UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Análisis y predicción de datos obtenidos del funcionamiento de un AGV

Presentado por Gonzalo Burgos de la Hera
en Universidad de Burgos — 29 de junio de 2023
Tutor: Bruno Baruque Zanón y Jesús Enrique Sierra García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Gonzalo Burgos de la Hera, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 29 de junio de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En cualquier entorno industrial, poder predecir el comportamiento de un sistema de manera precisa ofrece una gran ventaja, pues permite ahorrar costes mejorando la productividad.

En este trabajo se propone por tanto un sistema desarrollado como un conjunto de microservicios, capaz de almacenar y predecir datos de un AGV (Autonomous Guided Vehicle), concretamente la velocidad de sus ruedas. Con el fin de conseguir el sistema más óptimo y eficiente posible, se ha realizado un estudio comparativo entre diferentes sistemas gestores de bases de datos y de modelos predictivos.

Descriptores

AGV, series temporales, gestores de bases de datos, modelos predictivos, aprendizaje automático, microservicios, ...

Abstract

In any industrial environment, being able to predict the behavior of a system in an accurate way offers a great advantage, since it allows cost savings by improving productivity.

In this work we propose a system developed as a set of microservices, capable of storing and predicting data from an AGV (Autonomous Guided Vehicle), specifically the speed of its wheels. In order to achieve the most optimal and efficient system possible, a comparative study has been carried out between different database and predictive model management systems.

Keywords

AGV, time series, database management systems, predictive models, machine learning, microservices, ...

Índice general

Índice general iii

Índice de figuras v

Índice de tablas vi

Introducción 1

Objetivos del proyecto 5

Conceptos teóricos 7
3.1. Series temporales 7
3.2. Predicción de series temporales 9

Técnicas y herramientas 13
4.1. Metodologías 13
4.2. Herramientas de desarrollo 14
4.3. Entorno de desarrollo 15
4.4. Otras herramientas 16

Aspectos relevantes del desarrollo del proyecto 17
5.1. Primera fase 17
5.2. Segunda fase 21
5.3. Puesta en marcha 26
5.4. Depurado 26

Trabajos relacionados 27

Conclusiones y Líneas de trabajo futuras **29**
7.1. Conclusiones 29
7.2. Líneas de trabajo futuras 30

Bibliografía **33**

Índice de figuras

3.1. Ejemplo de serie temporal 8
5.1. Diseño de la arquitectura 18
5.2. Diagramas de flujo del simulador 20
5.3. Diagrama de flujo del servicio “Reciever” 21
5.4. Arquitectura final 22
5.5. Datos del encoder derecho 23
5.6. Diagrama del servicio de predicción. 25
5.7. Plantilla con los gráficos con las predicciones 26
7.1. Diseño futuro de la arquitectura 31

Índice de tablas

Introducción

Los AGV, Autonomous Guided Vehicles por sus siglas en inglés, son complejos sistemas robóticos, capaces de moverse en un entorno concreto, cuyo uso es transportar cargas pesadas en fábricas o almacenes, y que están diseñados para mejorar la eficiencia y la productividad en la logística y el transporte de materiales. Debido a sus ventajas en seguridad, flexibilidad y velocidad, esta tecnología se está convirtiendo cada vez más importante [1].

Aunque estos sistemas pueden mejorar la productividad, desajustes en su configuración u otros errores operacionales pueden producir una reducción de su rendimiento, y, en casos extremos, causar una detención de la línea de producción. Por este motivo, es necesario extraer información de los sistemas en marcha para analizar el rendimiento de las máquinas y las aplicaciones logísticas. Esta información puede usarse para predecir comportamientos futuros del sistema, realizar mantenimiento predictivo y proveer retroalimentación con el fin de diseñar mejoras continuas de las máquinas. Estas predicciones pueden ser conseguidas con el uso de algoritmos de análisis de series temporales, que permitan anticipar futuras condiciones del sistema. [2]

Algunos de estos datos obtenidos del sistema tienen una baja frecuencia de actualización, como puede ser la temperatura y voltaje de la batería, pero otros cambian cada pocos milisegundos, como la corriente eléctrica, la velocidad, la posición del vehículo, errores y estado, etc. Toda esta información proveída por el AGV debe estar relacionada con el tiempo en el que fue generada, por lo que puede ser agrupada en series temporales [3]. Cualquier tipo de base de datos puede usarse para almacenar esta información generada por los AGV, sin embargo, ya que se trata de series temporales, es preferible utilizar bases de datos para series temporales para optimizar el rendimiento del sistema.

Estos datos pueden ser posteriormente analizados y utilizados para entrenar un modelo basado en técnicas estadísticas o en redes neuronales, que sirva para predecir los indicadores de rendimiento de estos vehículos. Con estas predicciones, se pueden detectar errores con mucha mayor antelación, lo que puede permitir tomar mediadas que no sería factible tomar de otra manera. De esta forma, se pueden conseguir reducir el número de errores críticos que supongan una parada del sistema, reduciendo considerablemente los costes y aumentando la productividad.

Este proyecto está enfocado a utilizarse en un entorno industrial, por lo que su principal objetivo es claro: reducir costes a partir de una mejora de la eficiencia del sistema. Por ello, escoger la base de datos más óptima para esta tarea, así como el mejor modelo para realizar las predicciones son tareas esenciales. Es por esto que este trabajo está muy centrado precisamente en dichas tareas de investigación.

Estrucutra de la memoria

Esta memoria segurá la siguiente estructura:

- **Introducción:** contiene una breve descripción del trabajo, así como una guía de contenidos del mismo.
- **Obtetivos del proyecto:** detalla los objetivos principales del proyecto.
- **Conceptos teóricos:** explicación de varios conceptos necesarios para una mejor comprensión del proyecto.
- **Técnicas y herramientas:** metodologías y técnicas utilizadas durante todo el desarrollo del trabajo.
- **Aspectos relevantes del desarrollo:** descripción del proceso de desarrollo, con los aspectos más relevantes del mismo.
- **Trabajos relacionados:** exposición y comparativa con otros trabajos relacionados.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas durante el desarrollo, así como ideas a futuro para la mejora del mismo.

Junto a la memoria, se incluye a demás unos anexos que incluyen:

- **Plan de proyecto:** contiene la planificación temporal y un estudio de viabilidad legal y económica.
- **Requisitos:** describe los requisitos funcionales y no funcionales del sistema, así como una comparativa de gestores de bases de datos y modelos de predicción, basándose en dichos requisitos.
- **Diseño:** se define el diseño de los datos, procedimental y arquitectónico del sistema.
- **Manual del programador:** recoge los aspectos más importantes para futuros programadores del suistema.
- **Manual de usuario:** detalla las funcionalidades del proyecto para futuros usuarios del mismo.

Materiales adjuntos

Junto con esta memoria y anexos, se incluye:

- Prototipos. Recoge el conjunto de prototipos realizados durante el desarrollo.
- Análisis de rendimiento. Contiene los programas utilizados para hacer las pruebas de rendimiento de las bases de datos y de los modelos de predicción.
- Código del sistema desarrollado.
- Dataset de prueba. Se incluye, incluido en los directorios de los servicios, un dataset de un AGV real con el que realizar pruebas.

Objetivos del proyecto

El objetivo principal de este trabajo es el de desarrollar un sistema dedicado a un entorno profesional, capaz de mejorar la eficiencia y reducir costes en procesos industriales.

Este proyecto ha sido diseñado como un microservicio [4]. Las aplicaciones diseñadas de esta manera se dividen en elementos más pequeños e independientes entre sí, y que deben comunicarse y coordinarse. De esta forma, se consigue que la aplicación sea modular, más fácil de escalar, de mantener y de desarrollar.

Al dividir la aplicación en lo que a efectos prácticos son aplicaciones más simples, se consigue un código mucho más sencillo, por lo que el diseño del propio código se simplifica. En nuestro caso, se ha utilizado un enfoque de programación funcional, en vez de programación orientada a objetos, pues la simplicidad del propio código hace que esta última metodología no sea la más apropiada.

Los objetivos pueden quedar resumidos en la siguiente lista:

- Desarrollar la aplicación como un microservicio.
- Utilizar Docker para agrupar cada servicio en su contenedor independiente.
- Utilizar un enfoque de programación funcional.
- Realizar un estudio comparativo de sistemas gestores de bases de datos.
- Realizar otro estudio comparando modelos de predicción de series temporales.

- Hacer uso de Git como herramienta de control de versiones.
- Utilizar herramientas de integración continua como CodeClimate.
- Aplicar la metodología ágil durante el desarrollo del software.

Conceptos teóricos

Como se ha mencionado en apartados anteriores, los datos recibidos por el AGV se agrupan en series temporales. Conviene por tanto explicar que son las series temporales, así como los modelos que se utilizarán para intentar predecirlas.

3.1. Series temporales

Una serie temporal es una colección de observaciones obtenidas mediante mediciones repetidas a lo largo del tiempo [5].

Normalmente, las series temporales presentan patrones que pueden utilizarse para realizar predicciones. Estos patrones son:

- **Tendencia**. La tendencia existe cuando hay un incremento o decremento del valor medido a largo plazo. Esta tendencia no tiene por qué ser lineal.
- **Estacionalidad**. El patrón de estacionalidad se presenta cuando una serie temporal se ve afectada por factores estacionales, como puede ser el día de la semana. Siempre tiene una frecuencia fija y conocida.
- **Ciclos**. Un ciclo ocurre cuando los datos muestran incrementos o decrementos a una frecuencia no fija.

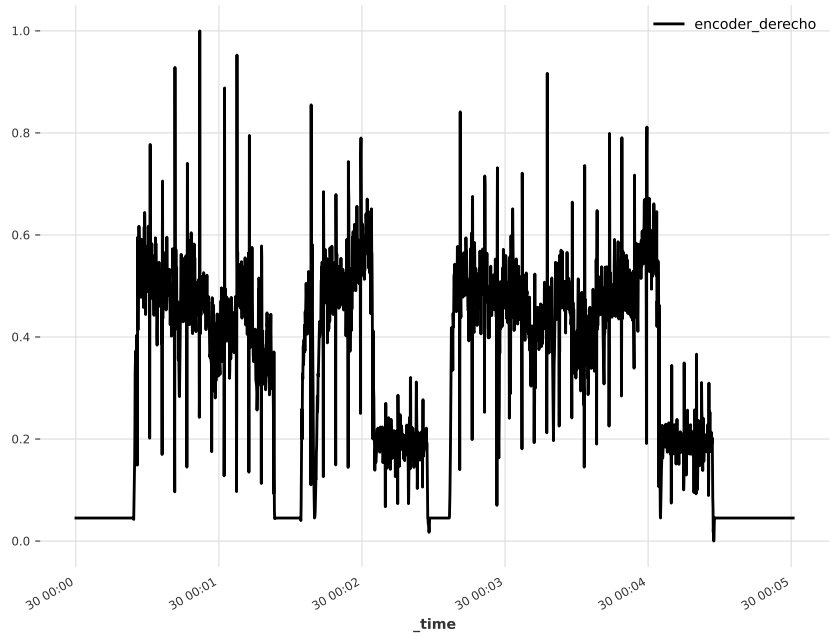


Figura 3.1: Ejemplo de serie temporal

Existen distintos tipos de clasificaciones para las series temporales, según varios puntos de vista [6], de las cuales destacan:

- **Continuas o discretas**. Las series temporales continuas son aquellas en las que la información se obtiene, valga la redundancia, de forma continua, normalmente por un dispositivo analógico, como podría ser los datos recibidos de un sismógrafo. Por otro lado, las series temporales discretas son aquellas en las que la información se obtiene en intervalos de tiempo concretos. Estos intervalos pueden ser equidistantes, o bien ser irregulares. Normalmente, las series temporales medidas por medios digitales son discretas. En nuestro caso, las series temporales enviadas por el AGV son discretas espaciadas en intervalos irregulares, pues el AGV manda dicha información cada varios milisegundos, de manera irregular.

- **Univariantes o multivariantes**. Aquellas series temporales que tengan solo una observación por cada momento del tiempo son series univariantes. Por contra, aquellas en las que se obtengan de manera simultánea mediciones de dos o más fenómenos son multivariantes. Esto

será importante a la hora de escoger que modelo utilizar para realizar predicciones, pues hay modelos que solo soportan series temporales univariantes.

- **Estacionarias o no estacionarias**. Una serie estacionaria [7] es aquella en la que sus propiedades no dependen del momento en el que se observan. Por ello, aquellas que presenten tendencias o estacionalidad no son estacionarias. Por otra parte, una serie de ruido blanco es estacionaria: no importa cuándo se observe, debería tener el mismo aspecto en cualquier momento. De manera general, las series estacionarias no tendrán patrones predecibles en el largo plazo, por lo que conviene convertir las series estacionarias en no estacionarias.

3.2. Predicción de series temporales

La predicción de series temporales es una actividad muy importante en muchos sectores: predicción de datos financieros, predicciones del clima, etc. Debido a esto, existe una gran cantidad de modelos para realizar dichas predicciones. Hay que tener en cuenta sin embargo que predecir datos futuros es una tarea especialmente complicada, y no siempre se obtiene una gran precisión.

Los siguientes modelos se tendrán en cuenta en este trabajo:

- **ARIMA.** Este modelo (Autoregressive Integrated Average) [7] es un enfoque estadístico utilizado para el análisis y pronostico de series temporales. Es una combinación de tres componentes principales:

  - Componente autorregresivo (AR): este componente utiliza la información de valores pasados de la serie temporal. Se basa en la idea de que los valores pasados tienen influencia en el futuro. Este modelo indica cuantos valores pasados se utilizan en la predicción.
  - Componente de media móvil (MA): tiene en cuenta el error residual de las predicciones anteriores para mejorar la precisión, haciendo una media de los errores pasados para predecir los futuros. Indica cuantos errores pasados se tienen en cuenta.
  - Componente de Integración (I): se refiere al proceso de diferenciación de la serie temporal para hacerla estacionaria. El orden de Integración indica cuántas veces se diferencia la serie temporal.

La combinación de estos tres componentes forman el modelo ARIMA(p, d, q), donde "p" representa el orden del componente autorregresivo, "d" es el orden del componente de integración y "q" es el orden del componente de media móvil.

- **TCN.** Una Red neuronal Convolucional Temporal (Convolutional Temporal Network en inglés) [8] es un tipo de red neuronal utilizada para analizar series temporales. Las TCN tienen en cuenta la estructura temporal de los datos y aplican operaciones convolucionales para capturar patrones. Una convolución [9] es un operador matemático que transforma dos funciones en una nueva. Un ejemplo de convolución es la media móvil, o un filtro de aumento de nitidez para imágenes.

  TCN utiliza capas convolucionales de una dimensión para aprender características de la serie temporal. Estas capas son aplicadas sobre ventanas deslizantes de la secuencia para extraer características en diferentes puntos de tiempo.

- **N-HiTS.** Este modelo (Neural Hierarchical interpolation for Time Series) es una extensión del modelo N-BEATS, mejorando su rendimiento y velocidad de entrenamiento [10]. N-BEATS [11] está formado por dos componentes: stack y bloque. Un bloque está formado por una red multicapa que predice valores futuros y pasados. Estos bloques se organizan en pilas (stacks), que agregan las predicciones y errores residuales.

- **Transformer Model.** El Modelo Transformador [12] es una red neuronal que aprende el contexto de la información. Este modelo usa una arquitectura codificador-decodificador, en la que el codificador procesa la entrada de forma iterativa, y el decodificador hace lo mismo con la salida del codificador.

Estos modelos serán los utilizados para la comparación en el desarrollo del proyecto. Para ello, se utilizan las siguientes métricas:

- **MAE**. Siglas de Mean Absolute Error, o Error Absoluto Medio en español. Mide el error medio de una predicción sin tener en cuenta la dirección de dicho error.

- **MASE**. El MASE [13] (Mean Absolte Scaled Error), o error absoluto escalado medio, mide como de bueno es el modelo comparado con un modelo "ingenuo" (modelo que predice un valor como su valor previo).

Un valor por encima de 1 significa que nuestro modelo es peor que dicho modelo ingenuo.

- **DTW**. Siglas de Dynamic Time Warping, es utilizado para medir la similitud entre dos series temporales. Por ejemplo, una predicción con forma de ecuación lineal puede tener el mismo MAE que otra predicción más irregular, pero esta segunda puede parecerse más a los datos reales.

Técnicas y herramientas

4.1. Metodologías

Las siguientes metodologías han sido utilizadas:

- **SCRUM**: es un marco de trabajo cuyas características principales son: desarrollo incremental en lugar de una ejecución completa, basar la calidad en el conocimiento de los integrantes del equipo en vez de en la calidad de los procesos y solapar las fases del proyecto.

  En SCRUM, el trabajo se divide en sprints, periodos de tiempo de entre 1 y 4 semanas. Al principio de cada sprint se planifica el trabajo a desarrollar, y se mantienen reuniones diarias que sirven para mantener la comunicación entre desarrolladores. Al final del sprint se tiene una última reunión en la que se analiza el trabajo realizado.

  En nuestro caso, los sprints han sido de entre 1 y 2 semanas, y no se han tenido las reuniones diarias.

- **Pomodoro** es un método diseñado para mejorar la productividad mediante una correcta gestión del tiempo. Se establece un tiempo de trabajo con una duración de entre 20 y 30 minutos, y un tiempo de descanso de 5 minutos. Después de cuatro ciclos de trabajo/descanso, se realiza un descanso más largo, de unos 15 a 30 minutos.

  Este método ha sido utilizado especialmente a la hora de escribir esta memoria y los anexos.

4.2. Herramientas de desarrollo

Desarrollo

El proyecto ha sido desarrollado íntegramente en el lenguaje de programación Python en su versión 3.10.6.

Además, todos los módulos que más tarde se explican han sido desarrollados como contenedores de Docker. Para ello se han utilizado:

- Docker, en su versión 24.0.2. Docker permite el despliegue de aplicaciones dentro de contenedores, similares a máquinas virtuales, de forma que cada contenedor está aislado del resto del sistema. Docker permite también la ejecución de dichas aplicaciones de forma independiente al sistema operativo, por lo que permite una gran compatibilidad.

- Docker-compose versión 2.16.0. Esta herramienta sirve para definir y ejecutar aplicaciones multi-contenedor.

Sistema Gestor de Base de Datos

El sistema gestor de bases de datos escogido ha sido InfluxDB. Otros gestores planteados para su uso han sido TimescaleDB, Prometheus, Graphite y kdb+.

InfluxDB es un sistema gestor de bases de datos de series temporales. En InfluxDB, los datos se guardan en series. Una serie es un conjunto de puntos que comparten:

- Medida (Measurement): equivalente en SQL a una tabla.

- Conjunto de etiquetas (tag set): equivalente en SQL a una columna indexada. Solo pueden ser cadenas de texto. Sirven para almacenar metadatos.

- Conjunto de valores (field set): equivalente en SQL a una columna sin indexar. Guardan los datos.

Por último, estas series se guardan en “Buckets”, que son el equivalente en SQL a una base de datos.

Predicción

Como librería utilizada para la predicción de las series temporales se ha utilizado Darts [14]. Se consideraron otras alternativas, como Loud ML, herramienta desarrollada específicamente para InfluxDB, y Facebook Prophet. La primera fue descartada debido a que el proyecto está abandonado, y la segunda fue descartada debido a que Darts incluye muchos más modelos, incluido el mismo Prophet.

Se ha utilizado también la libreria Optuna para la optimización de los hiperparámetros de los modelos de predicción.

4.3. Entorno de desarrollo

Todo el trabajo ha sido realizado con el editor de texto de Microsoft Visual Studio Code. Esto ha sido así por varios motivos, principalmente mi comodidad con él, pues es el editor de texto que estoy acostumbrado a usar, y la gran cantidad de extensiones existentes que permitan una mejor experiencia de usuario.

En cuanto a las extensiones utilizadas, caben destacar:

- Docker: utilizada para un manejo más simple de los contenedores de Docker.
- Python: extensión esencial para el desarrollo en Python, pues ofrece características como un Debugger, resaltado de la sintaxis, IntelliSense (autocompletado), etc.
- Todo Tree: utilizada para marcar elementos no terminados, de forma que luego sean fáciles de encontrar.
- LaTeX Workshop: permite compilar de forma automática al guardar los archivos de LaTeX, autocompletado, etc.
- LTeX - LanguageTool grammar/spell checkingLTex: como su nombre en inglés indica, esta extensión analiza errores gramaticales en los archivos de LaTeX.

4.4. Otras herramientas

Control de versiones

El proyecto se aloja en la plataforma GitHub, y la herramienta utilizada para el control de versiones es git. Inicialmente se empezó a usar ZenHub para llevar el control de los sprints. Sin embargo, debido a un problema de licencias a mitad del desarrollo se dejó de utilizar en favor de simplemente usar el apartado de Issues del repositorio para dicha labor.

Documentos

La memoria y los anexos han sido desarrollados en LaTeX, utilizando la distribución TeX Live[15] para la compilación de dichos documentos.

Análisis de código

Se han utilizado dos herramientas para el análisis estático del código:

- Codeclimate: Analiza la mantenibilidad del código, cobertura de pruebas, etc.
- Pylance: Analiza y puntúa el estilo del código.

Estas herramientas se integran de forma que se ejecutan cada vez que se hace un nuevo commit en el repositorio del proyecto. Se pueden ver también en el README los resultados de dichas pruebas.

Aspectos relevantes del desarrollo del proyecto

El desarrollo de este proyecto puede decirse que ha estado dividido principalmente en dos partes. La primera fase estuvo relacionada con el diseño de la arquitectura, así como la elección del sistema gestor de bases de datos. En la segunda fase, se diseñó todo lo relativo a la predicción de datos y a la elección de las herramientas necesarias para dicha tarea.

5.1. Primera fase

Como ha sido mencionado en el párrafo anterior, en la primera fase se llevó a cabo el diseño de la arquitectura del sistema, así como el desarrollo de los servicios que forman dicho sistema. Además, se ha realizado una comparativa de diferentes sistemas gestores de bases de datos con el fin de elegir el que mejor se adapte a nuestros requisitos.

Diseño de la arquitectura

La arquitectura del sistema estudiado se representa en la figura 5.1. El sistema está formado por los siguientes sistemas software:

AGV Coordinator Es un servicio encargado de recibir la información enviada por los AGV a través de una conexión 5G/Wifi. Esta información está codificada como una cadena de bytes, que es decodificada y transformada a formato JSON. Estos mensajes son enviados al servicio "Receiver". Este servicio no ha sido desarrollado como parte de este trabajo.

Simulator Es, como su nombre en inglés indica, un servicio encargado de simular un AGV en caso de no disponer de AGV reales y sea necesario realizar pruebas.

Receiver Este servicio recibe mensajes a través del protocolo UDP bien del "AGV Coordinator" o bien del simulador, y se encarga de insertar dichos datos en la base de datos.

Database Como su nombre indica, este servicio será la base de datos encargada de almacenar todos los datos recibidos.

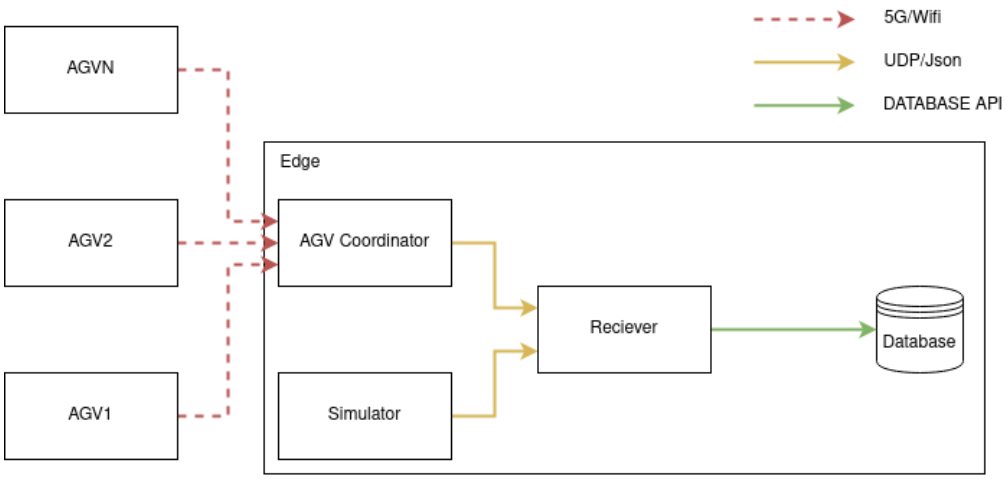


Figura 5.1: Diseño de la arquitectura

Comparativa de Sistemas Gestores de Bases de Datos

Inicialmente, se seleccionaron los cinco sistemas gestores de bases de datos para series temporales más populares según el ranking DB-Engines [16]:

1. InfluxDB
2. Prometheus
3. Kdb+
4. Graphite
5. TimescaleDB

A partir de esta lista de gestores, se ha realizado una comparación exhaustiva de los mismos según los requisitos del proyecto. Esta comparativa puede verse en el apartado B.5 de los anexos que se incluyen de forma complementaria a esta memoria.

Como resumen de dicha comparativa, tanto InfluxDB como TimescaleDB cumplen con los requisitos especificados. Finalmente, el sistema escogido ha sido InfluxDB debido a su menor uso de recursos con un rendimiento suficiente y a la implementación por defecto de una herramienta de visualización de la información.

Desarrollo de los servicios

Una vez diseñada la arquitectura y el sistema gestor de bases de datos escogido, se procedió a desarrollar los sistemas definidos. Todos estos sistemas se ejecutan cada uno en su contenedor de Docker independiente, los cuales se comunican entre sí a través de una red especificada en el archivo de configuración de docker-compose.

Simulator

Inicialmente, no disponía de datos reales del AGV, por lo que la primera versión (Figura 5.2a) simplemente generaba datos aleatorios con campos aleatorios, y los enviaba al nodo "Receiver" por UDP utilizando el puerto 5004.

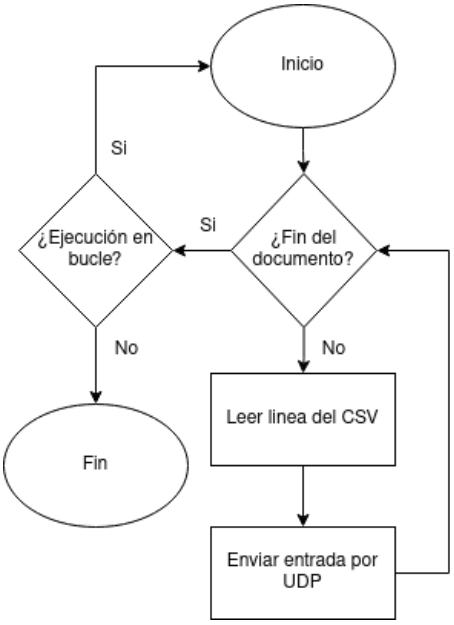
Después, se intentó desarrollar un simulador capaz de, valga la redundancia, simular el comportamiento de un AGV. Sin embargo, una vez dispuse de datos reales del AGV, esta idea se descartó, pues simular dicha información de forma precisa iba a ser demasiado complejo, y se escapa del objetivo de este proyecto. Por tanto, se decidió simular el comportamiento del AGV leyendo los datos de un CSV (Figura 5.2b) obtenido a partir de uno real.

Por último, en la versión final se unificaron los dos procedimientos, de forma que el comportamiento del simulador se decide según lo especificado en un archivo de configuración.

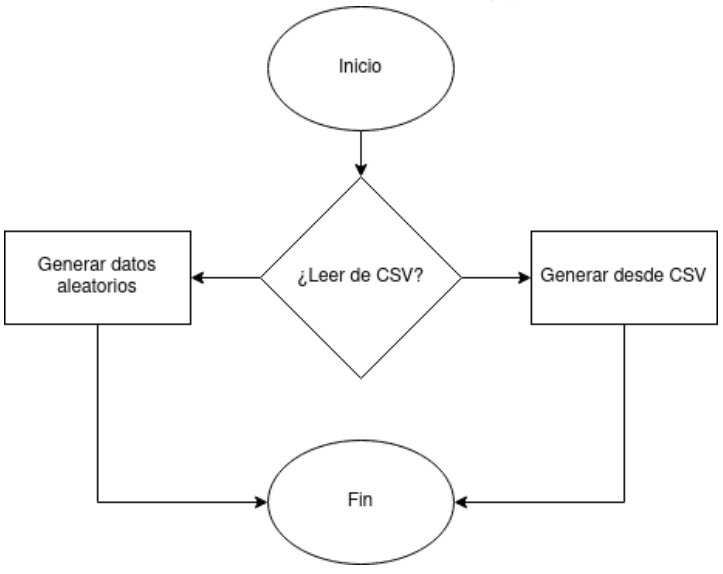
Receiver

El funcionamiento del nodo "Reciever" es muy simple (Figura 5.3): escucha el puerto UDP 5004, y cuando recibe información, la inserta en la base de datos. Inicialmente, el servicio intentará conectarse a la base de

(a) Generación aleatoria



(b) Generación por CSV



(c) Versión final

Figura 5.2: Diagramas de flujo del simulador

datos. Si esta conexión falla un número determinado de veces, el servicio fallará informando de que la conexión no ha podido realizarse.

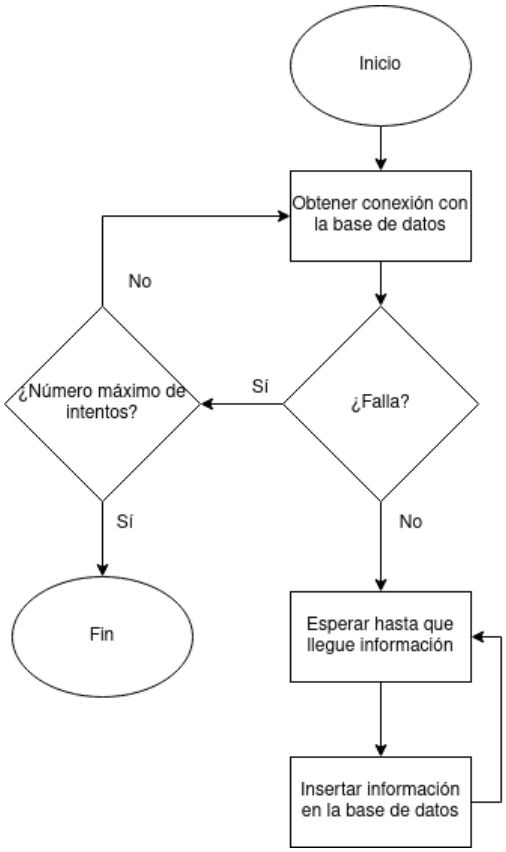


Figura 5.3: Diagrama de flujo del servicio “Reciever”

Database

Este servicio simplemente inicia un contenedor de Docker con la base de datos. Es necesario sin embargo especificar las credenciales del usuario administrador que tendrá los permisos necesarios para insertar información en dicha base de datos.

5.2. Segunda fase

La segunda fase se centró en el desarrollo y diseño del servicio de predicción de nuevos datos. Se ha realizado un análisis comparativo de diferentes modelos predictivos, así como un análisis de los datos recibidos por el AGV para hacer la mejor elección.

Modificación de la arquitectura

Ya que se ha pretendido desarrollar una arquitectura lo más modular posible, la predicción de la información se realiza desde un nuevo servicio. Por ende, la arquitectura modificada queda de la siguiente manera:

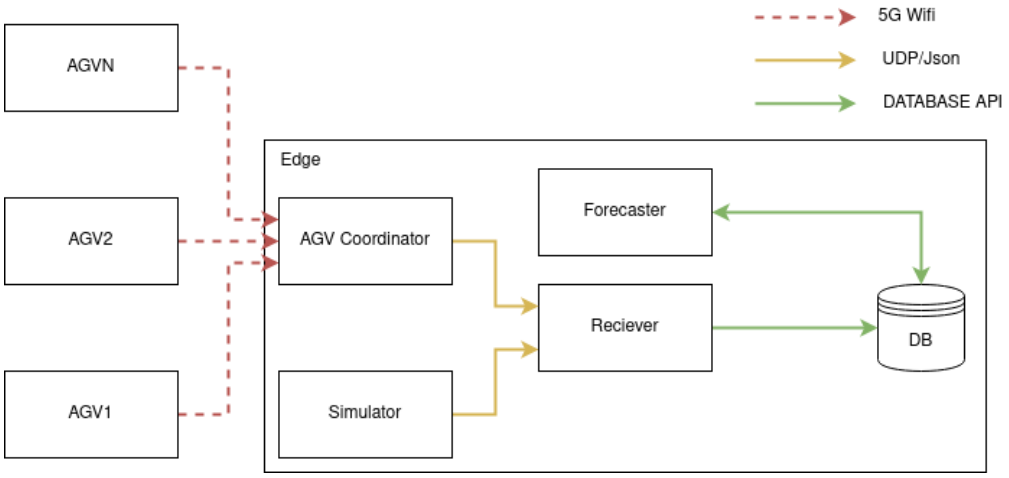


Figura 5.4: Arquitectura final

Este nuevo servicio se conecta directamente a la base de datos de la que extrae la información necesaria para hacer las predicciones. Una vez hechas se insertan en la base de datos para poder ser visualizadas en Chronograf.

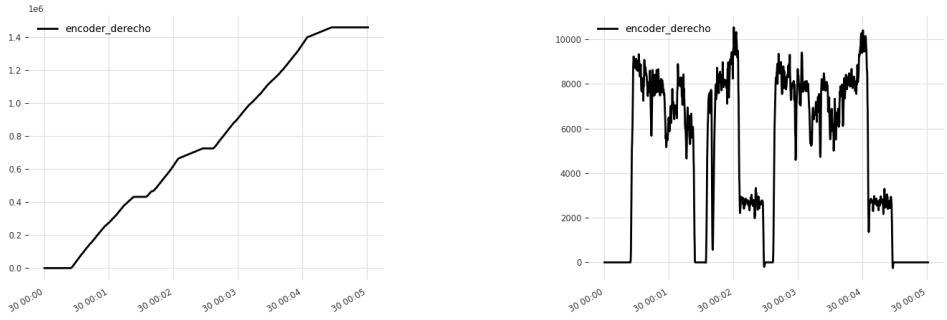
Análisis de los datos

Antes de hacer nada relativo a la predicción de los datos, necesitamos conocer que datos tenemos y cuáles queremos predecir. Del AGV recibimos los siguientes datos:

- Tiempo: tiempo que ha pasado en segundos desde que se inicia el AGV.
- Encoder derecho: valor del codificador de la rueda derecha. Cuando el AGV avanza este valor se incrementa, y cuando retrocede se decrementa.
- Encoder izquierdo: igual al anterior, pero con la rueda izquierda.
- CorrienteL:
- CorrienteH:

- Medida batería:
- Error de guiado: distancia que el AGV está desviado de la banda magnética por la que se guía.
- Consigna de velocidad derecha: valor de la velocidad enviado a la rueda derecha.
- Consigna de velocidad izquierda: similar al anterior pero con la rueda izquierda.
- Display:

Cabe mencionar que, como los encoders derecho e izquierdo no nos muestran directamente la velocidad de las ruedas (Figura 5.5a), antes hay que preprocesar dichos datos, restando cada valor con su anterior, obteniendo así el dato de la velocidad (Figura 5.5b).



(a) Sin procesar (b) Después de diferenciación

Figura 5.5: Datos del encoder derecho

El objetivo es realizar una predicción de los valores encoder derecho y encoder izquierdo para poder compararlos con los obtenidos y detectar posibles errores. Para ello, utilizamos valores anteriores de los mismos datos, así como los valores de las consignas de velocidad derecha e izquierda, pues su correlación es muy alta. Para predecir los valores derechos, y viceversa, se usan también los valores izquierdos, pues aunque su correlación no sea tan alta, también están correlacionados. Ya que estos datos se mandan en intervalos de tiempo irregulares, el primer paso es agruparlos en ventanas de 200 milisegundos. Esto es así porque para realizar las predicciones es necesario que los datos de la serie temporal estén distribuidos de manera uniforme.

Las pruebas realizadas a los modelos han sido realizadas únicamente con el encoder derecho, pues al ser muy similar al encoder izquierdo se van a obtener resultados prácticamente idénticos en los dos casos, por lo que no merece la pena realizar las pruebas sobre los dos datos.

Comparativa de modelos de predicción

Al igual que con la comparativa de sistemas gestores de bases de datos, una comparativa exhaustiva de los modelos de predicción puede encontrarse en la sección B.6 de los anexos complementarios.

De dicha comparación se sacan las siguientes conclusiones:

- El tiempo empleado para la optimización de los modelos no ha sido suficiente, pues los modelos supuestamente optimizados dan peores resultados, tanto en las métricas como en el tiempo de entrenamiento.
- El mejor modelo para predicciones a largo plazo es el Transformer por defecto, aunque N-HiTS da buenos resultados también con un tiempo de entrenamiento menor.
- Para predicciones a corto plazo, ARIMA resulta el mejor modelo.

Ya que nuestra intención es realizar predicciones a relativamente largo plazo, el modelo escogido ha sido el modelo Transformer. Podría haberse escogido también el modelo N-HiTS, pero como el modelo solo se entrena una vez el tiempo de entrenamiento tendrá una menor importancia comparada con el resto de métricas.

Desarrollo del servicio

Como se puede ver en la siguiente imagen (Figura 5.6), este servicio carga un modelo desde un archivo o bien lo entrena desde cero. Para poder entrenar dicho modelo, se necesita que la base de datos ya tenga datos cargados, por lo que la rutina de entrenamiento espera algunos minutos antes de empezar a entrenar para que haya los datos necesarios para poder realizar dicha tarea. La cantidad de tiempo a esperar antes de entrenar se especifica en un archivo de configuración.

En el caso de cargar el modelo desde un archivo, también es necesario esperar un tiempo a que la base de datos se cargue de información, pues es necesario escalar los datos antes de hacer la predicción. Dicha escala se

ajusta según los datos introducidos, por lo que cuanto más tiempo esperamos en este paso, más parecida será a la especificada a la hora de entrenar el modelo.

Los datos generados en la predicción son insertados a un "bucket" de la base de datos diferente al que están los datos. Este "bucket" tiene un periodo de retención bajo. Esto significa que los datos insertados se eliminan después de cierto tiempo, pues no nos interesa guardar predicciones antiguas.

Existen dos versiones de este servicio: una utiliza una GPU de Nvidia para entrenar y la otra utiliza la CPU. Entrenar con GPU es mucho más rápido que con CPU, por lo que siempre que se disponga de una es muy recomendable utilizar esta versión. En caso de no disponer de una GPU, o de disponer de una que no soporte la versión 11.8 de CUDA, siempre puede utilizarse el servicio complementario.

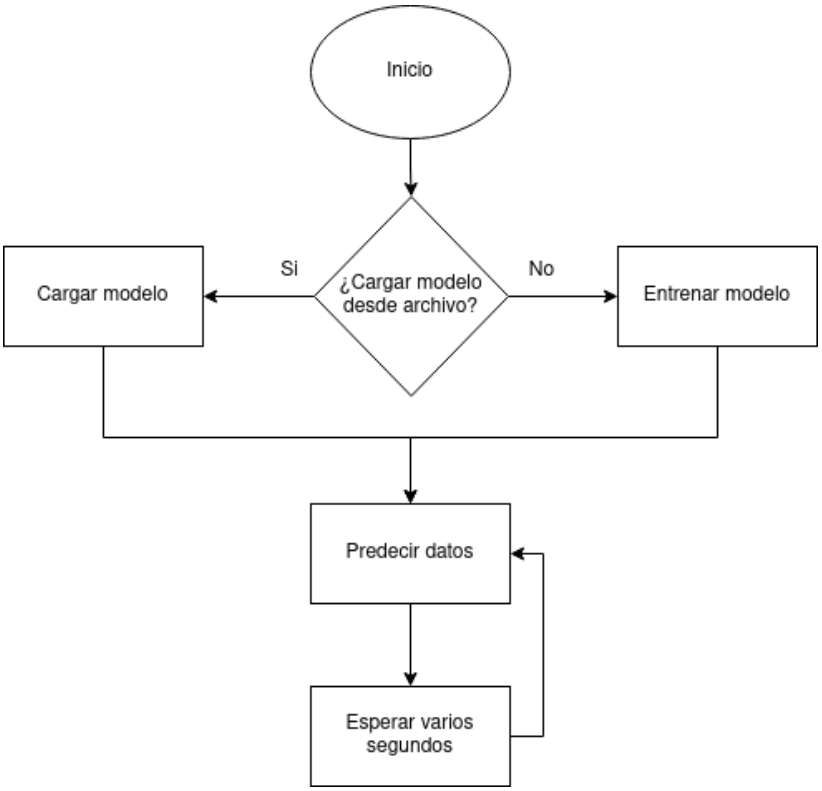


Figura 5.6: Diagrama del servicio de predicción.

5.3. Puesta en marcha

Cada servicio queda definido en un archivo "Dockerfile", que definirá el contenido y acciones de cada uno de los contenedores.

Todos estos servicios se inician utilizando la herramienta docker compose. Con esta herramienta podremos crear una red virtual a la que se conectan los diferentes servicios para poder comunicarse entre sí, especificar los puertos que usa cada contenedor, variables de entorno, volúmenes, etc. Con esta herramienta podemos también ver los registros de cada contenedor, lo que nos permitirá ver posibles errores y trazas de información.

Figura 5.7: Plantilla con los gráficos con las predicciones

En la Figura 5.7 podemos observar las predicciones realizadas en naranja y los datos recibidos por el AGV en azul. Una explicación en detalle de como acceder y utilizar esta interfaz se da en la sección E.4 de los anexos.

5.4. Depurado

Al final del desarrollo, se integraron en el proyecto las herramientas de integración continua Code Climate y Pylint. Gracias a estas herramientas, se localizaron y resolvieron errores estilísticos y de mantenibilidad del código.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

A continuación, se exponen las conclusiones obtenidas tras la finalización del proyecto, así como mejoras a realizar en el futuro.

7.1. Conclusiones

El objetivo general del proyecto ha sido cumplido de manera satisfactoria: se ha conseguido el desarrollo de un sistema capaz de almacenar los datos recibidos por un AGV o por el simulador, y capaz de realizar predicciones precisas que permitan realizar mantenimiento predictivo.

Se ha desarrollado un sistema modular, basado en el desarrollo de microservicios independientes entre sí. Gracias a esto, se ha conseguido también que el sistema se adapte a las limitaciones de hardware del cliente.

Gracias a la utilización de Python y Docker, se ha conseguido crear un sistema sobre el que es muy fácil de realizar modificaciones a futuro: Python agiliza mucho el desarrollo al ser un lenguaje interpretado, que no necesita de compilaciones que resten tiempo al proceso de despliegue, y Docker permite modificar solo aquellos servicios en los que haya cambios.

Durante el desarrollo del proyecto, se han utilizado metodologías y conocimientos obtenidos durante el grado. Se ha seguido una metodología ágil para el desarrollo, se han aprovechado los conocimientos obtenidos sobre bases de datos para la elección del sistema gestor de bases de datos, se han utilizado los conocimientos obtenidos sobre inteligencia artificial y aprendizaje automático, etc.

El desarrollo del proyecto ha servido también para ampliar mis conocimientos en dichos campos. Me ha servido también para adquirir nuevos conocimientos, como herramientas de Integración continua, nuevos métodos de aprendizaje automático, uso de contenedores de Docker para la creación de microservicios, etc.

Por el trabajo de investigación realizado durante el desarrollo del proyecto, mi capacidad para buscar información de calidad ha mejorado de manera sustancial, así como mi capacidad para leer documentos científicos.

Quiero destacar también la labor realizada por mis tutores, guiándome de manera efectiva durante todo el desarrollo del trabajo. También, por sugerencia suya, una parte de este trabajo ha sido publicado en la conferencia SOCO 2023.

7.2. Líneas de trabajo futuras

Con el fin de mejorar la modularidad del proyecto, se sugieren los siguientes cambios:

- Crear un nuevo servicio que haga de intermediario para otros servicios que quieran interaccionar con la propia base de datos (Figura 7.1). Para ello, se propone el desarrollado de una API Rest, mediante la cual se hagan las peticiones a este servicio para hacer consultas e inserciones. De esta forma, se consigue desacoplar los diferentes servicios de la base de datos, lo que resultaría en un diseño mucho más modular, y que simplificaría mucho el desarrollo de nuevos servicios en el futuro.
- Modificar el servicio "Forecaster", para que pueda detectar comportamientos anómalos de los AGV de manera automática utilizando la detección de anomalías de Darts [17], herramienta utilizada para realizar las predicciones.
- Modificar el servicio "Simulator" de manera que permita simular más de un AGV. De manera similar, se sugiere modificar también el servicio "Forecaster" para que pueda realizar predicciones de varios vehículos.
- Ejecutar la optimización de los modelos de predicción comparados durante más tiempo. Por limitaciones de plazos, no fue posible optimizar dichos modelos durante mucho tiempo, por lo que todavía se pueden obtener mejores modelos potenciales.

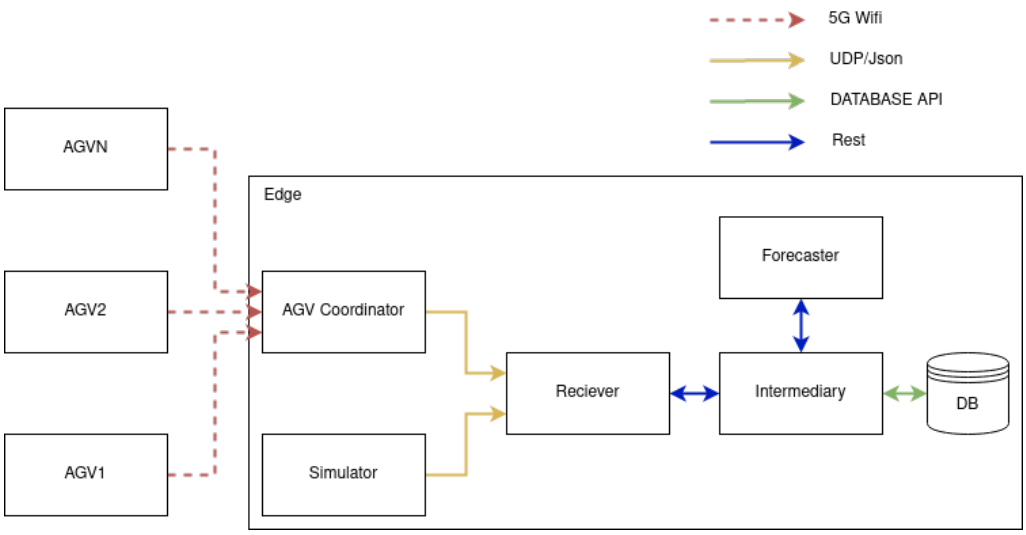


Figura 7.1: Diseño futuro de la arquitectura

Bibliografía

[1] F Espinosa, C Santos y JE Sierra-García. "Transporte multi-AGV de una carga: estado del arte y propuesta centralizada". En: *Revista Iberoamericana de Automática e Informática industrial* 18.1 (2020), págs. 82-91.

[2] Bruno Baruque et al. "Geothermal heat exchanger energy prediction based on time series and monitoring sensors optimization". En: *Energy* 171 (2019), págs. 49-60. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2018.12.207. URL: https://www.sciencedirect.com/science/article/pii/S0360544218325817.

[3] Fatoumata Dama y Christine Sinoquet. "Analysis and modeling to forecast in time series: a systematic review". En: *CoRR* abs/2104.00164 (2021). arXiv: 2104.00164. URL: https://arxiv.org/abs/2104.00164.

[4] Amazon. *¿Qué son los microservicios?* 2023. URL: https://aws.amazon.com/es/microservices/.

[5] InfluxData. *What is time series data?* [Internet; read on 22-march-2023]. URL: https://www.influxdata.com/what-is-time-series-data/.

[6] Genshiro Kitagawa. *Introduction to time series modeling*. CRC press, 2010.

[7] Rob J. Hyndman y George Athanasopoulos. *Forecasting: Principles and Practice*. 2nd. OTexts.com/fpp2. Melbourne, Australia: OTexts, 2018.

[8] Shaojie Bai, J. Zico Kolter y Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". En: *CoRR* abs/1803.01271 (2018). arXiv: 1803.01271. URL: http://arxiv.org/abs/1803.01271.

[9] Mathworks. *Convolución*. 2023. URL: https://la.mathworks.com/discovery/convolution.html.

[10] Cristian Challu et al. "N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting". En: *CoRR* abs/2201.12886 (2022). arXiv: 2201.12886. URL: https://arxiv.org/abs/2201.12886.

[11] Boris N. Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". En: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=r1ecqn4YwB.

[12] Ashish Vaswani et al. "Attention Is All You Need". En: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[13] IBM. *Forecasting statistical details*. 2023. URL: https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=forecasting-statistical-details.

[14] Julien Herzen et al. "Darts: User-Friendly Modern Machine Learning for Time Series". En: *Journal of Machine Learning Research* 23.124 (2022), págs. 1-6. URL: http://jmlr.org/papers/v23/21-1177.html.

[15] TeX user groups. *Tex Live - Tex Users Group*. URL: https://www.tug.org/texlive/.

[16] DB-Engines. *DB-Engines Ranking of Time Series DBMS*. [Internet; read on 22-march-2023]. Mar. de 2023. URL: https://db-engines.com/en/ranking/time+series+dbms.

[17] Unit8co. *Darts - Anomaly Detection*. 2023. URL: https://unit8co.github.io/darts/generated_api/darts.ad.html.