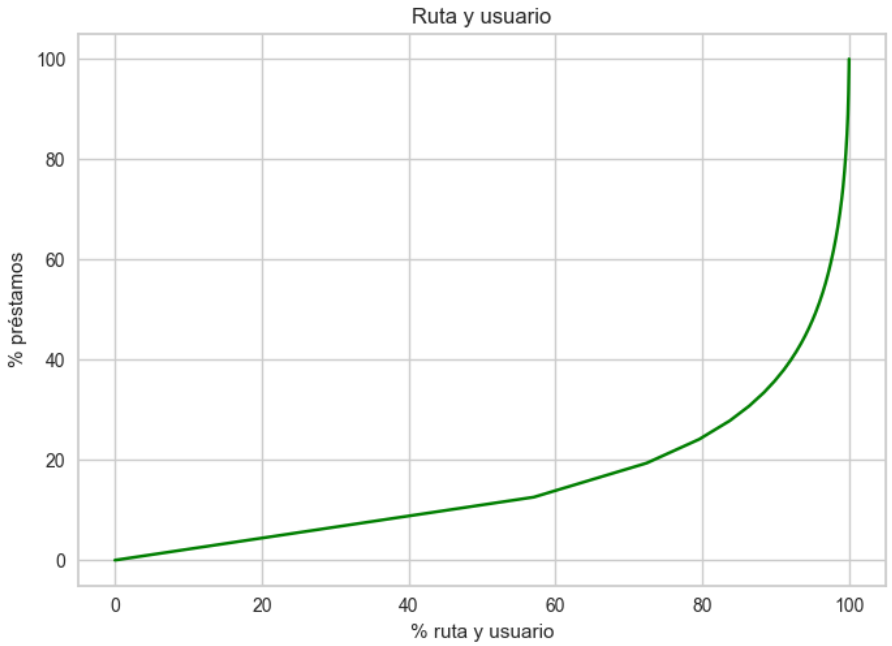

Figura 20: Relación entre rutas, usuarios y préstamos

Respecto a la diferencia en el número de estaciones representadas en los datos respecto al número actual, se ha decidido realizar la predicción de la demanda con los datos disponibles, pero incluyendo todas las estaciones en el sistema de balanceado. Las estaciones para las que no se dispone de datos utilizarán un valor por defecto como demanda diaria de bicicletas disponibles. Este valor se ha calculado como la mitad de la capacidad de la estación. Esta decisión considera que la demanda de candados libres y de bicicletas es semejante, ante la falta de datos.

8.1.3 Elaboración del fichero descriptor del servicio

Con los datos obtenidos en este análisis se ha elaborado un documento, denominado *estaciones*, que recoge los datos de las estaciones del servicio en formato csv (valores separados por comas). Este documento representará las características del BSS.

Para obtener las coordenadas de las estaciones y el número de candados se ha utilizado Google Maps y Google Imágenes.

Campo	**Descripción**	**Ejemplo**
Identificador de la estación	Nombre que identifica a la estación en el sistema de balanceado	*PLAZA_POETA_IGLESIAS*
Latitud	Ubicación de la estación expresada en coordenadas geodésicas	*40.96444722525418*
Longitud		*-5.663977000231472*
Número de bicicletas	Número de bicicletas disponibles en la estación al iniciar el sistema	*0*

Número de candados	Número de candados disponibles en la estación	*17*
Nombre de la estación en los datos	Nombre que identifica a la estación en los datos proporcionados por Salenbici	*Plaza Poeta Iglesias*
Identificador en Salenbici	Código que identifica a la estación en el servicio de préstamo Salenbici	*C1*

Tabla 7: Campos del fichero descriptor del servicio

8.1.4 Análisis de la distancia entre las estaciones

Se ha estudiado también la distancia entre las estaciones. Interesa especialmente saber cuál es la distancia habitual entre una estación y la estación más cercana, ya que este valor se puede utilizar para calibrar el sistema de balanceado.

Para calcular la distancia entre las estaciones se han utilizado las coordenadas geodésicas obtenidas anteriormente, almacenadas en el documento *estaciones.* La distancia se calcula utilizando la fórmula de Haversine, que tiene en cuenta la forma esférica de la Tierra. No es un método del todo preciso, ya que ignora el achatamiento de la Tierra en los polos, pero es suficientemente preciso para este propósito, ya que las distancias son pequeñas.

Las distancias calculadas entre cada una de las estaciones y las demás se han representado en el siguiente gráfico.

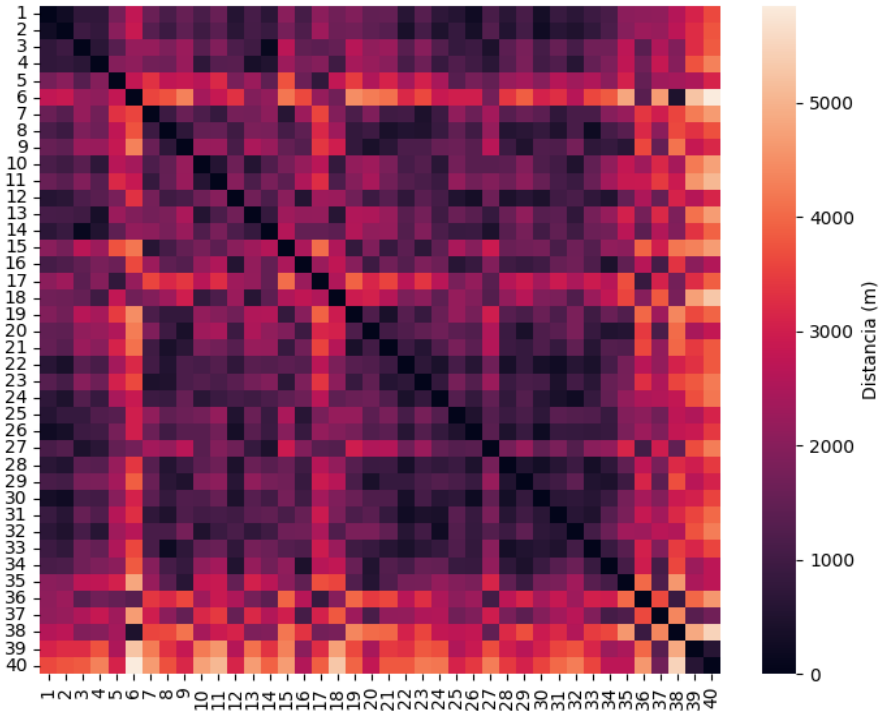

Figura 21: Distancias entre las estaciones de Salenbici

En la siguiente tabla se muestra el indicador de cada estación, su estación más cercana y la distancia entre ellas, en metros.

Nº	Estación	Estación más cercana	Distancia (m)
1	PLAZA_POETA_IGLESIAS	PLAZA_SAN_JULIAN	244
2	PLAZA_DE_LOS_BANDOS	PLAZA_SANTA_EULALIA	247
3	PUENTE_ROMANO	CAMPUS_CIENCIAS	132
4	HOSPITAL_VIRGEN_DE_LA_VEGA	CAMPUS_UNAMUNO	366
5	BARRIO_DE_SAN_JOSE	ZURGUEN	774
6	TEJARES	BARRIO_DE_LOS_ALCALDES	466
7	PARQUE_CALLE_LA_BAÑEZA	PLAZA_DE_TOROS	450
8	AVENIDA_DE_FEDERICO_ANAYA	AVENIDA_PORTUGAL_LOS_OVALLE	228
9	ESTACION_DE_TRENES	GLORIETA_ALTO_DEL_ROLLO	407
10	ESTACION_DE_AUTOBUSES	AVENIDA_VILLAMAYOR	501
11	PLAZA_IGLESIA_PIZARRALES	ESTACION_DE_AUTOBUSES	593
12	CALLE_LA_MARINA_JESUITAS	PLAZA_SAN_JULIAN	341
13	CAMPUS_UNAMUNO	HOSPITAL_VIRGEN_DE_LA_VEGA	366
14	CAMPUS_CIENCIAS	PUENTE_ROMANO	132
15	CAMPUS_CLARETIANOS	PARQUE_CALLE_LA_BAÑEZA	596
16	CALLE_VERGARA	PLAZA_DE_TRUJILLO	452
17	ZURGUEN	BARRIO_DE_SAN_JOSE	774
18	HUERTA_OTEA	CAMPUS_UNAMUNO	611
19	GLORIETA_CYL	AVENIDA_DE_FEDERICO_ANAYA-LOS_TILOS	390
20	GLORIETA_ALTO_DEL_ROLLO	TUNEL_DE_LA_RADIO	398
21	AVENIDA_DE_FEDERICO_ANAYA-LOS_TILOS	GLORIETA_CYL	390
22	PLAZA_DE_GABRIEL_Y_GALAN	GLORIETA_UDS	273
23	PLAZA_DE_TOROS	GLORIETA_UDS	443
24	PLAZA_DEL_OESTE	AVENIDA_VILLAMAYOR	240
25	PASEO_CANALEJAS	PLAZA_SAN_JULIAN	499
26	PLAZA_SAN_JULIAN	PLAZA_SANTA_EULALIA	226
27	SALAS_BAJAS	CAMPUS_CIENCIAS	438
28	PARQUE_ALAMEDILLA	PLAZA_SANTA_EULALIA	316
29	TUNEL_DE_LA_RADIO	GLORIETA_ALTO_DEL_ROLLO	398
30	PLAZA_SANTA_EULALIA	PLAZA_SAN_JULIAN	226
31	GLORIETA_UDS	PLAZA_DE_GABRIEL_Y_GALAN	273
32	AVENIDA_VILLAMAYOR	PLAZA_DEL_OESTE	240
33	AVENIDA_PORTUGAL_LOS_OVALLE	AVENIDA_DE_FEDERICO_ANAYA	228

34	PLAZA_DE_TRUJILLO	TUNEL_DE_LA_RADIO	441
35	PUENTE_LADRILLO	GLORIETA_ALTO_DEL_ROLLO	574
36	VISTAHERMOSA	ZURGUEN	850
37	ALDEHUELA	CALLE_VERGARA	1068
38	BARRIO_DE_LOS_ALCALDES	TEJARES	466
39	SANTA_MARTA_1	SANTA_MARTA_2_CTRA_MADRID	616
40	SANTA_MARTA_2_CTRA_MADRID	SANTA_MARTA_1	616

Tabla 8: Estaciones más cercanas a cada estación

A partir de esta tabla se han calculado también los valores estadísticos representativos y la distribución.

Distancias entre estaciones más cercanas	
media	432
std	201
min	132
25%	267
50%	402
75%	519
max	1068

Tabla 9: Valores estadísticos de distancias a la estación más cercana

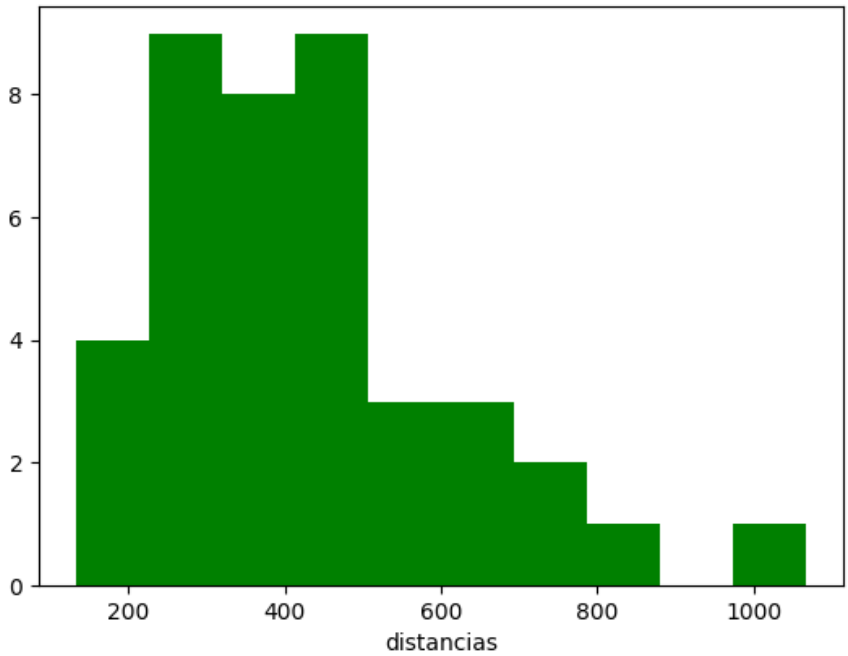

Figura 22: Distribución de las distancias a la estación más cercana

Se han extraído las siguientes conclusiones:

- El 75% de las estaciones cuentan con otra estación a menos de 520 metros. Esta distancia se puede tomar como referencia a la hora de calibrar el sistema.

- La media de distancia entre las estaciones más cercanas es de 432 metros.

- Excluyendo las estaciones situadas en poblaciones cercanas y barrios periféricos la mayoría de las estaciones están próximas a dos o más estaciones.

8.2 Arquitectura del sistema

Es este apartado se va a describir de forma general el sistema desarrollado, los factores tenidos en cuenta al diseñar su arquitectura y los distintos componentes que forman el sistema.

8.2.1 Factores en el diseño de la arquitectura

En el diseño de la arquitectura del sistema se ha tenido en cuenta dos de los objetivos planteados: la modularidad y la posibilidad de distribuir en varias máquinas los distintos componentes (sistema distribuido).

Debido a que la implementación del sistema de balanceado y la predicción de la demanda se han realizado en lenguajes de programación distintos, se ha optado por encapsular estos dos componentes en dos servidores distintos dentro de la misma máquina, y realizar las comunicaciones entre ellos utilizando el REST. Adicionalmente, esta solución permite distribuir estos componentes en máquinas distintas, en caso de ser necesario.

También es posible distribuir en distintas máquinas los agentes del sistema multi-agente de balanceado. Esto permitiría, por ejemplo, que cada agente estación estuviese alojado en una máquina distinta.

8.2.2 Componentes

El siguiente diagrama muestra la arquitectura del sistema:

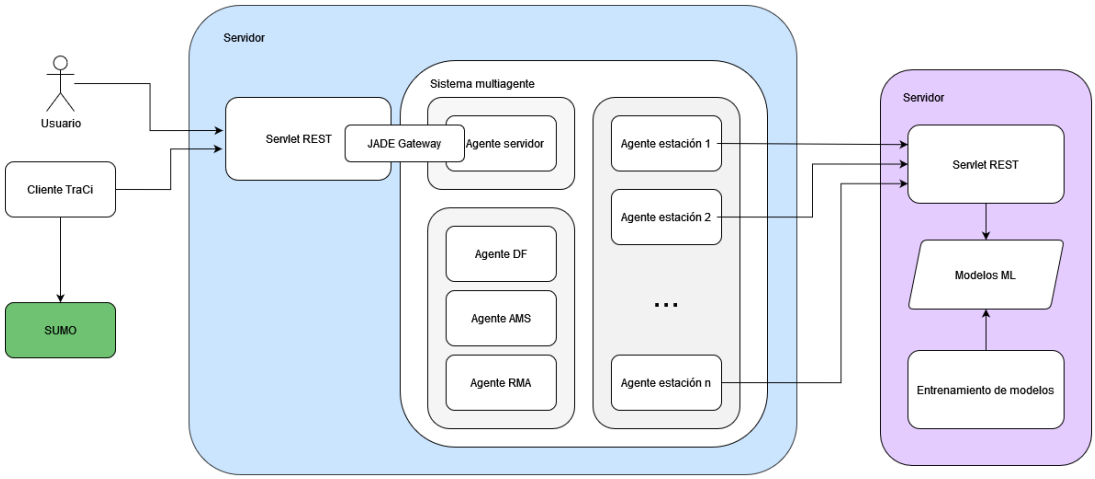

Figura 23: Arquitectura del sistema.

A continuación, se describen cada uno de los componentes del sistema:

- Sistema de balanceado: El sistema de balanceado se encarga de mantener la información sobre el estado del sistema, recibir las peticiones de los usuarios,

calcular las estaciones más cercanas y las recomendadas y devolver esta información al usuario.

  - Servidor: El sistema de balanceado está alojado en un servidor Tomcat, en la dirección de red localhost en el puerto 8080

  - Servlet REST: Recibe y procesa las peticiones. Está implementado utilizando Jersey.

  - Sistema multi-agente: El componente principal del sistema de balanceado es un sistema multi-agente implementado en JADE, plataforma puerto 1099. Este sistema está formado por un agente servidor, que recibe las peticiones, y tantos agentes estación como estaciones haya en el servicio BSS. Además, están presentes los agentes propios de la especificación FIPA para la gestión de los mensajes, los agentes y el directorio de servicios.

  - JADE Gateway: Comunica el *servlet* REST con el agente servidor del sistema multi-agente.

- Predicción de la demanda: Cada agente estación del sistema de balanceado se comunica con el sistema de predicción de la demanda, encargado de determinar el número de bicicletas y de espacios necesarios en cada estación para el día.

  - Servidor: El sistema de predicción de la demanda está alojado en el servidor de desarrollo de Flask, en el puerto 5000.

  - Servlet REST: Recibe y procesa las peticiones de los agentes. Esta implementado utilizado Flask.

  - Modelos ML: Modelos entrenados para predecir la demanda diaria. Estos modelos están serializados y son cargados por el *servlet*. Para cada estación hay dos modelos, uno para predecir los préstamos y otro para predecir las devoluciones.

  - Entrenamiento de modelos: Módulo encargado de entrenar los modelos a partir de los datos históricos. También permite visualizar los datos y los resultados del entrenamiento.

- Usuario: Representa al usuario del sistema, que es la persona que utiliza el BSS. Es un elemento fundamental, ya que el balanceado del sistema requiere su colaboración aceptando las estaciones propuestas. La comunicación entre el usuario y el sistema está basada en peticiones REST y en el estándar JSON, lo que permite que en el futuro se pueda desarrollar esta comunicación tanto desde una aplicación como desde una página web u otros sistemas.

- Simulación: Para comprobar el funcionamiento del sistema, se realiza una simulación, utilizando SUMO. Para ello se utilizan datos reales históricos de uso del BSS.

    - Cliente TraCI: Módulo que realiza peticiones al sistema de balanceado simulando ser los usuarios y se conecta a la simulación de SUMO para recrear los viajes realizados.

    - SUMO: TraCI se conecta a SUMO siguiendo un esquema cliente-servidor. SUMO recibe los mensajes de TraCI en el puerto 8813.

En los siguientes apartados se explicarán los componentes del sistema más detalladamente.

8.3 Sistema de balanceado

El sistema de balanceado se encarga de mantener la información sobre el estado del sistema e intentar corregir los desequilibrios mediante la recomendación de estaciones a los usuarios. Los usuarios realizan consultas al sistema solicitando información sobre las paradas más cercanas al lugar geográfico al que desean llegar o del que desean partir. El sistema facilita esta información y adicionalmente recomienda el uso de otras estaciones que beneficien al balanceado de las estaciones. Este sistema esta implementado como un sistema multi-agente con el cual el usuario se comunica mediante un servlet REST.

8.3.1 Sistema multi-agente

El sistema multi-agente está compuesto por dos tipos de agentes, además de los agentes auxiliares proporcionados por JADE. Los agentes estación representan las estaciones del BSS. El agente servidor gestiona las peticiones y recomendaciones a los usuarios y se comunica con las estaciones para obtener información de estas que permita optimizar el balanceado del sistema. También se comunica con estas para cambiar el estado del sistema cuando un usuario realiza un préstamo o devolución. Los agentes estación se pueden añadir o eliminar en cualquier momento de la ejecución.

En los siguientes subapartados se detallarán las funciones de cada agente y los mecanismos de comunicación entre ellos y con el exterior.

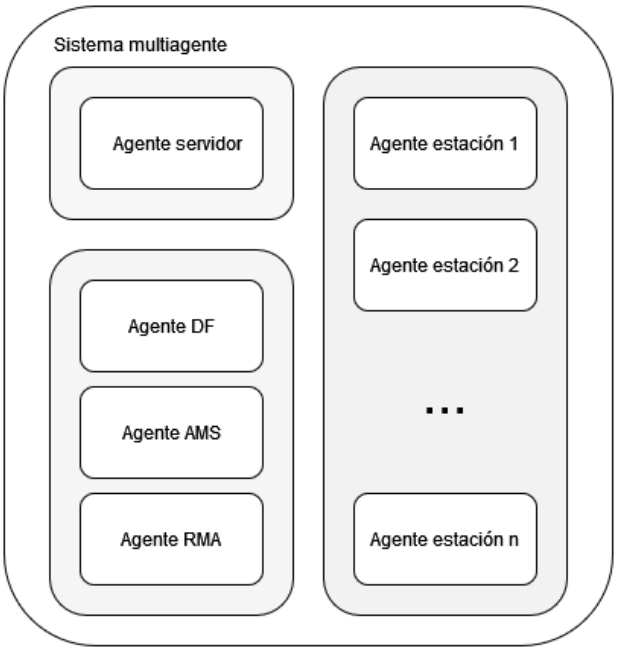

Figura 24: Diagrama del sistema multi-agente

8.3.2 Plataforma y contenedores

La implementación del sistema multi-agente se ha realizado en JADE y sigue las especificaciones FIPA. El sistema está compuesto de una plataforma con varios contenedores, que pueden encontrarse en una misma máquina o en distintas máquinas.

El contenedor principal contiene los agentes que ofrecen los servicios requeridos por FIPA para la gestión de agentes.

- El agente ams (*Agent Management System*) se encarga de la gestión de los agentes, por ejemplo, de registrarlos en la plataforma con un identificador único o AID.

- El agente df (*Directory Facilitator*) mantiene un directorio con los servicios que ofrecen los agentes.

- El agente rma (*Remote Monitoring Agent*) permite monitorizar a los agentes, por ejemplo, a través de su interfaz gráfica.

Se han creado otros dos contenedores en la plataforma, uno en el que residen los agentes estación y otro en el cual reside el agente servidor.

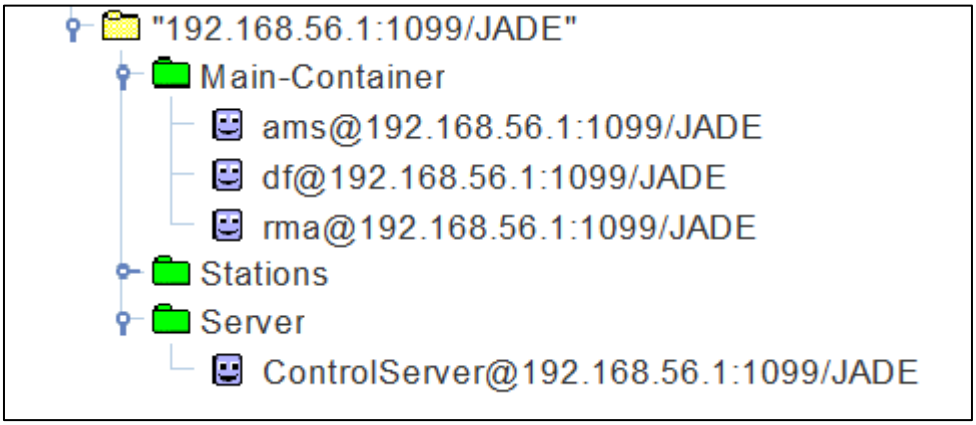

Figura 25: Distribución de los contenedores y agentes

8.3.3 Agentes estación

Los agentes estación almacenan la información necesaria para el funcionamiento de la estación y para colaborar en el balanceado del sistema, en concreto los siguientes datos:

- Coordenadas de la estación

- Capacidad máxima de la estación (número de candados)

- Número actual de bicicletas

- Número mínimo de bicicletas que debe tener la estación para funcionar correctamente

- Número mínimo de espacios disponibles que debe haber en la estación para su buen funcionamiento

- Predicción diaria de demanda

- Predicción por defecto si no se consigue obtener la predicción diaria

El agente ejecuta dos comportamientos de forma paralela:

- Predicción de la demanda: Comportamiento que se ejecuta mediante un temporizador una vez al día para obtener el número de préstamos y devoluciones esperados en la estación. Para ello realiza una petición al sistema de predicción de la demanda que se describirá más adelante, indicando el identificador de la estación. Extiende la clase `TickerBehaviour` de JADE.

  Una vez obtenidos el número de préstamos esperados y el número de devoluciones, se calcula el reparto ideal de bicicletas y espacios en la estación, teniendo en cuenta que se respeten unos números mínimos de bicicletas y espacios.

- Respuesta a mensajes del servidor: Comportamiento cíclico que espera la llegada de mensajes de manera bloqueante. Cuando recibe un mensaje lo procesa dependiendo del protocolo y de la performativa. Este comportamiento extiende la clase `CyclicBehaviour` de JADE

El agente puede recibir tres clases de mensaje del servidor:

- Solicitud de métricas: El agente servidor solicita a la estación sus métricas. Estas métricas son la distancia a un punto geográfico indicado en el mensaje y la prioridad de la estación. La prioridad de la estación se calcula como la relación entre el número de bicicletas y la predicción de la demanda realizada. Este valor parametriza las necesidades actuales de la estación.

  La función de esta acción es conocer la estación más cercana y la más apropiada para recomendar al usuario con el fin de equilibrar el número de bicicletas entre las estaciones. Cuando el agente estación recibe este mensaje calcula ambas métricas y las envía al agente servidor. La distancia al punto se calcula mediante el método de Haversine y la prioridad mediante el algoritmo descrito en la siguiente figura.

```
si capacidad máxima == 0 devolver 0;
       si número actual de bicicletas == 0 devolver 20;
       si número actual de bicicletas == capacidad máxima devolver -20;
       si número actual de bicicletas < mínimo de bicicletas devolver 15;
       si(capacidad máxima - número actual de bicicletas) < mínimo de espacios devolver -15;
       si no {
       devolver (predicción diaria - número actual de bicicletas) / predicción diaria * 10;
     }
```

Figura 26: Algoritmo de cálculo de métricas de las estaciones

- Solicitud de préstamo o devolución: El agente realiza esta solicitud cuando un usuario ha indicado que quiere tomar prestada o devolver una bicicleta en la estación. Al recibir el mensaje, se comprueba si es posible realizar la acción. Si no hay espacios disponibles no se puede realizar una devolución, por ejemplo. Tampoco se puede realizar un préstamo si no hay bicicletas disponibles. Si es posible realizar el préstamo se modifica el número actual de bicicletas de la estación según proceda.

- Solicitud de información sobre balanceado: El agente estación responde a esta solicitud con la información sobre el estado de balanceado de la estación. Como todas las estaciones tienen una capacidad diferente, este valor se calcula según la siguiente fórmula, que devuelve el valor normalizado entre -1 y 1:

$$balanceado = 2 * \frac{número\ actual\ de\ bicicletas}{capacidad\ máxima\ de\ la\ estación} - 1$$

8.3.4 Agente servidor

El agente servidor inicialmente no ejecuta ningún comportamiento. Cuando llega una petición de un usuario, valora el tipo de acción a realizar y añade un comportamiento simple (o `OneShotBehaviour` en JADE), que realiza la acción necesaria y finaliza. Estas acciones requieren de la comunicación con los agentes estación. Las acciones disponibles son:

- Recomendación de estaciones: El agente recibe unas coordenadas del usuario y solicita a todos los agentes estación sus métricas con respecto a esas coordenadas. Cuando recibe todas las métricas o se agota el tiempo de recepción de respuestas, calcula la estación más cercana al punto y la estación a recomendar, y responde al usuario con estas. La estación recomendada se calcula a partir de las distancias y las prioridades enviadas por los agentes estación en las métricas.

  La prioridad es un valor que varía entre -20 y 20 según la ocupación de la estación. Si la estación está completamente vacía, su prioridad es 20. Si la estación está completamente llena su prioridad es -20.

  El algoritmo de selección es el siguiente en el caso de realizar una devolución:

```
para cada estación en estaciones {
    si prioridad de la estación >= prioridad de la estación recomendada {
        si distancia de la estación <= distancia de la estación recomendada {
            estación recomendada = estación;
        }
        si no {
            si distancia de la estación - distancia de la estación más cercana < DIFERENCIA MÁXIMA) {
                estación recomendada = estación;
            }
        }
    }
}
```

Figura 27: Algoritmo de selección de las estaciones

Para un préstamo se utiliza el mismo algoritmo, pero se busca que la prioridad sea lo más baja posible, ya que en este caso indicaría que la estación tiene más bicis de las que se requieren y faltan espacios.

La respuesta a la petición del usuario se realiza en el formato JSON. La respuesta contiene la información sobre la estación más cercana y sobre la estación recomendada, con los siguientes campos para las dos:

Campo		**Ejemplo**
Identificador de la estación		*PUENTE_ROMANO*
Coordenadas	Latitud	*40.959216404952215*
	Longitud	*-5.669309603365671*
Distancia		*0.32560276870601007*
Prioridad		*20*
Número actual de bicicletas		*0*
Número actual de espacios		*11*
Incentivo		*30*

Tabla 10: Respuestas a las peticiones de sugerencia de estación

- Solicitud de préstamo: El agente solicita al agente estación la realización de un préstamo o devolución.

- Solicitud de información de balanceado: El agente solicita a todas las estaciones su información de balanceado y la devuelve al usuario.

8.3.5 Concurrencia

Los comportamientos generalmente se añaden al agente en cualquier momento de la vida de este, y su ejecución sigue un esquema *Round Robin* no preventivo, es decir, los comportamientos se añaden a una cola y se ejecutan de manera secuencial, sin tener en cuenta si uno de los comportamientos evita la ejecución de los demás.

La ejecución de comportamientos de manera concurrente se logra mediante el uso de la clase de JADE `ThreadedBehaviourFactory`. Esta clase permite encapsular los comportamientos y ejecutarlos en hilos distintos.

JADE proporciona también un comportamiento denominado `ParallelBehaviour`, que permite ejecutar dos comportamientos de manera paralela. Este es el comportamiento que se ha utilizado en el agente estación para permitir mantener al

agente en espera de mensajes y realizar acciones a la vez. Adicionalmente, se ha ejecutado uno de ellos en un hilo diferente para mejorar el rendimiento.

En el agente servidor crea un hilo por cada petición del usuario, para evitar el bloqueo de otras peticiones mientras se trata una de ellas.

El uso de hilos requiere que se utilicen mecanismos de sincronización cuando existan variables compartidas. En este caso, ha sido necesario proteger una variable compartida por los distintos hilos en el agente servidor, utilizada para asignar identificadores a las distintas conversaciones.

8.3.6 Comunicación entre los agentes

La comunicación entre los agentes se realiza mediante mensajes ACL.

Se han definido tres protocolos:

- Solicitud de métricas: Este protocolo se utiliza cuando el agente servidor quiere obtener información sobre las estaciones para decidir cuál de ellas recomendar. El agente servidor envía un mensaje a todas las estaciones solicitando sus métricas (performativa REQUEST) y estas contestan con un mensaje cuyo contenido es un objeto que representa las métricas solicitadas. Se ha utilizado la performativa PROPOSE para las respuestas ya que se considera que las estaciones realizando una propuesta con el objetivo de ser escogidas como estación recomendada al usuario.

  El mensaje enviado por el agente servidor contiene las coordenadas geográficas del punto que el usuario ha solicitado como punto de origen o de llegada. Las métricas del mensaje de respuesta contienen la distancia al punto solicitado y la prioridad de la estación.

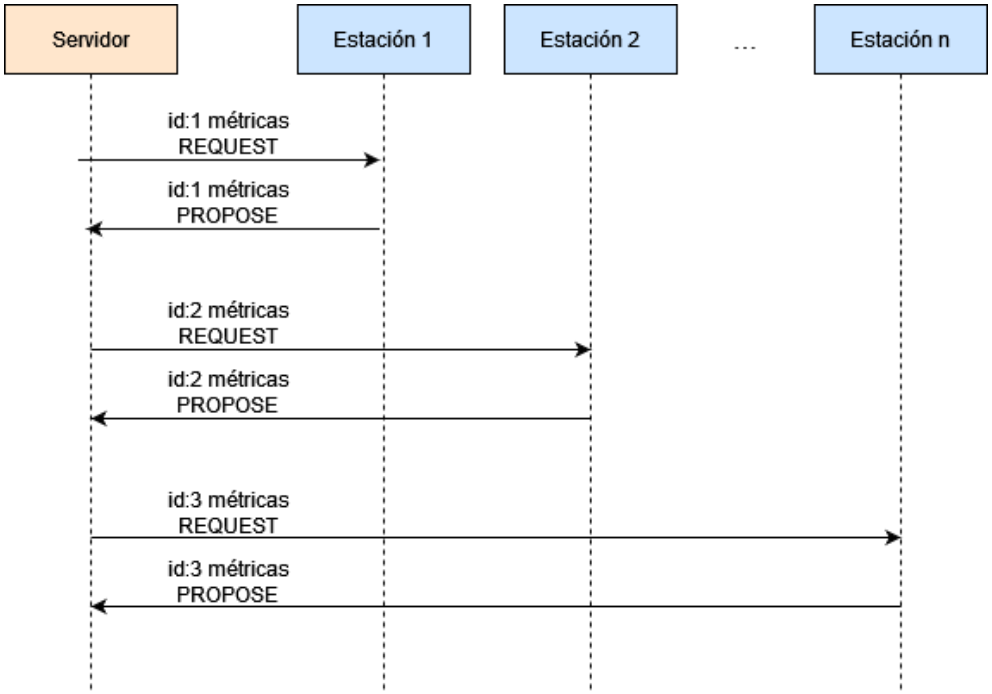

Figura 28: Protocolo de solicitud de métricas

Si el agente estación no logra obtener las métricas solicitadas responde con la performativa FAILURE.

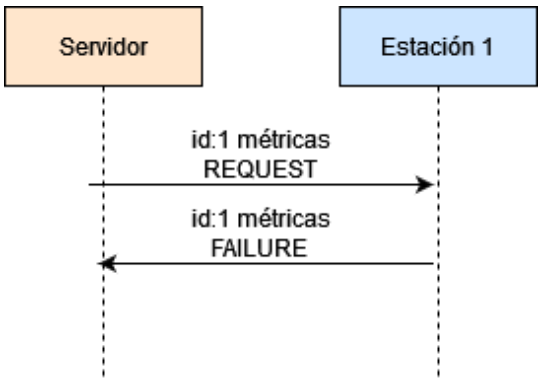

Figura 29: Protocolo de solicitud de métricas cuando ocurre un error

El agente estación espera a recibir todos los mensajes, pero utiliza un temporizador para evitar quedarse bloqueado si alguno de los mensajes nunca es contestado. Si se agota el tiempo de espera, el agente continúa su ejecución.

- Gestión de préstamos: Este protocolo se utiliza para gestionar los préstamos realizados. Cómo las peticiones realizadas para solicitar un préstamo llegan al agente servidor, este necesita comunicar al agente estación correspondiente la intención del usuario de tomar prestada o devolver una bicicleta. Si es posible realizar el préstamo o devolución, el agente estación responde con la performativa CONFIRM y actualiza su información.

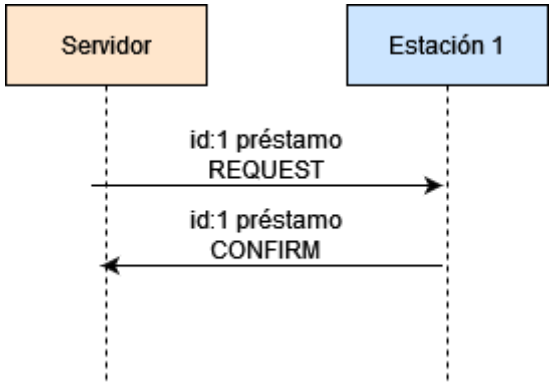

Figura 30: Protocolo de gestión de préstamos

Si no es posible realizar la acción, el agente estación contesta con la performativa REFUSE.

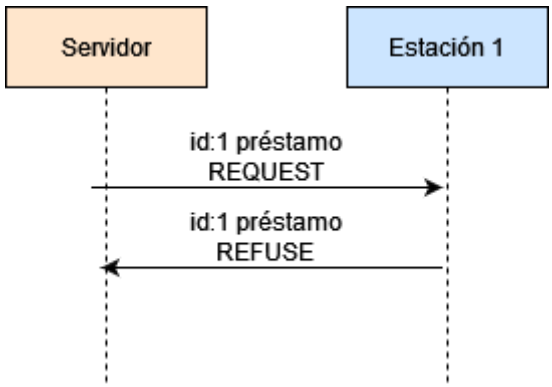

Figura 31: Protocolo de gestión de préstamos cuando no es posible el préstamo

- Información de balanceado: Este protocolo se utiliza cuando se desea conocer el estado del balanceado del servicio. El agente servidor solicita a cada una de las estaciones el estado de la estación, es decir, la relación entre el número de bicicletas y de espacios. Para ello utiliza la performativa REQUEST. El agente estación responde con la performativa INFORM y la información requerida.

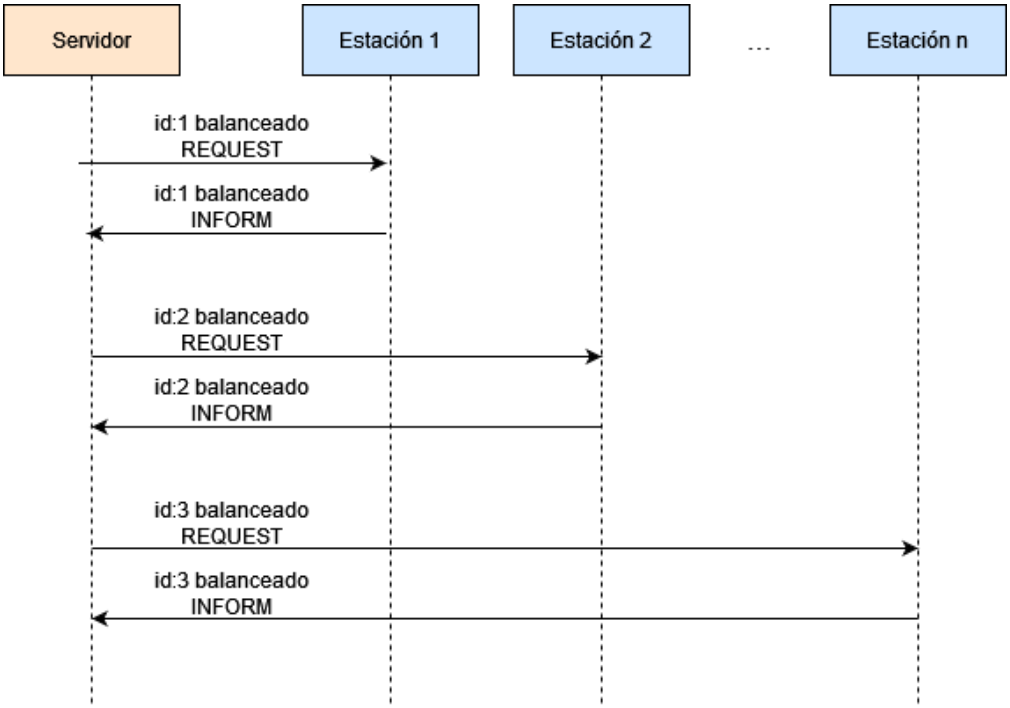

Figura 32: Protocolo de solicitud de la información de balanceado

Si el agente estación no puede obtener la información, responde con la performativa FAILURE.

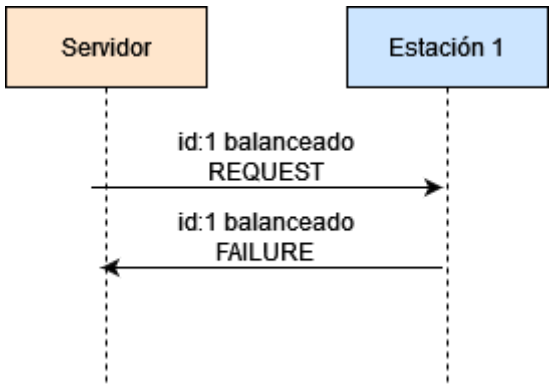

Figura 33: Protocolo de solicitud de la información de balanceado cuando ocurre un error

Como se pueden producir distintas conversaciones de un mismo protocolo a la vez para distinguir unas conversaciones de otras se utilizan identificadores de conversación. Estos consisten en un campo del mensaje en el cual mediante un número se indica la conversación a la que pertenece el mensaje.

Para obtener este número se utiliza una variable compartida por todos los hilos del servidor que se incrementa con cada nueva conversación, y se reinicia cuando llega a un valor lo suficientemente alto para no mantener la conversación número 0.

8.3.7 Registro de los agentes

Para poder comunicarse entre ellos, los agentes necesitan poder conocer el nombre de los demás. Para ello, los agentes registran sus servicios en el directorio de la plataforma. En este caso, sólo es necesario que se registren los agentes estación, ya que es el agente servidor el que inicia siempre la comunicación.

El registro de los servicios de los agentes se gestiona mediante el agente DF. Los agentes estación solicitan introducir la información sobre sus servicios en el directorio de la plataforma, y el agente servidor solicita a este la lista de AID de los agentes que prestan el servicio de estación.

La comunicación con el agente DF también se realiza mediante mensajes ACL.

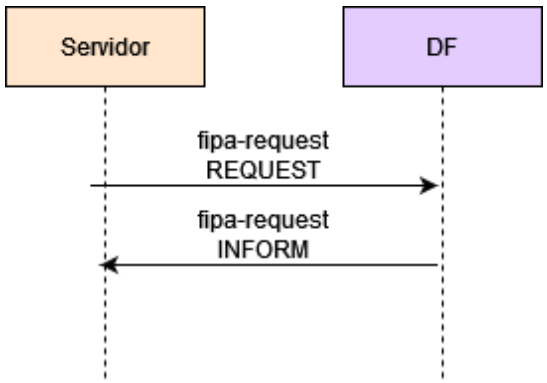

Figura 34: Comunicación con el agente DF

Los agentes estación se registran al inicio de su ejecución y eliminan su registro al terminar su ejecución, por ejemplo, al ser eliminados de la plataforma.

Cada vez que el agente servidor desea realizar una acción que implique la comunicación con todos los agentes estación disponibles, como en el caso de los protocolos de solicitud de métricas e información de balanceado, se comunica con el agente DF para obtener los AID de estos.

8.3.8 Comunicación entre el agente servidor y el servlet: Jade Gateway

La comunicación entre el sistema multi-agente y el exterior se realiza mediante un mecanismo facilitado por JADE denominado Jade Gateway. De esta forma podemos comunicar los agentes con el servlet, y hacer llegar a estos las peticiones de los usuarios. Este mecanismo consta de dos clases, por un lado, la clase `JadeGateway`, utilizada en el lado del servlet, y por otro, la clase `GatewayAgent`, que debe ser extendida por el agente designado como enlace dentro del sistema multi-agente, en este caso, el agente servidor.

La clase `JadeGateway` permite el paso de información de varias formas, por ejemplo, permite introducir un nuevo comportamiento en un agente, pero se ha optado por el

uso de comandos para transmitir la información deseada. Estos comandos son objetos instanciados de la clase que se desee.

En el lado del servlet, mediante el método `init()` de la clase `JadeGateway`, se indica la clase del agente que va a servir como enlace y algunas propiedades adicionales. En nuestro caso, se ha indicado el contenedor en el cual debe ser creado el agente. Este método se encarga de crear el agente, inicializarlo y añadirlo a la plataforma. Para enviar los comandos se utiliza el método `execute()`, indicando como parámetro el comando.

El agente servidor extiende la clase `GatewayAgent` y sobrescribe el método `processCommand()`. Este método recibe los distintos comandos y, dependiendo de su clase, decide qué acciones realizar. Cuando termina, escribe los resultados en el comando y utiliza el método `releaseCommand()`.para indicar que el comando ya ha sido procesado.

Una vez el comando ha sido procesado, desde el servlet se puede acceder a los resultados.

Para establecer el enlace al iniciar el servlet, se ha utilizado un *listener* que detecta el evento de inicio del servlet y ejecuta `JadeGateway.init()` en ese momento.

8.3.9 Servlet REST

Para la implementación del sistema de balanceado como un *servlet* se ha utilizado Jersey y Tomcat como servidor.

En primer lugar, ha sido necesario configurar el proyecto. Para ello se han añadido las bibliotecas tanto de Jersey como de JADE y de Tomcat al *build path* en la carpeta *web* del proyecto. Es importante que todas las bibliotecas estén accesibles desde esa carpeta.

En esta carpeta se ha incluido también el archivo web.xml que describe el servlet. En él se indican los paquetes donde se pueden encontrar los recursos web y el patrón url para mapear los recursos del servlet. Se ha establecido "/" como la raíz de las url de los distintos recursos. También se indica el *listener* que detecta el inicio del servlet.

8.3.10 Creación de los agentes estación a partir del archivo descriptor

Se ha creado también una funcionalidad para crear los agentes estación a partir del archivo descriptor del BSS.

8.4 Predicción de la demanda

Como se indica en el apartado anterior, los agentes estación del sistema de balanceado del BSS consultan diariamente la predicción de la demanda de bicicletas y espacios para la estación. El siguiente paso ha sido, por tanto, la implementación de la predicción de la demanda y el despliegue del servidor al que se realizan las peticiones. Las predicciones se obtienen a partir de la aplicación de técnicas de *Machine Learning* sobre los datos históricos de uso del servicio.

Se ha decidido separar los datos por estación y realizar las predicciones de forma aislada para cada una. Además, como se ha explicado previamente, algunas estaciones no existían en el período en el que se recogieron los datos con los que se ha trabajado, por lo que para ellas no se creará un modelo de predicción, sino que se tomará un valor por defecto.

Por otro lado, las variables a predecir son dos: el número de préstamos y el número de devoluciones que se van a producir en el día. Se han entrenado, por tanto, dos modelos por cada estación.

En primer lugar, se han limpiado y analizado los datos. El formato de los datos proporcionados por Salenbici se diferenciaba bastante del formato deseado, de modo que se ha necesitado convertir estos datos y obtener nuevas variables. Posteriormente se ha experimentado con varias técnicas y modelos, intentando lograr los mejores resultados y reducir el error. Una vez escogidos y entrenados los modelos, se han serializado. Estos modelos son cargados en el servidor al iniciar este la ejecución para realizar las predicciones según las peticiones entrantes.

Para realizar este procesamiento de los datos inicialmente se utilizó R para la limpieza y visualización de los datos y Python para el entrenamiento y puesta en funcionamiento del servidor. Esta decisión se tomó por dos motivos:

- R y el entorno de desarrollo RStudio proporcionaban mayor facilidad para la visualización de tablas y creación de gráficos

- Python cuenta con varias librerías potentes e intuitivas para el entrenamiento de modelos de *Machine Learning* y para el despliegue de servidores y tratamiento de peticiones REST.

Sin embargo, finalmente se decidió implementar todos los componentes en Python para facilitar el mantenimiento del sistema y conservar su sencillez, por lo que se descartó la implementación realizada en R.

8.4.1 Datos utilizados

Los datos utilizados han sido recogidos durante cuatro años de uso del servicio Salenbici (2017-2020). Como estos datos han sido generados de manera automática por el sistema de préstamos no presentan los fallos característicos de la recogida de datos realizada por humanos (errores tipográficos, inconsistencia…).

Los datos se proporcionaron en forma de archivos CSV y recogían las siguientes variables:

Variable	Descripción	Ejemplo
FechaInicio	Fecha y hora de realización del préstamo	*30/06/2017 21:38*
FechaFin	Fecha y hora de realización de la devolución	*30/06/2017 21:51*
Bicicleta	Identificador de la bicicleta prestada	*b11*
Origen	Estación donde se realizó el préstamo	*Puente Romano*
Destino	Estación donde se realizó la devolución	*Huerta Otea*
CandadoOrigen	Identificador del candado en el que se encontraba anclada la bicicleta al realizar el préstamo	*c302*
CandadoDestino	Identificador del candado donde fue devuelta la bicicleta	*c1804*
Tarjeta	Número de tarjeta del usuario que realizó el préstamo	*XXXXXXXXXXX*

Tabla 11: Formato de los datos proporcionados

Se realizaron las siguientes observaciones sobre el formato de los datos:

- Las variables bicicleta, candadoOrigen, candadoDestino y Tarjeta no aportan información relevante, por lo que se descartarán.

- Nos interesa conocer el número de préstamos y devoluciones diarios, por lo que tendremos que transformar los datos, que actualmente reflejan información sobre cada préstamo de manera individual. Se deberán separar los datos sobre el préstamo de los de la devolución y agruparlos por fecha.

- Se ha decidido hacer la predicción en intervalos de tiempo de un día, por lo que se normalizarán las fechas, eliminando la información sobre la hora.

- Los datos no recogen las suficientes variables características que nos permitan realizar una predicción. Hará falta crear nuevas variables que aporten información adicional.

8.4.2 Preparación de los datos

El algoritmo de preparación de los datos utilizado es el siguiente:

Figura 35: Algoritmo de preparación de los datos

Como se puede observar en la imagen, se han realizado varias acciones para transformar los datos al formato deseado. Principalmente se han agrupado los datos de préstamos individuales por estación, fecha, y acción (préstamo y devolución), y se han introducido observaciones con 0 préstamos y 0 devoluciones para las fechas en las cuales no se produjo ninguna acción en la estación.

Posteriormente se han añadido variables adicionales, tanto de los valores climatológicos registrados en cada fecha, como de los festivos.

8.4.3 Valores climatológicos (API AEMET)

Para obtener los valores climatológicos se ha utilizado la API REST de la AEMET (Agencia Estatal de Meteorología). En primer lugar, se ha necesitado solicitar una clave, requerida para hacer peticiones. Estas están restringidas además a 50 por minuto.

Para evitar el uso excesivo de la API, se almacenan los datos obtenidos en un fichero CSV.

De todos los recursos distintos que ofrece, se ha utilizado el recurso que ofrece los valores climatológicos registrados entre dos fechas para una estación específica. Debido a que se ha observado que este recurso falla si se supera la limitación existente de tiempo máximo entre las dos fechas introducidas, de cara a evitar problemas se ha decidido realizar las peticiones de año en año.

La URI del recurso utilizado es:

```
/api/valores/climatologicos/diarios/datos/fechaini/{fechaIniStr}/fechafin/{fechaFinStr}/estacion/{idema}
```

El parámetro `{idema}` indica la estación meteorológica automática (EMA) de la cual se obtendrán los datos. En nuestro caso se ha indicado la estación 2870, correspondiente a Salamanca.

8.4.4 Datos sobre festivos

Los datos sobre festivos se han obtenido de diversas fuentes de noticias que informan del calendario laboral. Se han escrito en varios ficheros los días festivos de cada año. Sería interesante en un futuro encontrar una forma de automatizar este proceso.

Una vez obtenidos ambos, estos se han convertido en tablas y se han añadido a los datos de préstamos mediante operaciones JOIN. Este proceso se puede observar en la siguiente figura:

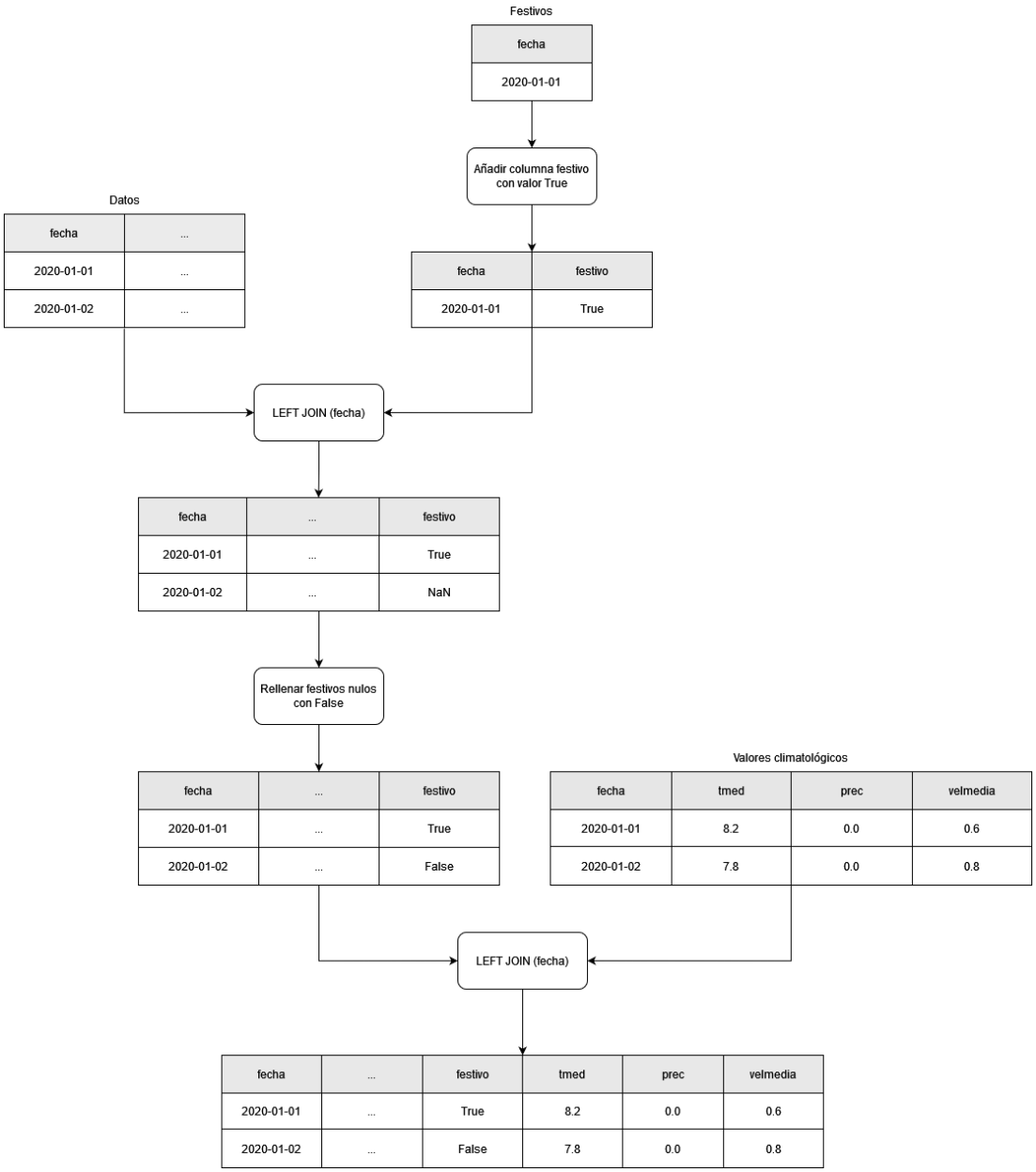

Figura 36: Datos meteorológicos y de festivos

Al finalizar la preparación de los datos, estos tienen el siguiente aspecto:

Variable	**Tipo**	**Descripción**	**Ejemplo**
fecha	datetime64	Id de la observación	*2017-01-03*
estacion	object	Nombre de la estación	*Campus Ciencias*
prestamos	float64	Número de bicicletas prestadas en la estación en la fecha	*1.0*
devoluciones	float64	Número de devoluciones de bicicletas registradas en la estación en la fecha	*2.0*
diaSemana	int64	Día de la semana correspondiente a la fecha. El lunes equivale a 0 y el domingo a 6.	*1*

dia	int64	Día del mes correspondiente a la fecha.	*3*
mes	int64	Mes. El 1 equivale a enero y el 12 a diciembre.	*1*
anno	int64	Año correspondiente a la fecha	*2017*
festivo	bool	Indica si el día fue festivo según el calendario laboral de Salamanca	*False*
tmed	float64	Temperatura media registrada en la fecha. (ºC)	*8.2*
prec	float64	Volumen de precipitaciones registrado en la fecha (mm)	*0.0*
velmedia	float64	Velocidad media del viento registrada en la fecha (m/s)	*0.8*

Tabla 12: Campos de los datos tras la preparación

8.4.5 Visualización de los datos

Una vez hemos ajustado los datos a nuestros propósitos, se han realizado varios gráficos para observar el comportamiento de estos, de cara a entrenar los modelos.

Como se han realizado modelos distintos para cada estación, se mostrará como ejemplo los gráficos obtenidos para la estación del Campus de Ciencias.

En primer lugar, se ha obtenido una visión general de las observaciones.

- Préstamos y devoluciones a lo largo del tiempo

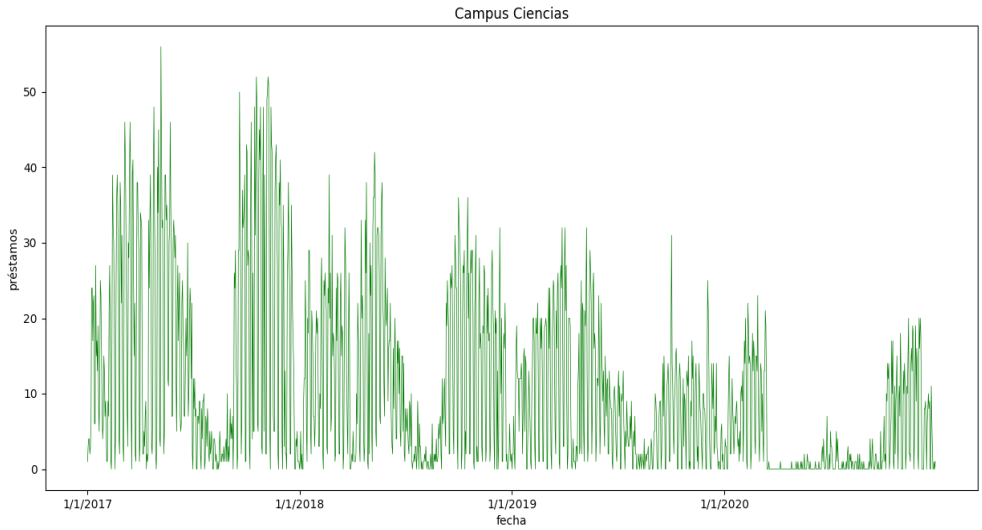

Figura 37: Uso de la estación a lo largo del tiempo

- Préstamos y devoluciones registrados en cada mes durante los cuatro años

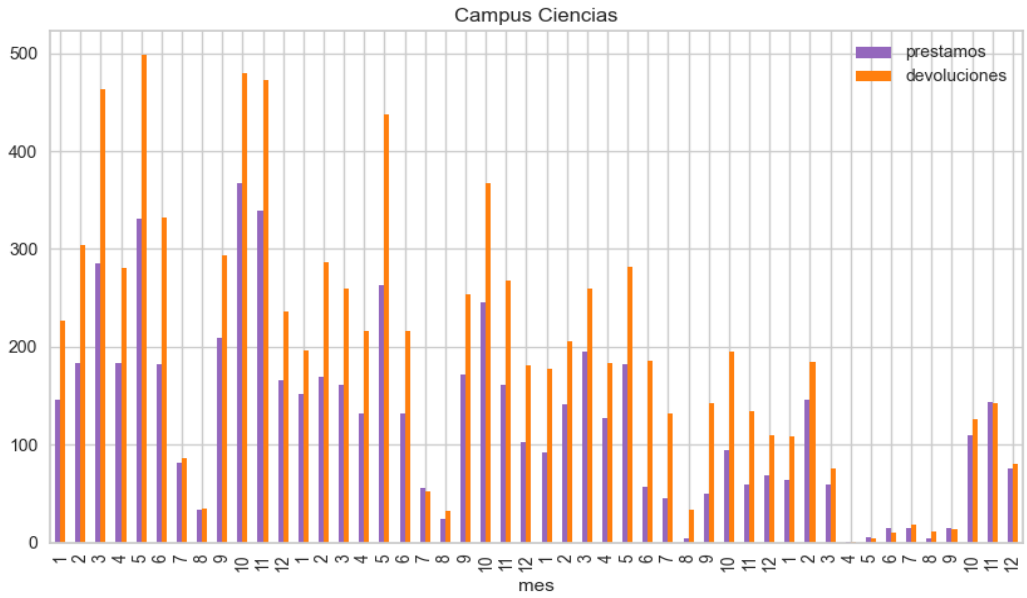

Figura 38: Préstamos y devoluciones por mes

Las variables discretas se han representado mediante diagramas de caja. Estos nos muestran varias medidas, descriptivas de la distribución de la variable a predecir con respecto a la variable independiente representada. Las cajas representan la mediana, los cuartiles y los *outliers* o valores atípicos. Las variables representadas son el día del mes, el mes, el año y si es festivo o no:

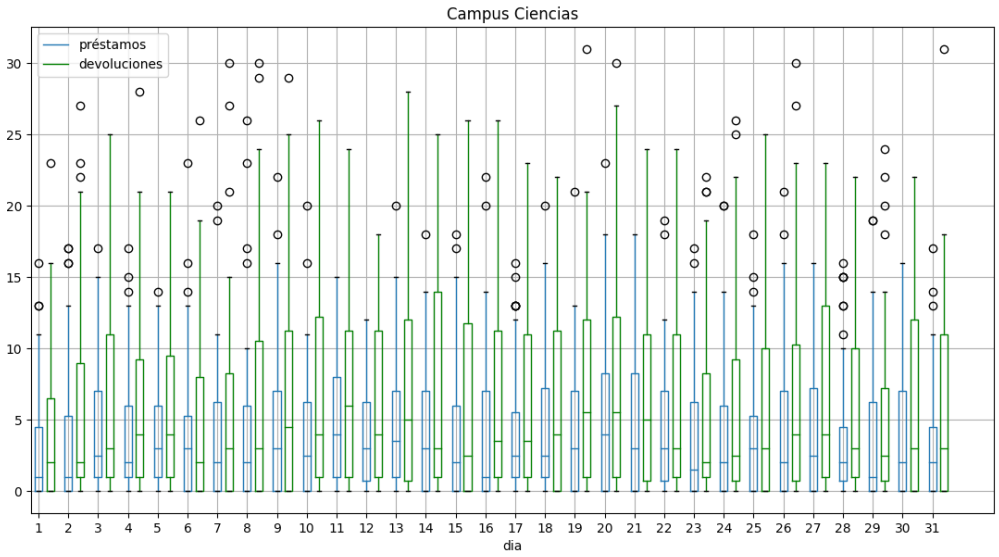

Figura 39: Relación entre la variable día y la variable a predecir

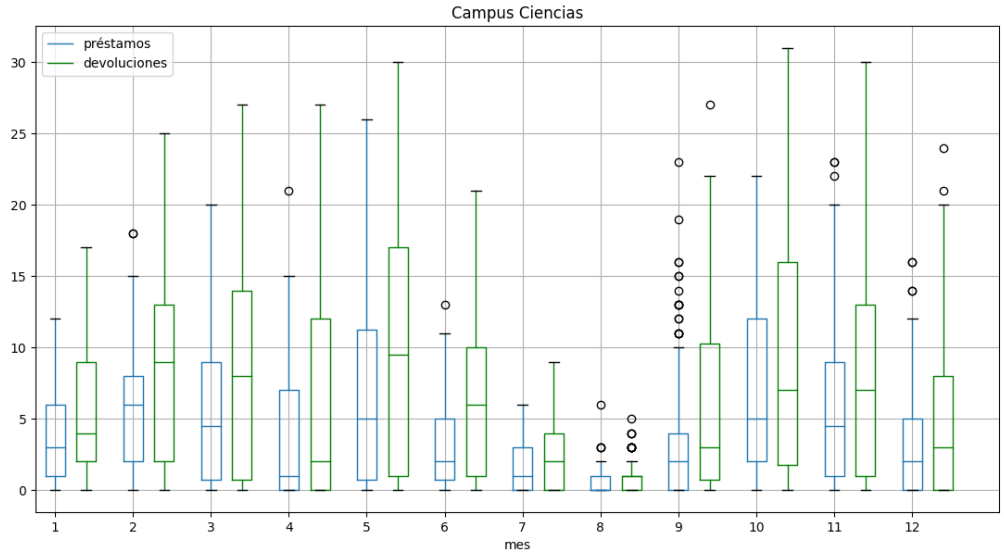

Figura 40: Relación entre la variable mes y la variable a predecir

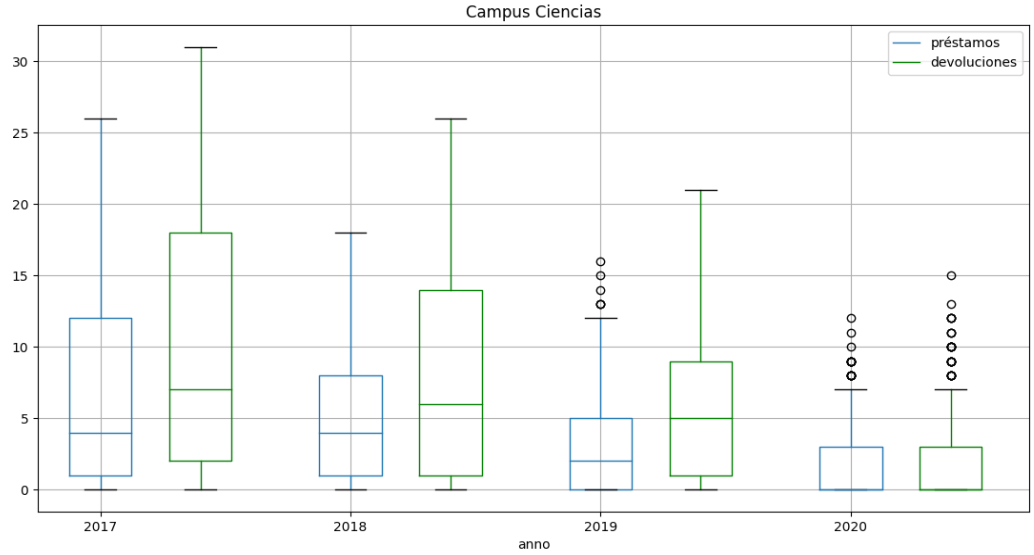

Figura 41: Relación entre la variable año y la variable a predecir

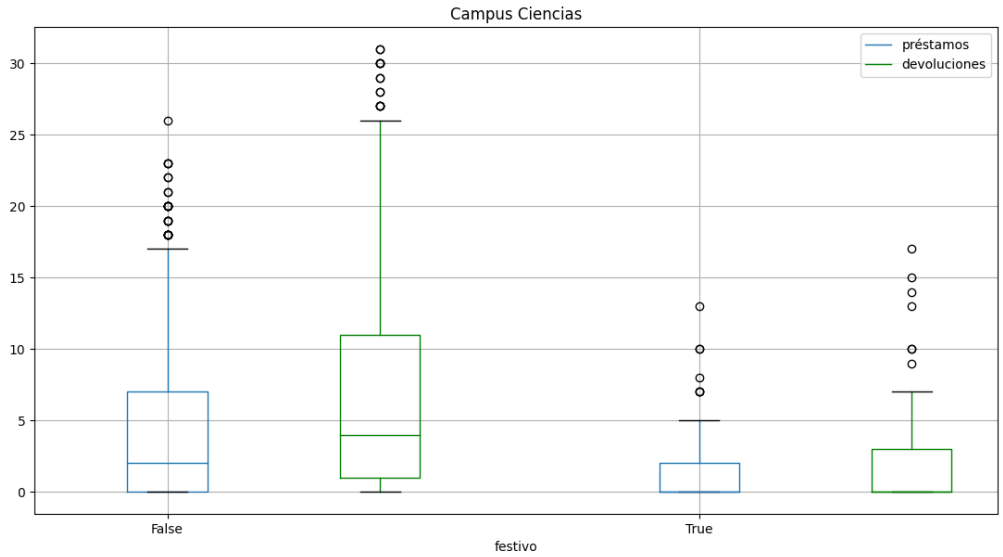

Figura 42: Relación entre la variable festivo y la variable a predecir

La relación entre las variables continuas independientes y la variable dependiente se ha representado mediante gráficos de dispersión. Estos gráficos se realizado de manera separada para los préstamos y devoluciones para facilitar su visualización. Las variables representadas son la temperatura media, las precipitaciones y la velocidad media del viento:

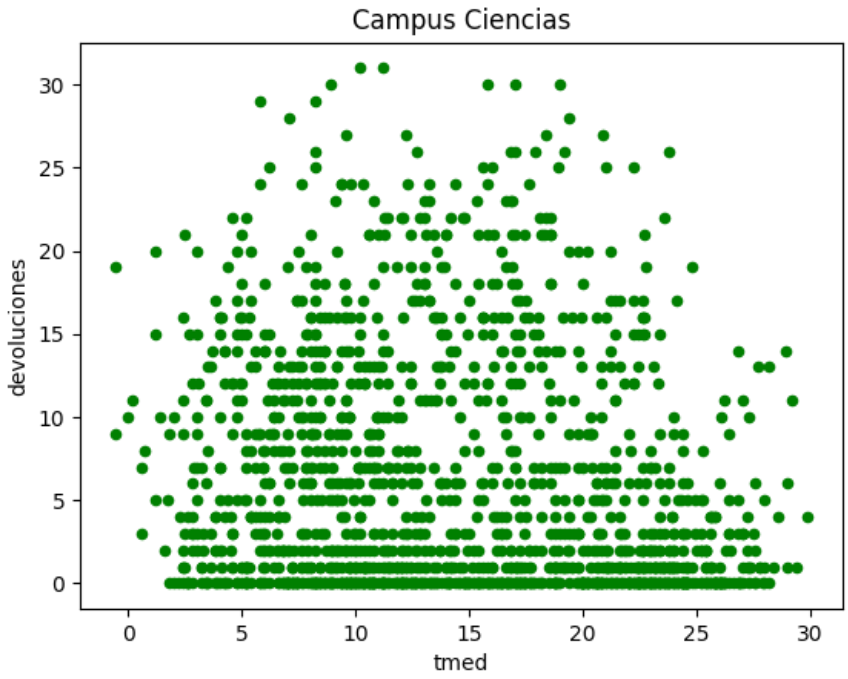

Figura 43: Relación entre la variable temperatura media y la variable a predecir

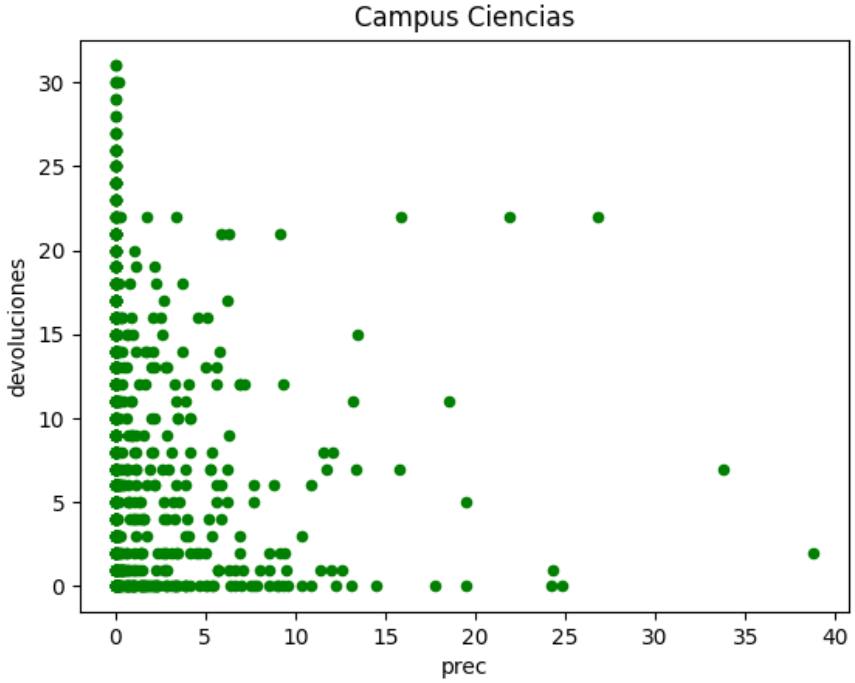

Figura 44: Relación entre la variable precipitaciones y la variable a predecir

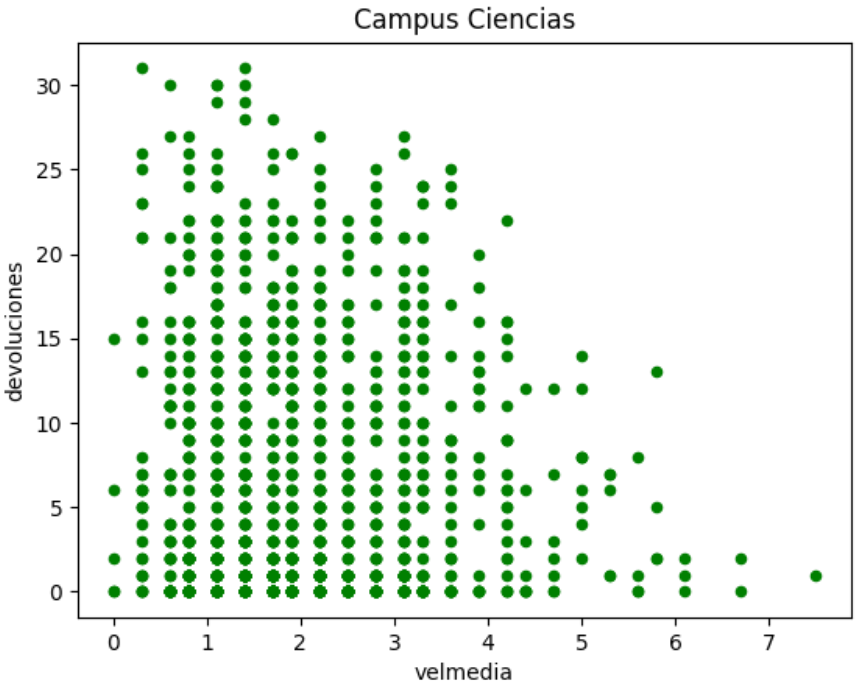

Figura 45: Relación entre la variable velocidad media y la variable a predecir

Se han utilizado también gráficos para observar la correlación entre las variables:

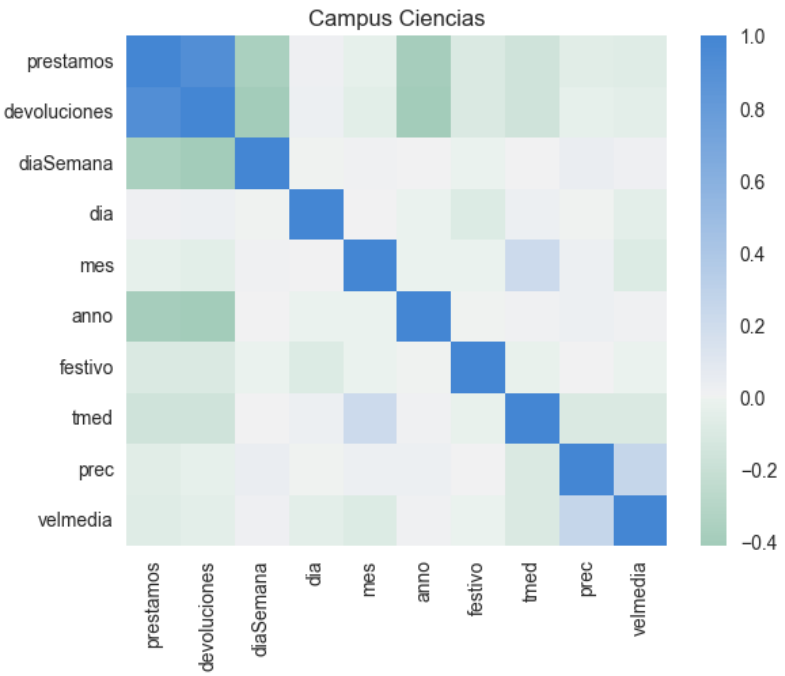

Figura 46: Correlación entre las variables

8.4.6 Entrenamiento y resultados

A continuación, se han entrenado los modelos y se han evaluado los resultados. Se han tomado algunas acciones sobre los datos buscando mejorar los resultados de los modelos, entre las que cabe destacar las siguientes:

- Se han eliminado los datos capturados en las fechas de mayor incidencia de la pandemia del Covid-19. Se ha considerado que se trata de una circunstancia excepcional que sesgaría los modelos.

- Se han eliminado los datos nulos

- Se ha escalado las variables para evitar que los datos con escalas más grandes tengan una mayor influencia en los modelos. Se ha comprobado que esta acción mejora los resultados de manera notable.

- Se ha probado a convertir a variables *dummy* algunas de las variables, como el mes y el día de la semana. Se ha observado que empeoraba los resultados.

Los modelos entrenados y sus resultados se describen en las siguientes secciones.

8.4.7 Regresión lineal

Representando los residuales vemos que el modelo de regresión lineal predice valores negativos, lo que no es coherente con las variables a predecir. Se podrían aplicar técnicas para corregir este problema, pero adicionalmente los resultados son peores que los de otros modelos. Por tanto, se ha decidido descartar este modelo. Los residuales son mayores para predicciones altas, lo que nos indica que es más difícil predecir los casos más altos. La distribución de los residuales es normal.

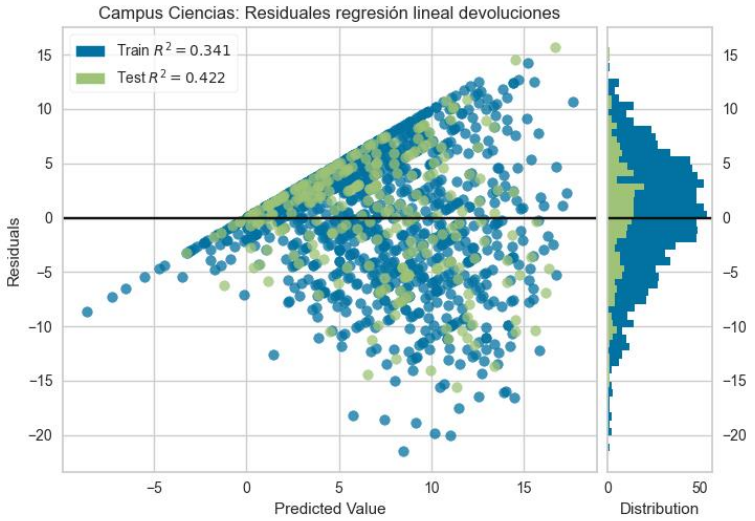

Figura 47: Resultados de la regresión lineal: residuales

En el gráfico de error compara el valor real con la predicción realizada. Vemos nuevamente que el modelo se ajusta mejor a los casos bajos y falla en las predicciones altas.

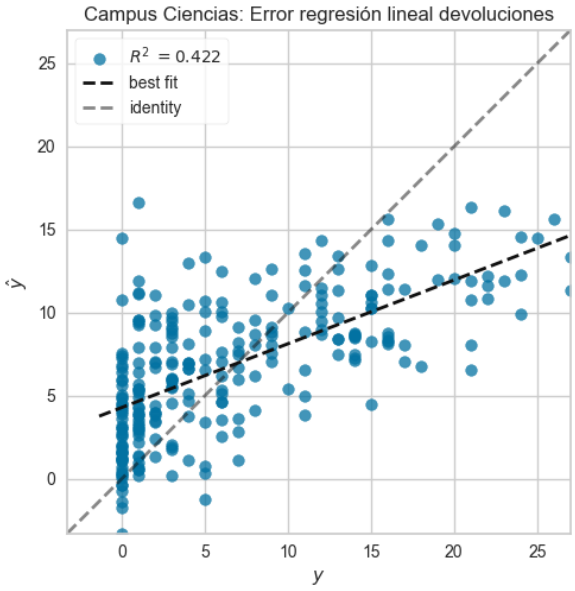

Figura 48: Resultados de la regresión lineal: error

8.4.8 Regresión polinómica

El modelo de regresión polinómica consigue unos resultados mejores que la regresión lineal, sin embargo, también predice valores negativos.

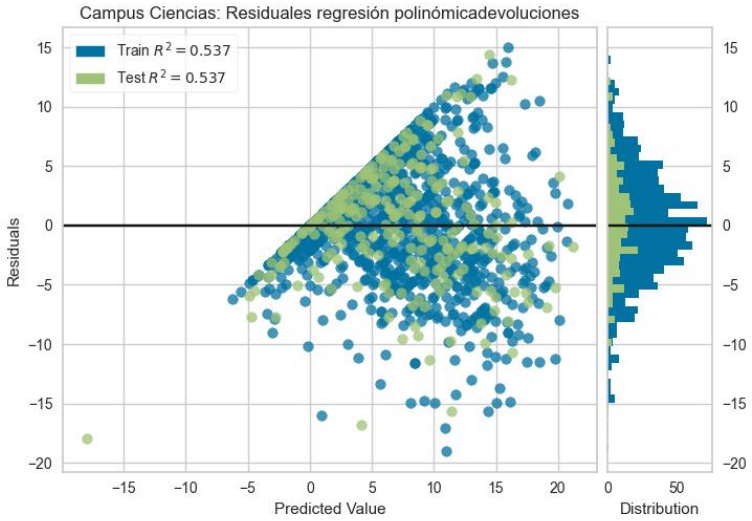

Figura 49: Resultados de la regresión polinómica: residuales

En el gráfico de error podemos ver claramente que las predicciones se ajustan mejor a la línea de los 45 grados. La mejora del modelo al utilizar regresión polinómica es coherente con los datos, ya que al realizar la visualización de la relación de las variables dependientes e independientes se observa que estas no tienen una tendencia lineal, sino más bien polinómica. Los mejores resultados se han obtenido con una ecuación polinómica de grado 2.

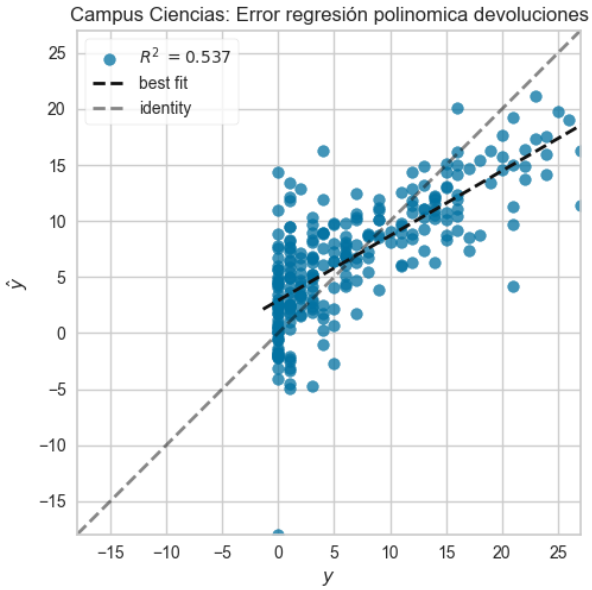

Figura 50: Resultados de la regresión polinómica: error

8.4.9 K Nearest Neighbors

Se ha observado que kNN produce unos buenos resultados, teniendo en cuenta que la variable a predecir está determinada por el comportamiento humano, y es por tanto más difícil de predecir que otros sistemas. Se ha considerado que kNN puede ser un buen modelo en este caso, ya que se basa en encontrar casos similares al caso a predecir. En el caso de un BSS es posible que en dos días con características similares se produzca una demanda parecida. También se ha observado que los resultados varían según el parametro k escogido. Los mejores resultados se han obtenido con un parámetro k 6. Se ha decidido escoger este modelo para realizar las predicciones.

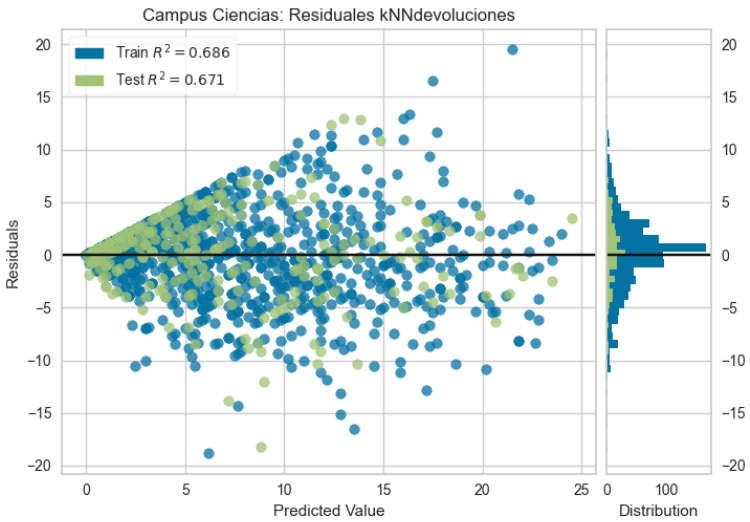

Figura 51: Resultados de kNN: residuales

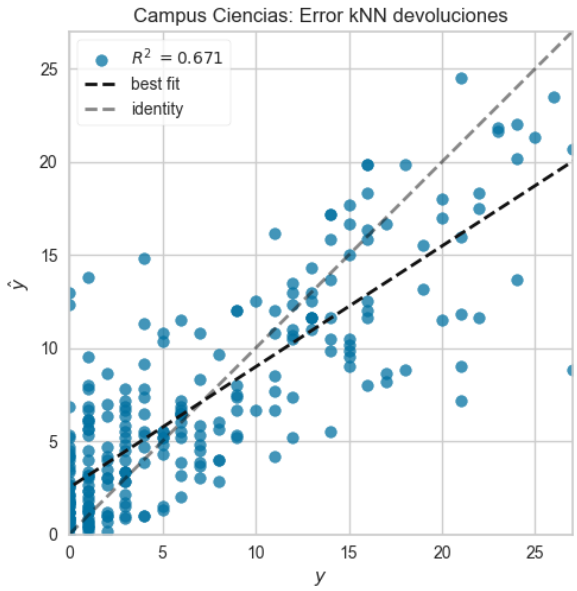

Figura 52: Resultados de kNN: error

8.4.10 SVM

Los resultados obtenidos utilizando SVM son también buenos, aunque como en los casos anteriores las predicciones más altas son peores. Los mejores resultados se han obtenido con un kernel RBF.

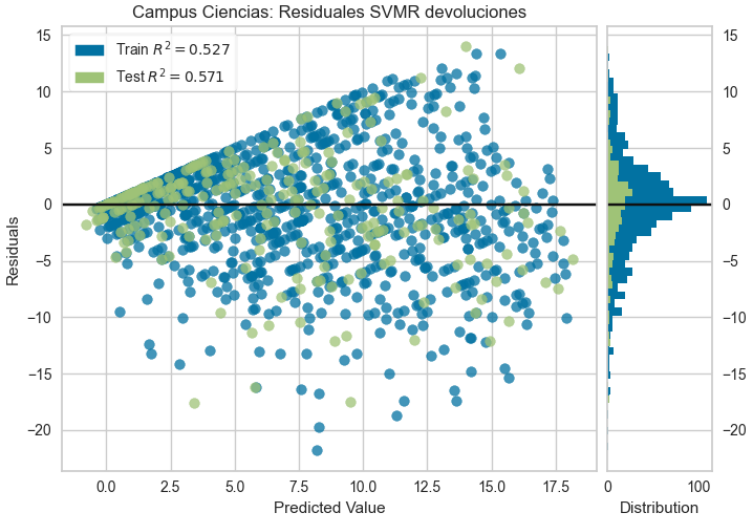

Figura 53: Resultados de SVM: residuales

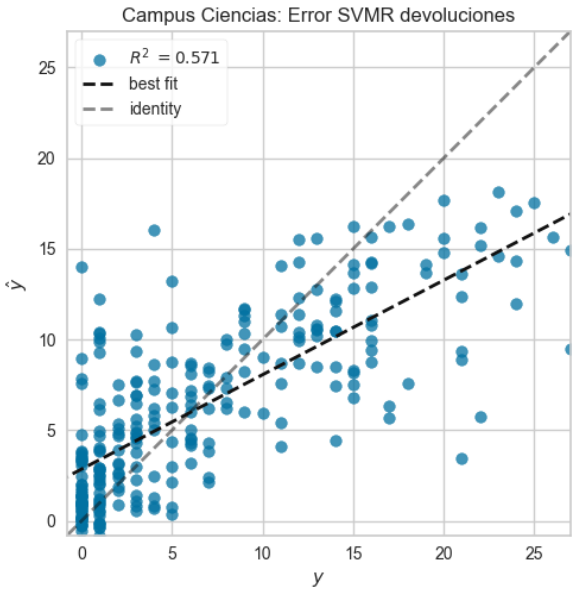

Figura 54: Resultados de SVM: error

8.4.11 AdaBoost

Aunque típicamente AdaBoost se entrena con un conjunto de árboles de decisión, los mejores resultados se han obtenido con regresores SVM. Sin embargo, los resultados no son mucho mejores que los de otros modelos más sencillos.

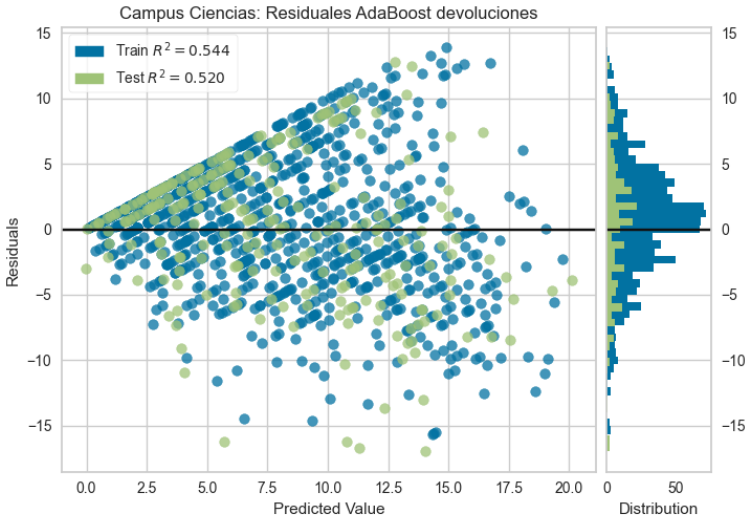

Figura 55: Resultados de AdaBoost: residuales

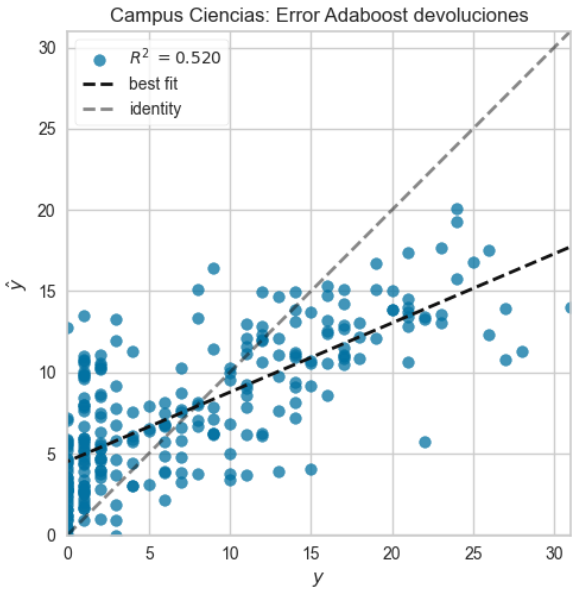

Figura 56: Resultados de Adaboost: error

8.5 Simulación

Para comprobar el impacto del sistema de balanceado en el BSS y los viajes de los usuarios se ha realizado una simulación mediante SUMO.

SUMO permite definir personas con varias etapas de viaje, utilizando distintos medios de transporte. La simulación se ha programado de tal manera que cada usuario realiza el trayecto del punto de partida a la estación de origen a pie, el trayecto entre estaciones utilizando un vehículo de tipo bicicleta, y el trayecto de la estación de destino al destino final nuevamente a pie.

El objetivo de la simulación es obtener las distancias recorridas por los usuarios y visualizar el estado del BSS en cada momento.

8.5.1 Preparación de los datos para la simulación

Para la preparación de los datos se ha empleado el siguiente algoritmo:

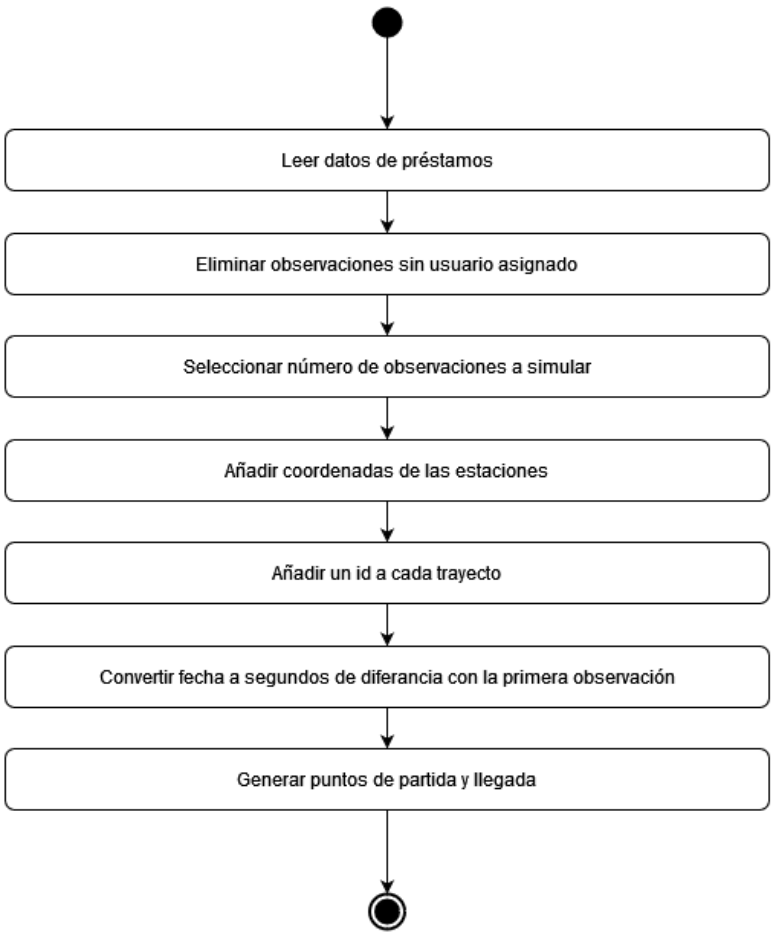

Figura 57: Preparación de los datos de la simulación

8.5.2 Generación de puntos de partida y llegada aleatorios

Los datos de los que disponemos no indican los puntos de partida y llegada deseados por los usuarios, únicamente las estaciones utilizadas. Se podría suponer que estas estaciones eran las más cercanas al lugar del que partieron, y el lugar al que querían llegar. Por tanto, para la simulación vamos a generar estos puntos de manera aleatoria en puntos cercanos a la estación.

Para ello se ha escogido una distancia máxima y se ha generado aleatoriamente un punto dentro del círculo cuyo radio corresponde la distancia máxima. Como esta distancia se define en metros, posteriormente hace falta proyectar este punto en las coordenadas geodésicas.

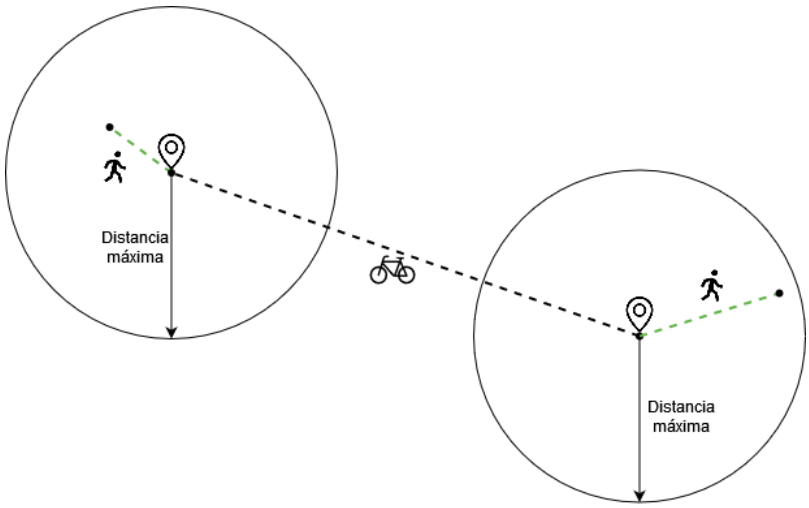

Figura 58: Generación de puntos de partida y llegada aleatorios

Para generar el punto aleatorio se ha calculado un valor aleatorio tanto para la distancia del punto a la estación (radio), como para la dirección (ángulo).

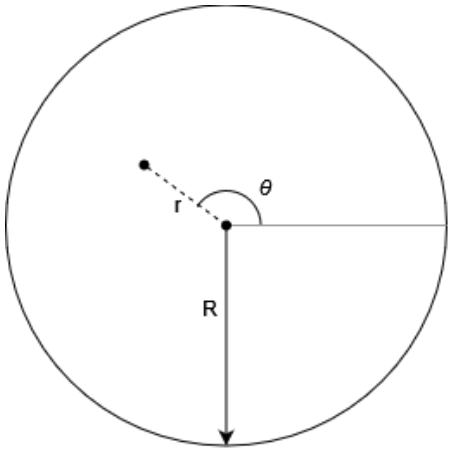

Figura 59: Generación de un punto aleatorio dentro de un círculo

Sin embargo, este método proporciona una distribución no uniforme, generándose más puntos en posiciones más cercanas al centro de manera exponencial. Para evitar este sesgo, uno de los métodos disponibles es generar un valor aleatorio entre 0 y 1, y multiplicar su raíz cuadrada por la distancia máxima.

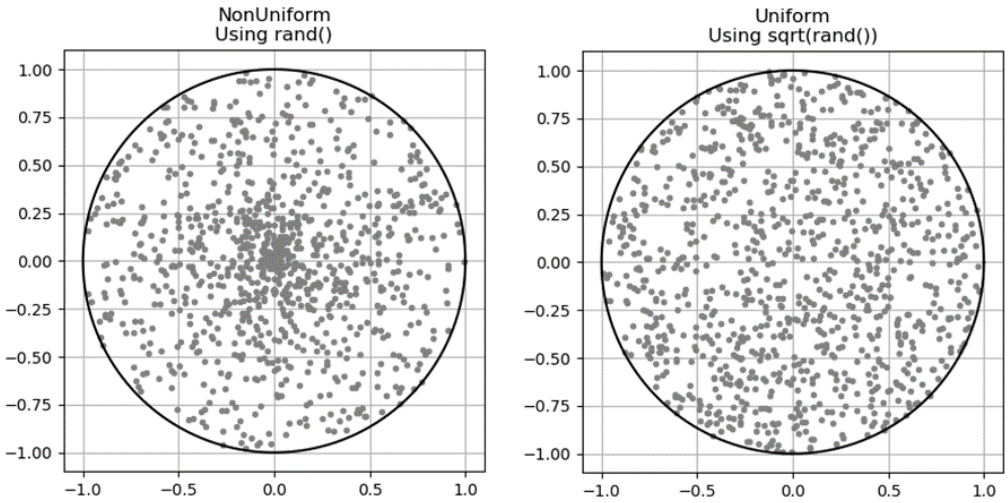

Figura 60: Distribución no uniforme y uniforme de puntos aleatorios (Generate uniform random points within a circle, s.f.)

Se ha optado por utilizar este método para generar una distribución uniforme, aunque una distribución no uniforme podría representar también correctamente la realidad, ya que es posible que los ciudadanos con estaciones cercanas a sus hogares o lugares frecuentados sean más propensos a utilizar el servicio.

8.5.3 Configuración de la simulación

Para crear el entorno de la simulación al que se conectará TraCi se ha utilizado un archivo de configuración. SUMO ofrece esta opción para facilitar la incorporación de toda la información sobre los distintos elementos de la simulación en un solo archivo. Este archivo es un xml que indica los archivos xml donde se define cada elemento. Los archivos creados y referenciados en la configuración son:

- Mapa: Archivo net.xml en el que se define la red de calles de la simulación.
- Vehículos: Dentro de un archivo rou.xml se han definido los tipos de vehículos bicicleta y peatón.
- Puntos de interés (POI): Se ha creado un script que transforma la información sobre las estaciones del archivo descriptor del sistema en un archivo add.xml con un punto de interés para cada estación. El identificador del punto de interés es el mismo que el de la estación. Además, se ha creado un icono que representa una estación de bicicletas y se ha asignado como imagen del POI.
- Opciones de visualización: En el archivo view.xml se indican los parámetros de visualización. Estos se han obtenido modificando las opciones de la interfaz gráfica de SUMO y exportando a un archivo la configuración realizada.

8.5.4 Mapa utilizado en la simulación

Para la realización de la simulación se decidió utilizar un mapa de las calles y carreteras de Salamanca obtenido mediante la herramienta OSMWizard. Sin embargo, se descubrió que este mapa no estaba fuertemente conectado, ya que muchas de sus calles no estaban conectadas con las demás, especialmente las calles peatonales.

Figura 61: Mapa obtenido de OSMWizard

En este caso, se necesita que todas las calles estén conectadas entre sí y se pueda llegar a cualquier punto del mapa desde otro. Si esto no sucede, se producen errores al calcular las rutas, y es imposible realizar la simulación.

Investigando sobre el problema, se descubrió un hilo sobre el tema en la página de problemas del repositorio de Sumo en GitHub. En él los desarrolladores indican tres opciones:

- Añadir las conexiones
- Excluir partes de la red
- Eliminar las partes de la red que no estén conectadas fuertemente

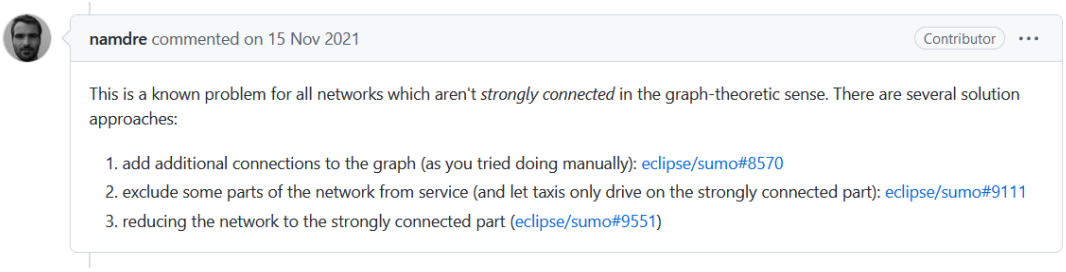

Figura 62: Hilo acerca del problema de la conexión débil de los mapas (SUMO issues: Missing connections between edges in Brunswick scenario, s.f.)

Respecto a las dos últimas soluciones, se utilizó la herramienta de SUMO netcheck para comprobar la conectividad del mapa, y se descubrió que el grupo de aristas conectadas entre sí más grande solo cubría el 39% del mapa. Eliminar el resto de las aristas implicaría perder gran parte de la información. Tampoco es una opción conectar manualmente los grupos de aristas, ya que la mayor parte de las aristas (más de 32000) son aristas individuales.

```
Total Edges: 54688
Largest Component: #0 Edge Count: 21579 Coverage: 39.46%

Edges   Incidence
1       32726
2       29
3       2
4       5
5       1
6       4
7       1
10      2
12      2
18      1
30      1
31      1
48      1
92      1
21579   1
```

Figura 63: Resultado netcheck

En cuanto a la primera solución, SUMO esta desarrollando una opción para su herramienta netcheck que solucionaría este problema. Esta opción se incluirá en la versión 2.0 de SUMO.

Figura 64: Cáptura de GitHub issues (SUMO issues: allow adding connector edges to avoid dead-ends, 2021)

Se decidió utilizar un mapa generado mediante la herramienta netgenerate en forma de cuadrícula para poder realizar la simulación.

8.5.5 Algoritmo de simulación

Para facilitar la realización de la simulación se han creado tres tablas para indicar los tres tipos de viajes: los que están sin iniciar, los que se encuentra en curso y los ya terminados.

La primera tabla contiene los viajes sin empezar. Esta se utiliza para determinar cuándo se debe introducir a los usuarios en la simulación. Los usuarios están ordenados según el segundo de la simulación en el que empiezan su viaje.

La segunda tabla contiene los viajes empezados y se utiliza para determinar la fase en la que se encuentran los usuarios.

La tercera tabla contiene los viajes ya finalizados.

Cada viaje contiene un campo fase que indica en qué parte del trayecto se encuentra el usuario:

- Fase 0: El usuario que aún no ha llegado a la estación de origen.
- Fase 1: El usuario se encuentra viajando en bicicleta de la estación de origen a la de destino.
- Fase 2: El usuario está desplazándose de la estación de destino al punto de llegada.
- Fase 3: El usuario ha llegado a su destino y ha finalizado el viaje.

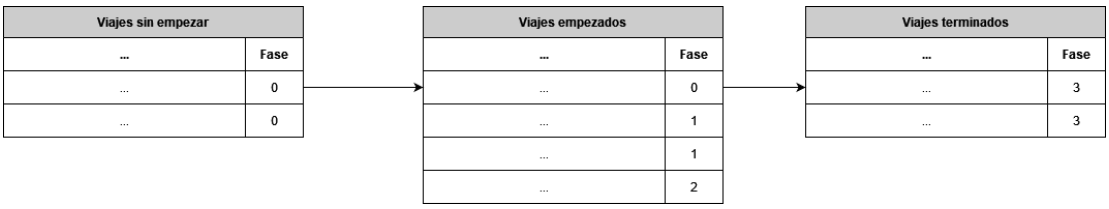

Figura 65: Tablas utilizadas en la simulación

El algoritmo de la simulación se desarrolla de la siguiente manera:

- En primer lugar, el script se conecta con SUMO y se abre la simulación.
- Se inicializan las variables y las tablas descritas anteriormente.
- El script obtiene información sobre el balanceado de las estaciones, la almacena y cambia los colores de los marcadores de las estaciones de acuerdo con el valor obtenido para cada una de ellas. Para mejorar el rendimiento de la simulación, este paso sólo se ejecuta al principio de la

simulación para obtener los valores iniciales y cuando se produce algún préstamo o devolución.

- Se comprueba si se ha producido la llegada de algún usuario a alguna estación mediante una función de TraCi que obtiene las personas que han terminado un trayecto en ese paso de la simulación. Si hay alguna llegada, se gestiona.

- Se comprueba en la primera tabla si hay algún viaje que deba iniciarse en el siguiente paso de la simulación. Si es así se prepara el viaje y se introduce en la simulación.

- Se ejecuta el siguiente paso de la simulación.

- Se comprueba si se han finalizado todos los trayectos.

- Si hay trayectos sin finalizar se repite el procedimiento.

- Cuando se han finalizado todos los trayectos se recopilan las estadísticas, tanto de la distancia recorrida por los usuarios como del balanceado del sistema durante la ejecución de la simulación, y se representan gráficamente. También se cierra la simulación.

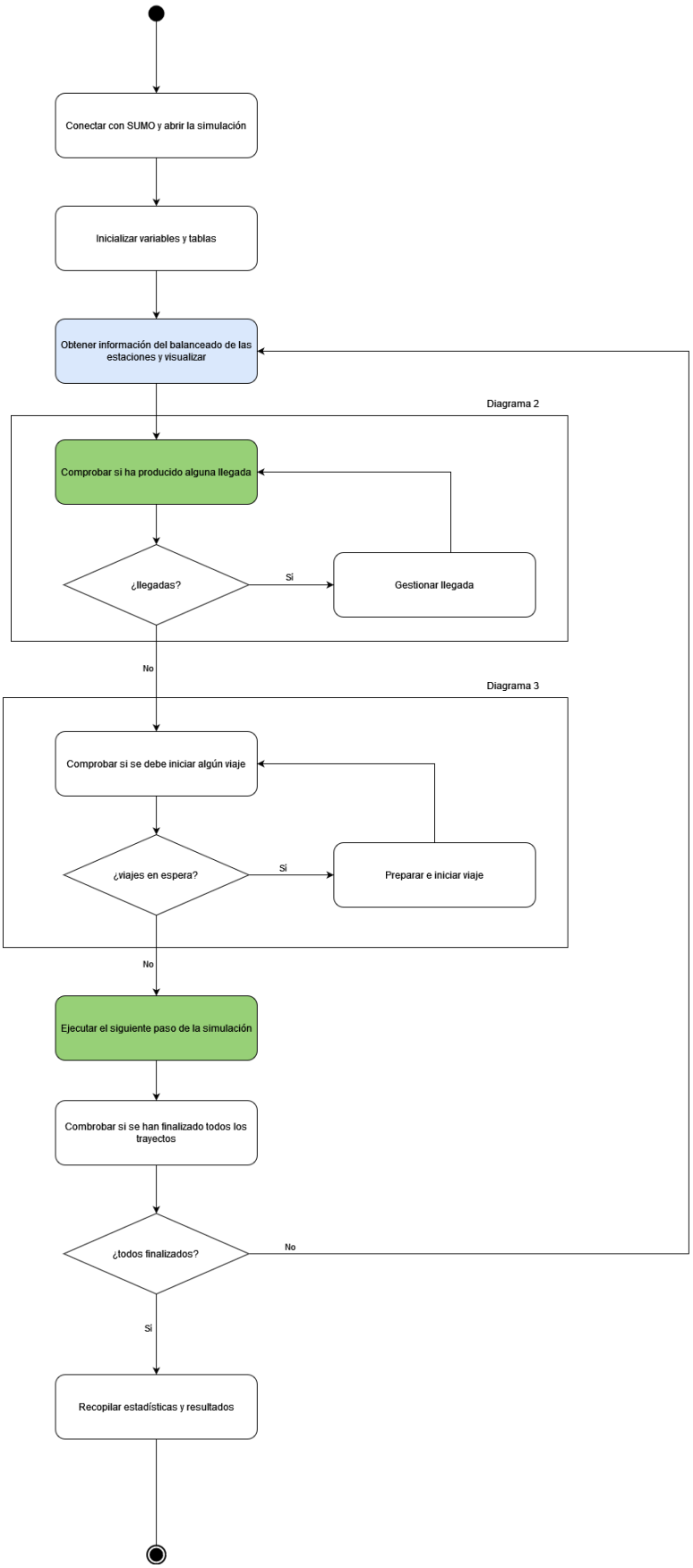

Figura 66: Algoritmo de la simulación: Diagrama 1

Cuando se produce una llegada de un usuario se comprueba el tipo de llegada mediante el número de fase del viaje en la tabla.

Si se trata de una llegada a la estación de origen:

- Se realiza una petición al sistema de balanceado para realizar el préstamo de la bicicleta. De esta forma el sistema registra que se ha producido un cambio en el número de bicicletas de la estación.
- Se calcula la ruta en bici a la estación de destino mediante una función de TraCi.
- Se inicia la ruta en la simulación.
- Se actualiza la información del trayecto. Es necesario cambiar el número de la fase en las tablas.
- Se registra la distancia calculada a recorrer en bici.

Si se trata de una llegada a la estación de destino:

- Se realiza una petición al sistema de balanceado para solicitar la devolución de la bicicleta.
- Se calcula la ruta a pie al destino final mediante TraCi
- Se inicia la ruta en la simulación
- Se actualiza la información de trayecto
- Se registra la distancia a recorrer a pie.

Si el usuario ha llegado a su destino final se actualiza su información y se añade el viaje a la tabla de viajes finalizados.

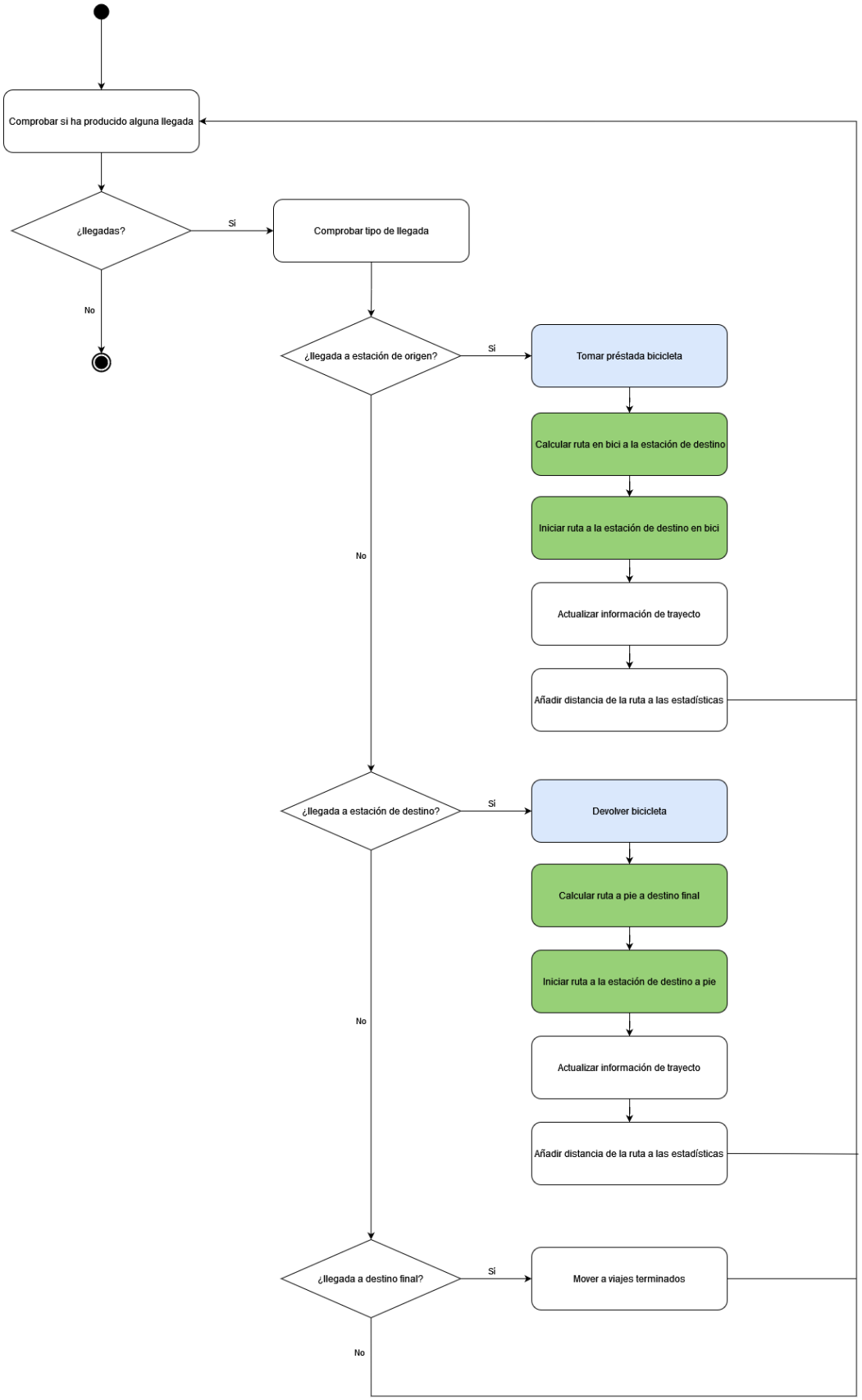

Figura 67: Algoritmo de la simulación: Diagrama 2

Cuando se detecta que se debe iniciar un nuevo viaje se ejecuta el siguiente algoritmo:

- Se comprueba si se está utilizando el sistema de balanceado para recomendar estaciones.

- Si se está utilizando se valora si se debe modificar la estación de origen o de destino de acuerdo con sugerencias de este.

- Se convierten las coordenadas de los puntos de llegada y de partida y de las estaciones a posiciones en el mapa de la simulación, determinadas por la calle y la posición dentro de esta más cercana a las coordenadas. Para ello se utiliza una función de TraCi.

- Se mueve el viaje de la tabla de viajes por empezar a la tabla de viajes empezados

- Se calcula la ruta a pie a la estación de origen, mediante una función de TraCi.

- Se inicia la ruta a la estación de origen en la simulación.

- Se registra la distancia que se va a recorrer a pie en las estadísticas de la simulación.

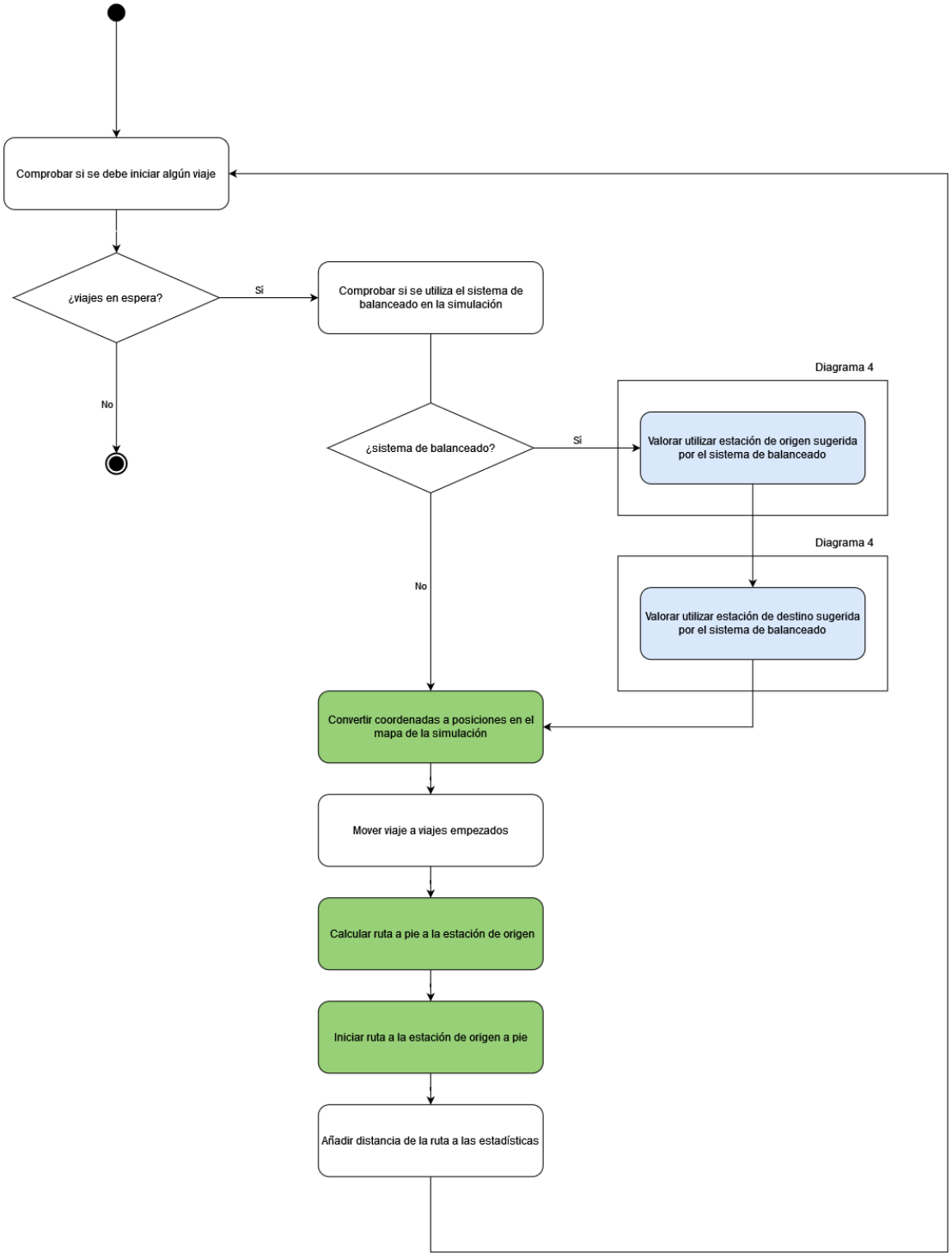

Figura 68: Algoritmo de la simulación: Diagrama 3

Para valorar si se acepta la propuesta del sistema se utiliza el siguiente algoritmo, que tiene en cuenta si la distancia a la estación propuesta es menor que la distancia definida como máxima.

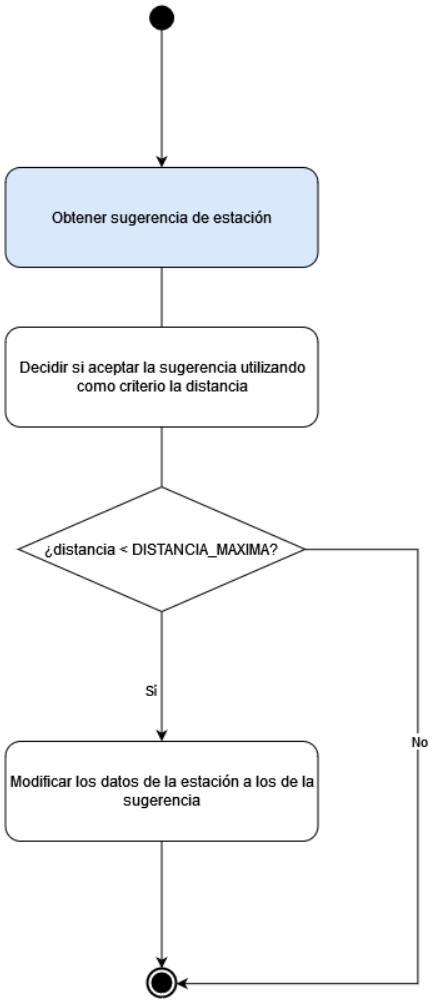

Figura 69:Algoritmo de la simulación: Diagrama 4

Cuando se acepta la propuesta, se sustituye la información en las tablas de la siguiente forma.

Id Trayecto	Estación de origen	Lat	Long	Estación de destino	Lat	Long	...
234	CAMPUS_CIENCIAS	40.9602863008297	-5.669999637335772	HUERTA_OTEA	40.9676948751129	-5.68365469151027	...

DISTANCIA_MAXIMA = 500

Estación de origen	Lat	Long	Distancia	...
PUENTE_ROMANO	40.9592164049522	-5.66930960336567	214	...

Figura 70: Sustitución de la estación original por la estación propuesta

9 RESULTADOS

9.1 Configuración de las simulaciones

Para comprobar el funcionamiento del sistema se ha realizado 6 simulaciones. En la primera de ellas no se ha utilizado el sistema. En las cinco restantes se han aceptado las recomendaciones del sistema en base a la distancia. En la segunda simulación se acepta la propuesta si la distancia a la estación es menor a 100 metros. La distancia se va incrementando hasta 600m. 600m es la distancia máxima a la que el sistema permite recomendar una estación, por lo que en esta simulación se han aceptado todas las propuestas.

El valor de 600m se ha escogido en base a los resultados del análisis de las distancias entre las estaciones.

La simulación se ha realizado con 200 viajes, realizados en un período aproximado de 55 horas. Gracias a las optimizaciones en la simulación, los tiempos de ejecución de la simulación son muy buenos, y se pueden realizar estas simulaciones en aproximadamente cinco minutos.

Todas las simulaciones se han realizado sobre los mismos datos, y con las mismas condiciones iniciales de balanceado del sistema. Los resultados se han almacenado para su posterior comparación.

9.2 Limitaciones de la simulación

Es habitual que, si la simulación se alarga, algunos usuarios lleguen a una estación y no puedan realizar el préstamo o la devolución, ya que el sistema se desequilibra con el paso del tiempo y no se reestablece el número de bicicletas en las estaciones por la noche, como sucede en la realidad.

Como la simulación no proporciona ningún mecanismo para que el usuario escoja otra estación en estos casos, se han contado estas ocurrencias. En las dos últimas simulaciones no se han producido errores ya que el sistema estaba perfectamente balanceado.

En la siguiente tabla se muestra la información de las simulaciones realizadas:

Nº	Condición aceptar propuesta	Préstamos fallidos	Devoluciones fallidas	Propuestas aceptadas
1	Nunca	0	2	0
2	< 100m	0	2	51
3	< 200m	1	7	209
4	< 300m	0	2	322
5	< 400m	0	0	383
6	< 600m	0	0	400

Tabla 13: Configuración de la simulación

Para la simulación no se ha utilizado el sistema de predicción de la demanda, ya que éste utiliza los datos meteorológicos y la fecha del día actual, y la simulación utiliza otras fechas. Se ha utilizado como valor óptimo de bicicletas la mitad de la capacidad de la estación.

9.3 Resultados

A continuación, se presentan los resultados de balanceado y distancia observados al concluir las simulaciones, y su interpretación. Estos resultados se pueden utilizar para mejorar el sistema, ya que nos muestran los aspectos que se deben controlar o variar.

9.3.1 Balanceado

Se ha observado que hay un impacto de los préstamos y devoluciones fallidos en el sistema: cuándo se producen, los resultados no son fiables. Se observa esta característica sobre todo en la tercera simulación, en la que se han producido 7 devoluciones fallidas.

Cómo durante la noche el servicio Salenbici no permite utilizar las bicicletas, en esos períodos de tiempo no se han producido cambios en el balanceado.

A continuación, se muestra el gráfico con los resultados del balanceado del sistema. Este resultado se calcula como la media del balanceado de todas las estaciones, utilizando el valor absoluto y normalizado entre 0 y 1. El mejor resultado es 0, que indicaría el equilibrio perfecto en todas las estaciones. 1 es el peor resultado, indicando que todas las estaciones están completas o vacías.

Cómo muchas de las estaciones no se llegan a utilizar (los préstamos tienden a concentrarse en las mismas estaciones céntricas), estas se mantienen en equilibrio y mitigan los efectos en el valor del desequilibrio producido en las estaciones centrales. Sin embargo, observando la simulación, se puede comprobar que el uso del sistema contribuye a mejorar en gran medida el balanceado del sistema en el centro de la ciudad.

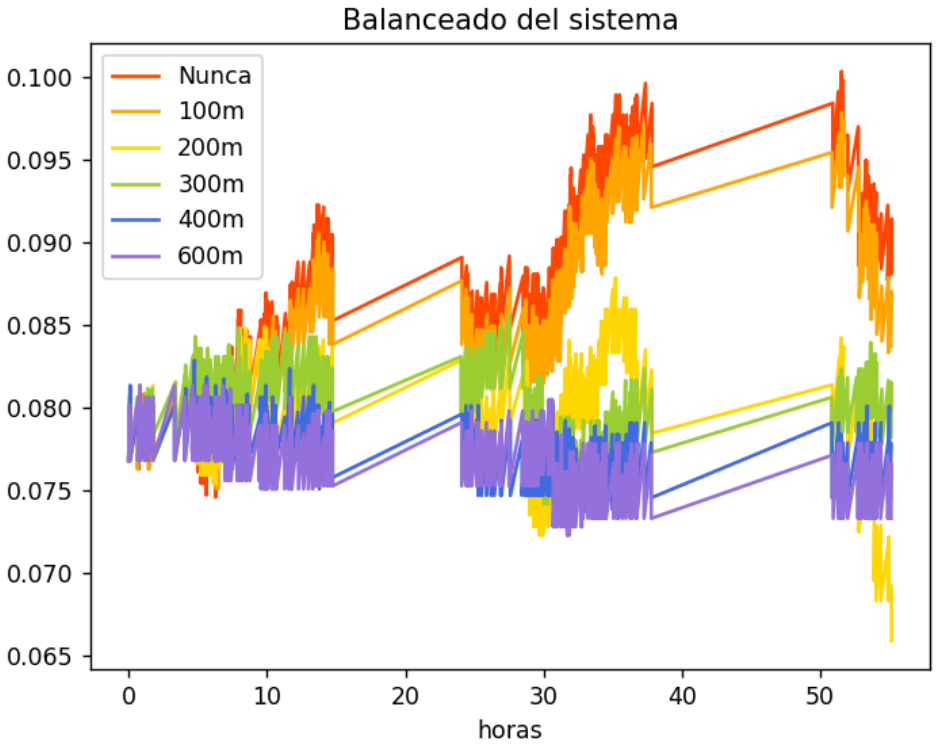

Figura 71: Resultados de balanceado del sistema

Los resultados visuales de las simulaciones nos muestran más información sobre el balanceado del sistema.

En la siguiente imagen se muestra el estado de las estaciones tras realizar la simulación sin utilizar las recomendaciones del sistema de balanceado.

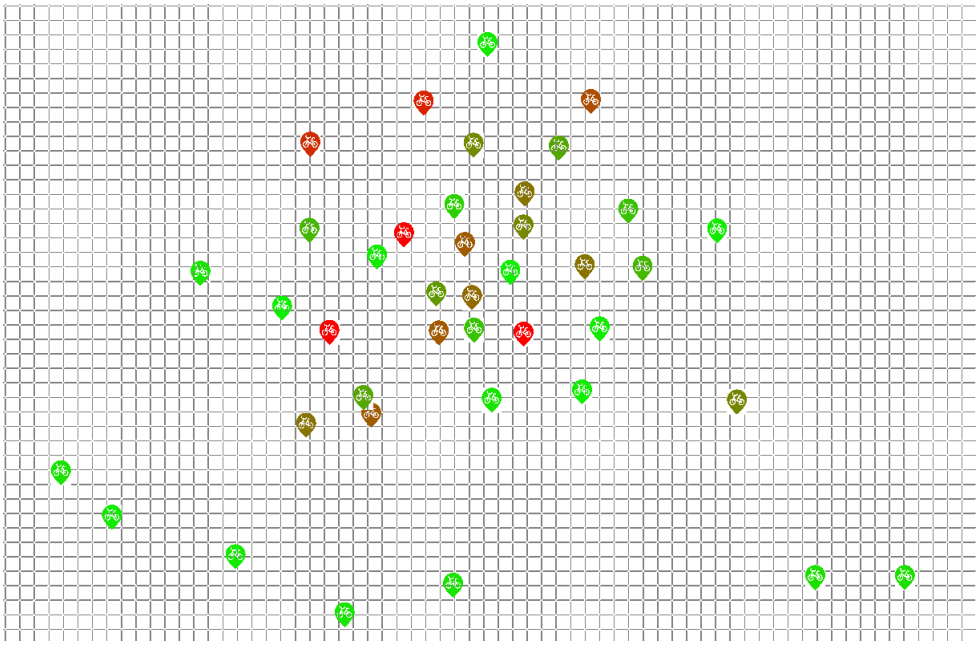

Figura 72: Simulación sin utilizar las recomendaciones del sistema de balanceado

Lo podemos comparar con el estado del sistema al finalizar la simulación cuándo el usuario acepta las sugerencias si estas están a menos de 200 metros. El cambio es notable.

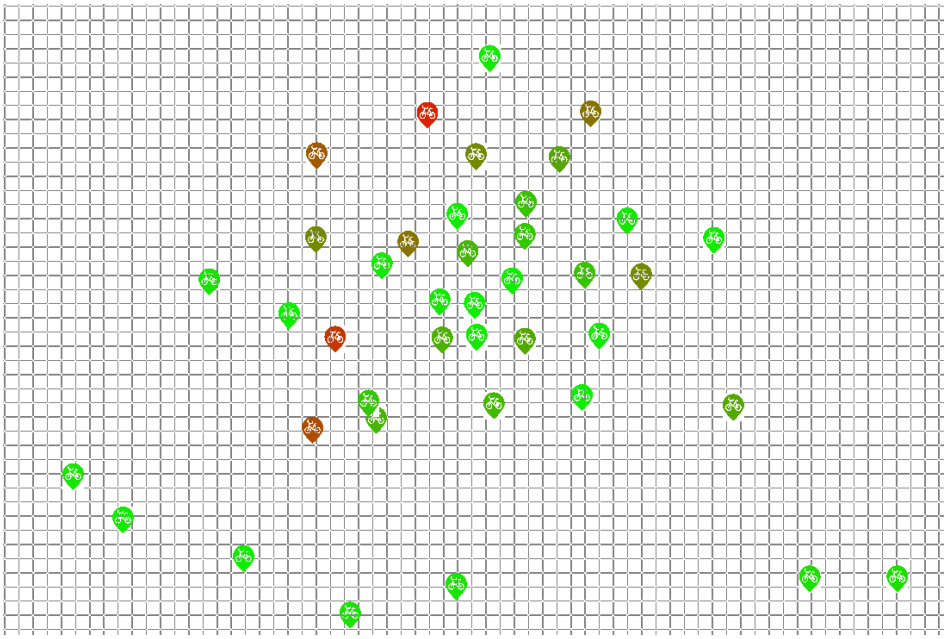

Figura 73: Simulación con distancia máxima 200m

El estado del sistema cuando se aceptan todas las peticiones es prácticamente óptimo. Sin embargo, esta situación es irreal, ya que requeriría la colaboración absoluta de todos los usuarios, aceptando distancias muy grandes.

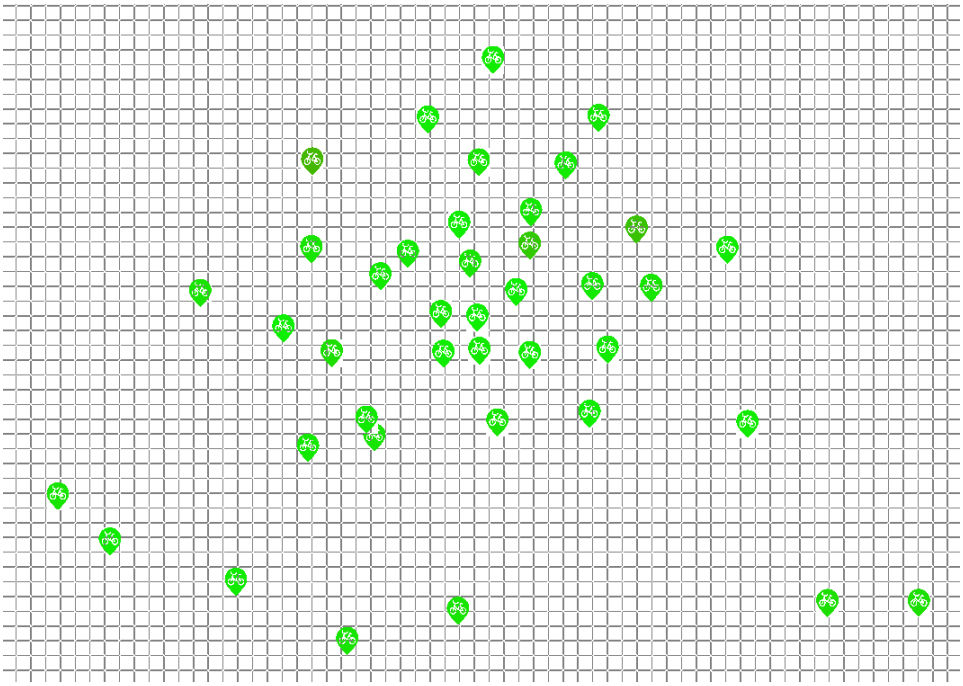

Figura 74: Simulación con distancia máxima 600m

La simulación permite también observar a su finalización el balanceado de cada estación individualmente.

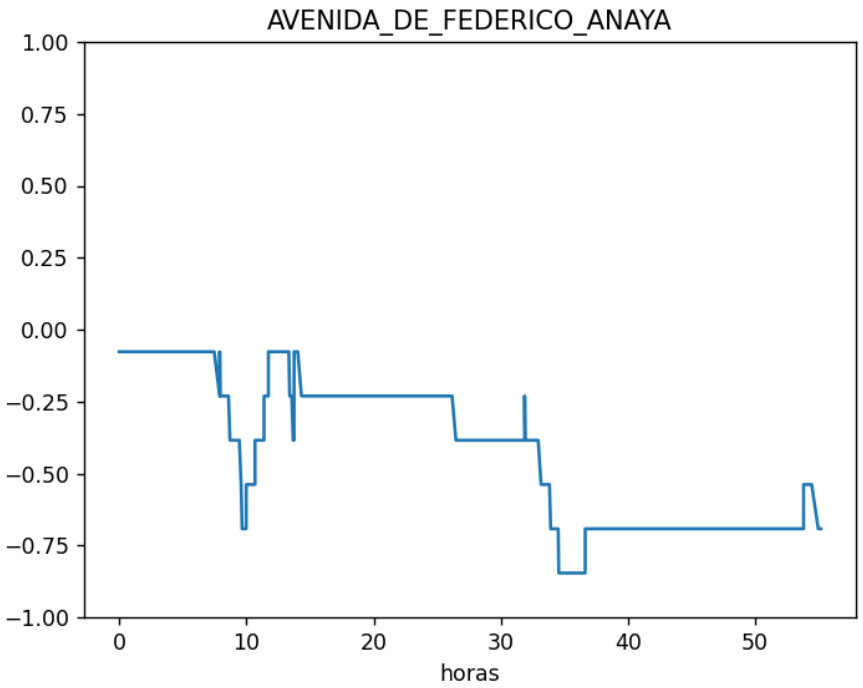

Figura 75: Balanceado de una estación

9.3.2 Distancia

Se han almacenado y representado también los resultados de distancia media recorrida por los usuarios.

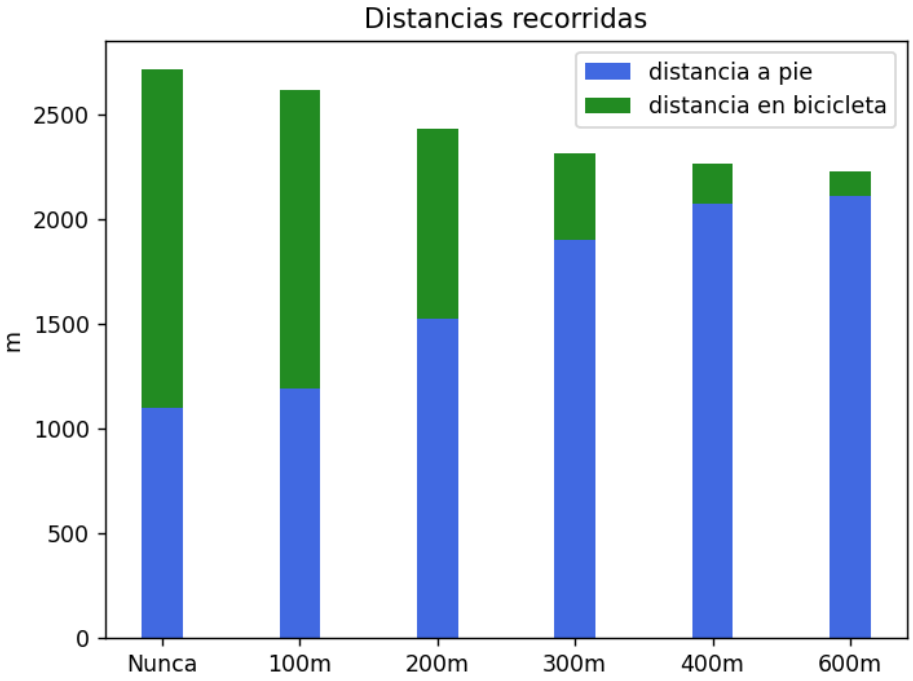

Figura 76: Resultados de distancias recorridas

Estos resultados no son favorables cuando la distancia es mayor de 100 metros, ya que la distancia recorrida a pie es mayor que la distancia en bicicleta, lo que no motivaría a los usuarios a cambiar de estación.

Una posible causa es que el sistema recomiende la estación de origen como estación de destino cuando el usuario quiere realizar un viaje corto, ya que esta puede estar dentro de la distancia máxima definida. En los resultados de balanceado se han observado préstamos y devoluciones instantáneas cuando la distancia es alta, lo que indicaría la ocurrencia de este suceso.

Es posible también que la distancia máxima de los puntos generados para el origen y el destino fuese excesivamente alta, ya que la distancia recorrida a pie es más alta de lo esperado para el caso base, especialmente en relación con la distancia recorrida en bicicleta. Se propone como mejora futura realizar simulaciones variando este parámetro, para observar el efecto en los resultados.

10 CONCLUSIONES

10.1 Aportaciones de este trabajo

Este trabajo realiza las siguientes aportaciones:

- Se ha elaborado un sistema de balanceado de un BSS basado en la recomendación de estaciones a los usuarios mediante un sistema multi-agente.
- Se ha entrenado un sistema de predicción de la demanda de las estaciones a partir de datos de uso históricos.
- Se ha demostrado que la simulación es un buen método para analizar este tipo de sistemas, ya que permite observar los errores y las mejoras que se pueden realizar en el calibrado del sistema. Adicionalmente, permite visualizar su funcionamiento en tiempo real.
- Se han logrado unos buenos resultados de rendimiento. En la simulación se han logrado simular 50 horas de funcionamiento del servicio en 5 minutos, gracias a la optimización del algoritmo de simulación planteado. En el sistema de balanceado, se han introducido mecanismos de concurrencia para mejorar los tiempos de respuesta de las peticiones.

10.2 Futuras mejoras

Según lo observado en los resultados, se podrían introducir las siguientes mejoras:

- Incorporar el mapa de Salamanca, una vez se puedan arreglar los problemas de conexión de las calles entre sí.
- Mejorar los algoritmos de recomendación de estaciones.
- Mejorar la predicción de la demanda: una idea sería utilizar métodos que tengan en cuenta el carácter temporal de los datos y den mayor peso a las observaciones más recientes.
- Realizar simulaciones con condiciones distintas a la distancia, por ejemplo, añadiendo un factor aleatorio.
- Variar los parámetros de la simulación de acuerdo con lo observado en los resultados para calibrar el sistema.
- Realizar simulaciones incorporando la predicción de la demanda.

Por otro lado, en este trabajo se ha desarrollado únicamente el sistema de balanceado con el objetivo de evaluar su rendimiento. En el futuro se podría desarrollar el sistema para su uso, añadiendo los siguientes componentes:

- Integración con el sistema del BSS
- Reentrenamiento automático de los modelos: Añadir un registro de los préstamos y reentrenar cada cierto tiempo los modelos para reflejar las nuevas tendencias.
- Gestión de puntos: Gestionar la asignación de puntos al usuario cuando utilice las estaciones sugeridas. Se propone utilizar una base de datos y un sistema de códigos que se obtengan al aceptar la sugerencia y se validen al realizar el préstamo. Habría que evitar también el uso fraudulento del sistema.
- Interfaz gráfica: Crear una interfaz gráfica para que el sistema pueda ser monitorizado por un empleado del servicio.
- Mejoras en la seguridad de las conexiones: Implementar un mecanismo de autentificación como OAuth, no mostrar información personal en las URIs…

También se podría desarrollar la aplicación móvil o web que proporcione al usuario la capacidad de utilizar el sistema. Las interacciones con el sistema se han diseñado utilizando la arquitectura REST y el estándar JSON para facilitar el desarrollo posterior de dicha aplicación.

10.3 Conclusiones finales

En conclusión, este trabajo presenta las bases para la creación de un sistema de balanceado de estaciones de un BSS mediante la colaboración del usuario utilizando agentes, un método para predecir la demanda de las estaciones e incorporarla al balanceado, y una herramienta para evaluar el funcionamiento del sistema a partir de casos reales de uso. Estos componentes se podrían integrar en el sistema de gestión de un BSS o ser utilizados para evaluar distintos algoritmos de balanceado.

A nivel personal, este trabajo ha supuesto una experiencia muy enriquecedora, que me ha permitido utilizar los conocimientos adquiridos en la carrera para realizar un proyecto interesante y con utilidad. Las lecciones aprendidas en la realización de este trabajo han sido numerosas y serán, con seguridad, de gran utilidad en el futuro.

REFERENCIAS Y BIBLIOGRAFÍA

AEMET. (s.f.). *Opendata AEMET*. Obtenido de https://opendata.aemet.es/centrodedescargas/inicio

Aeschbach, P., Zhang, X., Georghiou, A., & Lygeros, J. (2015). Balancing bike sharing systems through customer cooperation - a case study on London's Barclays Cycle Hire. *2015 54th IEEE Conference on Decision and Control (CDC)*, (págs. 4722-4727). doi:10.1109/CDC.2015.7402955

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., . . . Zimmermann, T. (2019). *Software Engineering for Machine Learning: A Case Study.* Microsoft. Obtenido de https://www.microsoft.com/en-us/research/uploads/prod/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf

Ayuntamiento de Salamanca. (s.f.). Obtenido de http://www.salamancasalenbici.com/

Bellini, P., Nesi, P., & Pantaleo, G. (2022). IoT-Enabled Smart Cities: A Review of Concepts, Frameworks and Key Technologies. *Applied Sciences, 3*(1607). doi:10.3390/app12031607

Butler, L., Yigitcanlar, T., & Paz, A. (2020). Smart Urban Mobility Innovations: A Comprehensive Review and Evaluation. *IEEE Access, 8*, 196034-196049. doi:0.1109/ACCESS.2020.3034596

Cardoso, R. C., & Ferrando, A. (2021). A Review of Agent-Based Programming for Multi-Agent Systems. *Computers, 2*(16). doi:10.3390/computers10020016

Dorokhin, S., Artemov, A., Likhachev, D., Novikov1, A., & Starkov, E. (2020). Traffic simulation: an analytical review. *IOP Conf. Series: Materials Science and Engineering*, *918.* doi:10.1088/1757-899X/918/1/012058

Durán Toro, A., & Bernárdez Jiménez, B. (2000). *Metodología para la Elicitación de Requisitos de Sistemas Software.* Obtenido de http://www.lsi.us.es/docs/informes/lsi-2000-10.pdf

Eclipse Foundation. (s.f.). *Eclipse*. Obtenido de https://www.eclipse.org/

Eclipse Foundation. (s.f.). *SUMO*. Obtenido de https://www.eclipse.org/sumo/

Eclipse Jersey. (s.f.). Obtenido de https://eclipse-ee4j.github.io/jersey/

Fernández, A., Billhardt, H., Timón, S., Ruiz, C., Sánchez, Ó., & Bernabé, I. (2019). Balancing Strategies for Bike Sharing Systems. En M. Lujak (Ed.), *Agreement Technologies. AT 2018. Lecture Notes in Computer Science, vol 11327* (págs. 208–222). Springer, Cham. doi:https://doi.org/10.1007/978-3-030-17294-7_16

FIPA TC Agent Management. (2004). *FIPA Agent Management Specification.* Obtenido de http://www.fipa.org/specs/fipa00023/SC00023K.pdf

FIPA TC Communication. (2002). *FIPA ACL Message Structure Specification.* Obtenido de http://www.fipa.org/specs/fipa00061/SC00061G.pdf

FIPA TC Communication. (2002). *FIPA Communicative Act Library Specification.* Obtenido de http://www.fipa.org/specs/fipa00037/SC00037J.pdf

FIPA TC Communication. (2002). *FIPA Subscribe Interaction Protocol Specification.* Obtenido de http://www.fipa.org/specs/fipa00035/SC00035H.pdf

Foundation for Intelligent Physical Agents. (2005). *FIPA Specifications*. Obtenido de http://www.fipa.org/specifications/index.html

Gaspero, L. D., Rendl, A., & Urli, T. (2013). Constraint-Based Approaches for Balancing Bike Sharing Systems. En C. Schulte (Ed.), *Principles and Practice of Constraint Programming. 19th International Conference CP 2013*, págs. 758-773. Uppsala, Sweden: Springer, Berlin, Heidelberg. doi:https://doi.org/10.1007/978-3-642-40627-0_56

Generate uniform random points within a circle. (s.f.). Obtenido de https://meyavuz.wordpress.com/2018/11/15/generate-uniform-random-points-within-a-circle/

Google. (s.f.). *Google Maps*. Obtenido de https://www.google.es/maps/

Gu, D., Andreev, K., & Dupre, M. E. (2021). Major Trends in Population Growth Around the World. *China CDC Weekly, 3*(28), 604-613. doi: 10.46234/ccdcw2021.160

Introducción a JSON. (s.f.). Obtenido de https://www.json.org/json-es.html

JADE. (s.f.). Obtenido de https://jade.tilab.com/

JetBrains. (s.f.). *PyCharm*. Obtenido de https://www.jetbrains.com/es-es/pycharm/

JGraph LTD. (s.f.). *Diagrams.net*. Obtenido de https://www.diagrams.net/

Kearney, P., Stark, J., Caire, G., Garijo, F. J., Sanz, J. J., Pavon, J., . . . Massonet, P. (2001). *MESSAGE: Methodology for Engineering Systems of Software Agents.* Obtenido de http://www.upv.es/sma/teoria/metodologias/articulos/D3finalReviewed.pdf

Lara, A. P., Costa, E. M., Furlani, T. Z., & Yigitcanla, T. (2016). Smartness that matters: towards a. *ournal of Open Innovation: Technology, Market, and Complexity, 2*(8). doi:10.1186/s40852-016-0034-z

Li, G., Cao, N., Zhu, P., Zhang, Y., Zhang, Y., Li, L., . . . Zhang, Y. (2021). Towards Smart Transportation System: A Case Study on the Rebalancing Problem of Bike Sharing System Based on Reinforcement Learning. (S.-B. Tsai, Ed.) *Journal of Organizational and End User Computing, 33*(3). doi:10.4018/JOEUC.20210501.oa3

Liu, J., Li, Q., Qu, M., Chen, W., Yang, J., Xiong, H., . . . Fu, Y. (2015). Station Site Optimization in Bike Sharing Systems. *2015 IEEE International Conference on Data Mining*, (pp. 883-888). doi:10.1109/ICDM.2015.99

Luo, M., Du, B., Klemmer, K., Zhu, H., & Wen, H. (2022). Deployment Optimization for Shared e-Mobility Systems With Multi-Agent Deep Neural Search. *IEEE Transactions on Intelligent Transportation Systems, 23*(3), 2549-2560. doi:10.1109/TITS.2021.3125745

Matplotlib. (s.f.). Obtenido de https://matplotlib.org/

Medium. (s.f.). *Towards Data Science*. Obtenido de https://towardsdatascience.com/

Mozilla Foundation. (s.f.). *Introducción a XML*. Obtenido de https://developer.mozilla.org/es/docs/Web/XML/XML_introduction

Open Street Maps. (s.f.). Obtenido de https://www.openstreetmap.org/about

Oracle. (s.f.). *Java*. Obtenido de https://www.java.com/es/

P. Kasznar, A., W. A. Hammad, A., Najjar, M., Linhares Qualharini, E., Figueiredo, K., Pereira Soares, C., & N. Haddad, A. (2021). Multiple Dimensions of Smart Cities' Infrastructure: A Review. *Buildings, 11*(73). doi:10.3390/buildings11020

Pallets. (s.f.). *Flask*. Obtenido de https://flask.palletsprojects.com/en/2.1.x/

Pandas. (s.f.). Obtenido de https://pandas.pydata.org/

Poslad, S. (2007). Specifying Protocols for Multi-Agent Systems Interaction. *ACM Transactions on Autonomous and Adaptive Systems, 2*(4). Obtenido de http://www.fipa.org/subgroups/ROFS-SG-docs/2007-TAAS-specifying-MAS.pdf

Pursula, M. (1999). Simulation of Traffic Systems - An Overview. *Journal of Geographic Information and Decision Analysis, 3*(1), 1-8.

Python Software Foundation. (s.f.). *Python*. Obtenido de https://www.python.org/downloads/

Python Software Foundation. (s.f.). *Requests*. Obtenido de https://pypi.org/project/requests/

R Foundation. (s.f.). *R*. Obtenido de https://www.r-project.org/other-docs.html

Rao, A. S., & Georgeff, M. P. (s.f.). BDI Agents: From Theory to Practice. *Proceedings of the First International Conference on Multiagent Systems.* Obtenido de https://www.aaai.org/Papers/ICMAS/1995/ICMAS95-042.pdf

Redacción Noticias Salamanca. (2021). El Ayuntamiento pone en marcha cuatro nuevas bases del servicio de préstamo 'SALenBICI'. *Noticias Salamanca*. Obtenido de https://noticiassalamanca.com/local/el-ayuntamiento-pone-en-marcha-cuatro-nuevas-bases-del-servicio-de-prestamo-salenbici/

RedHat. (s.f.). *What is a REST API?* Obtenido de RedHat: http://www.redhat.com/en/topics/api/what-is-a-rest-api

RStudio. (s.f.). *RStudio*. Obtenido de https://www.rstudio.com/

Sánchez-Corcuera, R., Nuñez-Marcos, A., Sesma-Solance, J., Bilbao-Jayo, A., Mulero, R., Zulaika, U., . . . Almeida, A. (2019). Smart cities survey: Technologies, application domains and challenges for the cities of the future. *International Journal of Distributed Sensor Networks, 15*(6). doi:https://doi.org/10.1177/1550147719853984

Scikit-learn. (s.f.). Obtenido de https://scikit-learn.org/stable/

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press.

Singla, A., Santoni, M., Bartók, G., Meenen, M., & Krause, A. (2015). Incentivizing Users for Balancing Bike Sharing Systems. *Twenty-Ninth AAAI Conference on Artificial Intelligence*, (págs. 723-729). Obtenido de https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9942/9319

Sublime. (s.f.). Obtenido de https://www.sublimetext.com/

SUMO issues: allow adding connector edges to avoid dead-ends. (2021). Obtenido de GitHub: https://github.com/eclipse/sumo/issues/8570

SUMO issues: Missing connections between edges in Brunswick scenario. (s.f.). Obtenido de GitHub.

The Apache Software Foundation. (s.f.). Obtenido de https://tomcat.apache.org/

Trindade, E. P., Hinnig, M. P., Costa, E. M., Marques, J. S., Bastos, R. C., & Yigitcanlar, T. (2017). Sustainable development of smart cities: a systematic review of the literature. *Journal of Open Innovation: Technology, Market, and Complexity, 3*(11). doi:10.1186/s40852-017-0063-2

Wang, S., He, T., Zhang, D., Liu, Y., & H. Son, S. (2019). Towards Efficient Sharing: A Usage Balancing Mechanism for Bike Sharing Systems. *The World Wide Web Conference* (págs. 2011–2021). San Francisco, CA, USA: Association for Computing Machinery. doi:10.1145/3308558.3313441

Waskom, M. (s.f.). *Seaborn*. Obtenido de https://seaborn.pydata.org/

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems, 3*, 285-312. doi:https://doi.org/10.1023/A:1010071910869

Yellowbrick. (s.f.). Obtenido de https://www.scikit-yb.org/en/latest/