



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Creación de un Servicio de Datos
Sociodemográficos y del Entorno para la
Mejora de la Predicción de Algoritmos
de IA**

Autor: Pablo Guerrero Álvarez

Tutor(a): Raúl Alonso Calvo

Madrid, julio de 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Título del Trabajo, con Mayúscula en Todas las Palabras que no Sean Conectivas (Artículos, Preposiciones, Conjunciones)

Mes Año

Autor: Pablo Guerrero Álvarez

Tutor:

Raúl Alonso Calvo

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

En los últimos años, muchos gobiernos se han sumado al movimiento de datos abiertos. Una iniciativa que busca promover la transparencia, la innovación y el acceso a la información. Por otro lado, la inteligencia artificial ya forma parte de nuestra vida diaria. Se aplica en campos como la economía, la medicina o la investigación científica. Un ejemplo de ello, son las redes de neuronas, que se utilizan para anticiparse a la aparición de enfermedades o detectar fraudes en operaciones bancarias.

A pesar de los esfuerzos de generar datos abiertos, normalmente no se aprovechan. Están dispersos en muchas fuentes, con formatos distintos y estructuras poco homogéneas. Tampoco lo es aplicar modelos de inteligencia artificial, ya que requieren conocimientos técnicos avanzados. Esta combinación de barreras dificulta el uso práctico de ambos conceptos, sobre todo para usuarios individuales o pequeñas empresas.

En este Trabajo de Fin de Grado se presenta una posible solución ante esta realidad. Un sistema que oculta al usuario final toda esa complejidad. Reúne datos meteorológicos y sociodemográficos de distintas fuentes públicas y los pone a disposición de forma unificada, sin necesidad de preocuparse por el formato o el origen. Para probar su utilidad, el sistema se ha usado como entrada en un modelo que predice las ventas de un negocio. El modelo está basado en una arquitectura LSTM (*Long Short-Term Memory*), un tipo de red neuronal. Después, se han comparado los resultados obtenidos con y sin el uso de los datos proporcionados por el sistema, para ver si realmente mejoran la predicción.

Una vez establecida la premisa, este Trabajo lo componen 3 bloques principales:

1. Proyecto Java dedicado a la obtención, homogeneización y almacenamiento de los datos provenientes de datos tanto meteorológicos como sociodemográficos de la AEMET y del INE.
2. Base de datos relacional, que sirve de repositorio de los datos homogeneizados y sirve de nexo entre los otros dos bloques.
3. Servicio REST, dividido a su vez en un servidor construido en torno al framework SpringBoot, siguiendo una arquitectura de 3 capas; y un cliente basado en Python, cuya pieza central es un modelo LSTM.

Para completar el primer bloque, han sido cruciales librerías como Gson, Apache POI y GeoTools para la deserialización, lectura de ficheros Excel y la manipulación de datos geográficos codificados en un archivo “shapefile”; respectivamente.

Por otro lado, el bloque de servicio cuenta, como hemos mencionado, con un modelo LSTM, creado con la librería TensorFlow/Keras. Tras una fase de obtención y preprocesado de los datos de entrada y, habiéndose entrenado y

evaluado cientos de modelos con diferentes combinaciones de hiperparámetros, se pasó a la fase de análisis y visualización de los resultados. De la mano de las librerías gráficas Seaborn y Matplotlib, se identificaron rangos de posibles valores para los hiperparámetros, que optimizaran el desempeño del modelo en la predicción de los datos de venta.

Este proyecto ha cumplido con su objetivo principal: crear un sistema que unifique datos abiertos y facilite su uso en modelos de predicción. Se ha demostrado que integrar variables meteorológicas y temporales puede mejorar la precisión de las predicciones de ventas. El modelo desarrollado ha logrado un error medio del 11,4 %, un resultado razonable teniendo en cuenta la complejidad de los datos, la falta de experiencia previa y la posible ausencia de variables clave. Aunque no se ha podido aprovechar la parte sociodemográfica por la falta de datos actualizados, el sistema ha sido funcional y queda abierto a futuras ampliaciones. Para ello, se han planteado futuras líneas de trabajo como incorporar nuevas fuentes de datos, desarrollar una interfaz web o desplegar el servicio online, ya que al estar basado en tecnologías abiertas, el sistema es escalable y adaptable.

Esta herramienta no solo permite integrar datos abiertos, sino que también muestra cómo pueden aprovecharse en casos reales como la predicción de ventas. Aunque estos datos son públicos, muchas personas no saben que existen o no tienen los medios para usarlos. Aplicaciones como esta pueden ayudar a pequeños negocios a tomar mejores decisiones, ajustando su stock, su personal o sus campañas de marketing en función de predicciones más precisas.

Abstract

In recent years, many governments have joined the open data movement—an initiative that aims to promote transparency, innovation, and access to information. On the other hand, artificial intelligence is already part of our daily lives. It is applied in fields such as economics, medicine, and scientific research. A good example of this is neural networks, which are used to anticipate the onset of diseases or detect fraud in banking operations.

Despite efforts to generate open data, it is often not fully utilized. The data is scattered across many sources, with different formats and non-uniform structures. Applying artificial intelligence models is not straightforward either, as they require advanced technical knowledge. This combination of barriers makes the practical use of both concepts difficult, especially for individual users or small businesses.

This Final Degree Project presents a possible solution to this reality: a system that hides all that complexity from the end user. It gathers meteorological and sociodemographic data from various public sources and makes them available in a unified way, without the need to worry about format or origin. To test its usefulness, the system has been used as input for a model that predicts business sales. The model is based on an LSTM (Long Short-Term Memory) architecture, a type of neural network. Afterwards, the results obtained with and without the use of the system's data were compared to assess whether prediction accuracy actually improves.

Once the premise is established, this project is structured into three main blocks:

1. A Java project dedicated to the collection, homogenization, and storage of data from both meteorological and sociodemographic sources, such as AEMET and INE.
2. A relational database that acts as a repository for the homogenized data and as a connection point between the other two blocks.
3. A REST service, divided into a server built using the SpringBoot framework, following a three-layer architecture; and a client based on Python, with an LSTM model as its core component.

To complete the first block, libraries such as Gson, Apache POI, and GeoTools were essential for deserialization, reading Excel files, and manipulating geographic data encoded in a shapefile, respectively.

On the other hand, the service block includes, as mentioned, an LSTM model created using the TensorFlow/Keras library. After a phase of data retrieval and preprocessing, and having trained and evaluated hundreds of models with different hyperparameter combinations, the process moved on to the analysis and visualization phase. With the help of the Seaborn and Matplotlib graphic

libraries, ranges of possible values for the hyperparameters were identified to optimize the model's performance in sales prediction.

This project has fulfilled its main objective: to create a system that unifies open data and facilitates its use in predictive models. It has been demonstrated that integrating meteorological and temporal variables can improve the accuracy of sales forecasts. The developed model achieved a mean error of 11.4%, a reasonable result considering the complexity of the data, the lack of prior experience, and the possible absence of key variables. Although the sociodemographic component could not be used due to a lack of updated data, the system proved functional and remains open to future expansions. For that purpose, future lines of work have been proposed, such as incorporating new data sources, developing a web interface, or deploying the service online. Since the system is based on open technologies, it is scalable and adaptable.

This tool not only enables the integration of open data but also demonstrates how it can be used in real-world scenarios like sales prediction. Although this data is public, many people are unaware of its existence or lack the means to use it. Applications like this can help small businesses make better decisions by adjusting their stock, staff, or marketing campaigns based on more accurate predictions.

Índice

1	Introducción	1
1.1	Contexto y Motivación	1
1.2	Problema a Resolver	2
1.3	Objetivos del Proyecto.....	2
1.4	Metodología y Enfoque.....	3
1.5	Estructura del Documento.....	4
2	Estado del Arte.....	6
2.1	Fundamentos Teóricos.....	6
2.1.1	Inteligencia Artificial	6
2.1.2	Open Data.....	16
2.2	Tecnologías	19
2.2.1	Generales	19
2.2.2	Tratamiento y Almacenamiento de Datos	20
2.2.3	Servicio web: Servidor	22
2.2.4	Servicio web: Cliente y LSTM	23
2.3	Proyectos Similares	24
3	Desarrollo	28
3.1	Obtención de los datos.....	29
3.1.1	AEMET.....	32
3.1.2	INE	40
3.2	BBDD	45
3.3	Servicio Web – Servidor.....	49
3.3.1	Estructura del proyecto. Organización en paquetes	51
3.3.2	Endpoints desarrollados.....	53
3.3.3	Documentación, dependencias y tratamiento de errores	58
3.3.4	Diagrama de secuencia	60
3.4	Servicio Web – Cliente y LSTM	60
3.4.1	Introducción	60
3.4.2	Preparación de los datos	61
3.4.3	Diseño del modelo LSTM	63
3.4.4	Entrenamiento y experimentación	65
3.4.5	Visualización de los datos	66
4	Evaluación	67
4.1	Objetivo 1 y 2	67

4.2	Objetivo 3.....	70
4.2.1	Coordenadas geográficas	70
4.2.2	Código postal.....	72
4.2.3	Comunidad.....	75
4.3	Objetivo 4.....	77
5	Resultados y conclusiones	94
6	Análisis de Impacto	97
7	Bibliografía.....	98
8	Anexos	103
8.1	Anexo A: Diagramas de Gantt con los objetivos.	103
8.1.1	Primer diagrama de Gantt	103
8.1.2	Segundo diagrama de Gantt	104
8.2	Anexo B: Figuras evaluación	105
8.3	Anexo C: Metadata de la API de la AEMET	110
8.3.1	Primer endpoint : Observación 12 horas	110
8.3.2	Segundo endpoint: Mediciones Históricas	112
8.4	Anexo D: Objeto devuelvo por los endpoints del servicio	113

Figura 2.1	Una visión simplificada de las subdisciplinas de la IA y su relación	7
Figura 2.2	Aprendizaje supervisado vs Aprendizaje no supervisado	8
Figura 2.3	Modelo del perceptrón. Fuente: José Mariano Álvarez	9
Figura 2.4	Representación de una red neuronal recurrente (RNN)	10
Figura 2.5	Representación esquemática de una celda LSTM, mostrando las puertas de olvido, entrada y salida, así como el flujo de información interna.	12
Figura 2.6	Curva de relación entre complejidad y error.....	15
Figura 2.7	Conjuntos de datos por nivel de administración.....	18
Figura 2.8	Porcentaje de distribuciones por formato por nivel de administración.	18
Figura 3.1	Diagrama de arquitectura general del sistema, dividido en 3 bloques: Procesamiento de datos, base de datos y cliente-servidor.....	29
Figura 3.2	Objeto JSON con la respuesta inicial del servidor de AEMET	33
Figura 3.3	Diagrama UML de clases para predicción por municipio	35
Figura 3.4	Diagrama UML del modelo de estaciones antiguas	37
Figura 3.5	Diagrama UML para la descarga del histórico diario.....	39
Figura 3.6	Fragmento de un Excel con datos del INE.....	42
Figura 3.7	Diagrama UML de las clases de obtención y carga de datos del INE	44
Figura 3.8	Diagrama entidad - relación.....	46
Figura 3.9	Pantalla principal Spring Initializr	50

Figura 3.10 Diagrama UML del servicio	53
Figura 3.11 Tabla de códigos AEMET.....	57
Figura 3.12 Captura del Swagger ofrecido por el servidor	59
Figura 3.13 Diagrama de secuencia.....	60
Figura 4.1 Captura del Swagger para coordenadas geográficas	70
Figura 4.2 Falta un parámetro de entrada	71
Figura 4.3 Petición correcta para coordenadas geográficas	71
Figura 4.4 Error en el formato de los parámetros	72
Figura 4.5 Captura Swagger código postal	72
Figura 4.6 Faltan parámetros de entrada código postal	73
Figura 4.7 Formato del código postal incorrecto pt1	73
Figura 4.8 Formato del código postal incorrecto pt2	74
Figura 4.9 Petición correcta para código postal	74
Figura 4.10 Captura Swagger comunidad	75
Figura 4.11 Faltan parámetros de entrada comunidad	75
Figura 4.12 Formato de id_comunidad incorrecto.....	76
Figura 4.13 Petición correcta para comunidad	76
Figura 4.14 No se encuentra el endpoint POST .../comunidad.....	77
Figura 4.15 Comparación entre ventas reales y predichas utilizando una red LSTM compleja de 128 neuronas	78
Figura 4.16 Evolución de la pérdida de entrenamiento y validación durante las 300 épocas.	79
Figura 4.17 Top 5 combinaciones de hiperparámetros (ventana, batch size, dropout, épocas) obtenidas para el modelo LSTM con 64 neuronas.	80
Figura 4.18 Comparación entre ventas reales y predichas utilizando el modelo LSTM S de 64 neuronas.	81
Figura 4.19 Evolución de la pérdida de entrenamiento y validación para el modelo LSTM S de 64 neuronas.	81
Figura 4.20 Top 5 combinaciones de hiperparámetros obtenidas para el modelo LSTM S con 16 neuronas.	82
Figura 4.21 Top 5 combinaciones de hiperparámetros obtenidas para el modelo LSTM S con 64 neuronas.	82
Figura 4.22 Segunda Prueba: Top combinaciones de hiperparámetros obtenidas.	83
Figura 4.23 Gráfica comparación de la predicción vs real del modelo LSTM S 64 optimizado.....	84
Figura 4.24 Evolución de la pérdida de entrenamiento y validación para el modelo LSTM S 64 optimizado modelo.	85
Figura 4.25 Gráfica de correlación	86
Figura 4.26 Gráfica comparación de la predicción vs real del modelo con la configuración de la Tabla 4.7 Configuración de hiperparámetros	88
Figura 4.27 Matriz de correlación promedio entre los hiperparámetros y las métricas de error.	91
Figura 4.28 Comparación final entre las ventas reales y las predicciones generadas por los dos modelos	93

Tabla 2.1 Unidades básicas de JDBC y su función	21
---	----

Tabla 2.2 Anotaciones generales Spring	22
Tabla 2.3 Anotaciones de gestión web y rest Spring.....	22
Tabla 2.4 Anotaciones de acceso a datos Spring	23
Tabla 3.1 Doble petición y deserialización con Gson	40
Tabla 3.2 Código SQL de creación de la tabla Mediciones	47
Tabla 3.3 Código inserción en la BBDD.....	48
Tabla 3.4 Ejemplos de querys SQL para testing.....	49
Tabla 3.5 Tabla con las respuestas HTTP para código postal.....	54
Tabla 3.6 Función que consulta la BBDD usando JPA.....	55
Tabla 3.7 Tabla con las respuestas HTTP para coordenadas geográficas	56
Tabla 3.8 Tabla con las respuestas HTTP para comunidad	58
Tabla 3.9 Anotaciones Swagger	59
Tabla 3.10 Asociación entre los métodos creados y el endpoint al que apuntan	61
Tabla 3.11 Método para obtener los datos del endpoint /codigopostal	61
Tabla 3.12 Tipos de pedidos aceptados	62
Tabla 3.13 Código para filtrar códigos postales dado un id de comunidad	62
Tabla 3.14 Comparación para el método get_dummies	63
Tabla 3.15 Método para crear secuencias usadas por el LSTM	64
Tabla 3.16 Modelo LSTM final	64
Tabla 3.17 Desescalamado y almacenamiento de las ventas predichas y reales .	65
Tabla 4.1 Mediciones	68
Tabla 4.2 Mediciones	68
Tabla 4.3 Datos secciones censales.....	69
Tabla 4.4 Total códigos postales en la base de datos.....	69
Tabla 4.5 Total secciones censales.....	69
Tabla 4.6 Tabla con las combinaciones posibles	83
Tabla 4.7 Configuración de hiperparámetros.....	87

Figura Anexo 1 Diagrama de Gantt del Trabajo de Fin de Grado, dividido por objetivos.....	103
Figura Anexo 2 Segundo diagrama de Gantt del Trabajo de Fin de Grado: planificación de revisiones.....	104
Figura Anexo 3 Evolución de la pérdida de entrenamiento y validación para este modelo.....	105
Figura Anexo 4 Gráfico de barras que muestra la correlación entre los hiperparámetros y la diferencia entre la pérdida de entrenamiento y validación. Ayuda a identificar combinaciones que podrían inducir sobreajuste.....	105
Figura Anexo 5 <i>Gráfico de barras que representa la correlación entre los hiperparámetros y el MAE calculado sobre medias móviles.</i>	106
Figura Anexo 6 Gráfico de barras con la correlación entre los hiperparámetros y la métrica MAE (Error Absoluto Medio).....	106
Figura Anexo 7 Gráfico de barras que muestra la correlación entre los hiperparámetros y la métrica MSE (Error Cuadrático Medio).....	107
Figura Anexo 8 Mapa de calor Épocas - Batch.....	107
Figura Anexo 9 Mapa de calor Batch – Neuronas.....	108
Figura Anexo 10 Mapa de calor Dropout - Épocas	108
Figura Anexo 11 Mapa de calor Épocas - Neuronas	109

Figura Anexo 12 Mapa de calor Ventana – Batch.....	109
Figura Anexo 13 Tabla con muestras de los entrenamientos del LSTM	110
Figura Anexo 14 Metadata primer endpoint AEMET	112
Figura Anexo 15 Metadata segundo endpoint AEMET.....	113
Figura Anexo 16 Formato JSON de ObjetoRespuesta.....	114

1 Introducción

1.1 Contexto y Motivación

El movimiento de datos abiertos ha cobrado impulso en las últimas décadas, impulsado por organizaciones e iniciativas como la *Open Knowledge Foundation*[1], nacida en 2004 y *Open Government Initiative*[2]. Estas iniciativas tienen como objetivo promover la transparencia, el acceso a la información y el uso de datos públicos para el bien común. *Open Knowledge Foundation* ha jugado un papel clave en la creación de estándares y recursos para facilitar el acceso y la reutilización de datos, mientras que la *Open Government Initiative* ha fomentado que gobiernos de todo el mundo publiquen datos de forma accesible para sus ciudadanos, facilitando un entorno donde se promueva la innovación.

Por otro lado, la inteligencia artificial (IA) está ganando una creciente importancia en múltiples campos de trabajo, transformando sectores como el deporte, las finanzas, la industria o la educación[3]. Sus aplicaciones permiten mejorar la toma de decisiones, optimizar procesos y descubrir patrones complejos en grandes volúmenes de datos. Por poner un ejemplo, siete de cada diez CEO de un conjunto de casi mil setecientos directivos de diversos países como Estados Unidos, Reino Unido, España o Francia, afirman que la IA es una “prioridad de inversión para su organización”[4]. Sin ir más lejos, este año, la Inteligencia Artificial ha sido reconocida en el ámbito científico: el Premio Nobel de Física de 2024 fue otorgado por “descubrimientos e invenciones fundamentales que habilitan el aprendizaje automático con redes neuronales artificiales”. Este logro destaca el impacto de la IA en nuestra sociedad y subraya su potencial para seguir innovando en numerosos campos[5][6][7].

Centrándonos en España, según el “IV Plan de Gobierno Abierto”, llevado a cabo por el gobierno, se están implementando medidas para “reforzar la transparencia y la rendición de cuentas”[8]. Sin embargo, pese a la mejora de las capacidades y de la gestión durante los últimos años, el impacto en la población apenas ha comenzado a despegar[9].

De esta información, obtenida del *Global Data Barometer*, podemos inferir que muchas de las aplicaciones prácticas aún no logran aprovechar estos datos de manera que beneficien directamente a los negocios y ciudadanos. Con este proyecto se busca contribuir a cambiar esta realidad mediante la implementación de un servicio web que, al integrar datos sociodemográficos y meteorológicos, proporciona información clave para mejorar la toma de decisiones en el comercio. Al emplear una red de neuronas, el sistema ofrece un enfoque innovador para potenciar las ventas, promoviendo así el valor de los datos abiertos en el ámbito empresarial español.

1.2 Problema a Resolver

Uno de los problemas principales es la dispersión de los datos. Enfocándonos en los que se tratarán en este proyecto, nos encontramos con la realidad de que los datos provienen de distintas fuentes. Por ejemplo, las mediciones históricas de las estaciones meteorológicas junto con las predicciones de la semana para cada municipio se encuentran en la página de AEMET OpenData[10], los datos sociodemográficos en una página del INE[11] y el shapefile con los datos geográficos relativos a los municipios españoles se encuentran en otra página del INE[12]. Además, cada conjunto de datos tiene un formato y una estructura diferentes lo que, en definitiva, hace que no solo sea compleja su extracción si no también su integración.

Pese a que existe una herramienta del gobierno para la búsqueda de datos públicos[13], no es posible encontrar toda la información completa de, por ejemplo, indicadores demográficos por municipio y en caso de encontrar algo que en un principio pueda parecer relevante, la falta de una descripción de los datos hace que en definitiva la población no encuentra la herramienta útil y fácil de utilizar.

Por ello, se propone el desarrollo de un sistema centralizado que integre todos estos datos en una base de datos relacional y proporcione un servicio web que permita su consulta de forma sencilla. Posteriormente, esta información será utilizada como entrada para un modelo de red neuronal LSTM, con el objetivo de predecir las ventas en un comercio, además de determinar si mejora en algo la predicción al usar datos adicionales.

1.3 Objetivos del Proyecto

Este proyecto se estructura en cuatro objetivos principales, cada uno de los cuales incluye una serie de tareas detalladas que se llevarán a cabo de manera secuencial y planificada¹. Los objetivos son los siguientes:

- I. **Obtención de datos meteorológicos y sociodemográficos:** Este objetivo consiste en la obtención de datos de diversas fuentes públicas, específicamente de la AEMET (Agencia Española de Meteorología) y del INE (Instituto Nacional de Estadística). A través de consultas a las APIs disponibles y el procesamiento de archivos Excel, se recopilarán los datos meteorológicos (predicciones y mediciones) a nivel de municipios y estaciones, así como los datos sociodemográficos (renta media y mediana, distribución por fuente de ingresos, indicadores demográficos y distribución geográfica) a nivel nacional y de secciones censales.

¹ Diagrama de Gantt con la distribución de los objetivos y tareas a lo largo del tiempo.
Se puede consultar en el **Anexo 8.1**

- II. **Integración de datos en una base de datos relacional:** Una vez obtenidos los datos, se procederá a su integración en una base de datos relacional. Para ello, se diseñará un modelo de datos adecuado, definiendo las entidades, atributos y relaciones, y se llevará a cabo la creación de la base de datos y el código necesario para la inserción y consulta de los datos.
- III. **Desarrollo de un servicio web RESTful para consultas:** El tercer objetivo se centra en la creación de un servicio web basado en arquitectura RESTful, el cual permitirá la consulta de los datos integrados. Este servicio web permitirá acceder a la información en función de diferentes parámetros de entrada, como coordenadas geográficas, el código postal o el id de la comunidad. El desarrollo del servicio se llevará a cabo principalmente en Java, utilizando el framework Spring, concretamente Spring Boot, lo cual facilitará la creación de endpoints RESTful escalables. El servicio se diseñará siguiendo principios de buenas prácticas REST, proporcionando un Swagger para mayor claridad en la documentación, además de cualquier documento que ayude a estructurar el servicio.
- IV. **Aplicación de inteligencia artificial para la predicción de ventas:** Finalmente, se evaluará el uso de la información recopilada para optimizar las predicciones de ventas en un comercio. Esto se llevará a cabo mediante el desarrollo de un modelo de red neuronal, entrenado con las variables adecuadas. Se evaluarán distintas configuraciones de entrada —como agrupaciones diarias, semanales; por código postal y por comunidad— con el fin de determinar cuál ofrece un mejor ajuste al modelo. Además, se evaluará la eficacia de las predicciones y se ajustarán las variables de entrada si es necesario para mejorar la precisión. Una vez escogido el mejor modelo, se entrenará 2 veces: uno usando variables de entrada además de las ventas para predecir y otro usando únicamente las ventas. Finalmente se realizará una comparación y una conclusión de si ha habido mejora o no.

1.4 Metodología y Enfoque

En cuanto a la metodología de este proyecto, el primer objetivo implica la obtención de datos mediante dos métodos diferentes. Las mediciones y predicciones meteorológicas de la AEMET se consultarán a través de su API REST, mientras que los datos sociodemográficos del INE se descargarán en formato Excel y se procesarán utilizando la biblioteca Apache POI. Una vez obtenidos, los datos se integrarán en una base de datos relacional en MySQL, con un modelo entidad-relación diseñado previamente y cargado mediante código Java.

Para el desarrollo del servicio RESTful se utilizará el framework “Spring” siguiendo buenas prácticas en su diseño. Por último, se desarrollará un cliente en Python que consultará estos datos, funcionando como enlace con el modelo de red de neuronas para obtener valor. Además, se llevará a cabo una

planificación de tareas en una agenda, con revisiones semanales para evaluar las tareas cumplidas y realizar ajustes según sea necesario.

Como parte de la planificación, se han elaborado dos diagramas de Gantt, uno correspondiente a la fase septiembre 2024 – enero 2025 y otro a la fase marzo 2025 – junio 2025. En ambos se detallan las tareas que se han completado para satisfacer los distintos objetivos.

Además, a lo largo del desarrollo, se han realizado diversos diagramas con el fin de facilitar la comprensión del sistema. Algunos de ellos son los siguientes:

- **Diagrama de arquitectura:** Para dar una visión general del proyecto y sus partes principales.
- **Diagrama de clases:** Detallando las implementaciones de ambos proyectos Java. El primero, relacionado con la obtención y almacenamiento de los datos en una BBDD; y el segundo, el servicio REST Spring.
- **Diagrama entidad-relación (E/R):** Para el diseño de la base de datos (BBDD).

1.5 Estructura del Documento

A continuación, se expresa brevemente la estructura seguida para la consecución de este proyecto.

En el segundo capítulo, “Estado del Arte”, se tratarán los fundamentos teóricos que rodean al proyecto como son la inteligencia artificial y dos de sus ramas que son el machine learning y las redes de neuronas. Vendrá seguido de una contextualización sobre las iniciativas de Open Data, los obstáculos que intentan abordar, su situación actual y futura y los principios por los que se rigen. Luego se hablará sobre la tecnología empleada para alcanzar los objetivos del proyecto, el lenguaje de programación utilizado junto con sus librerías, formatos de los datos empleados y las herramientas para la documentación. Para finalizar el segundo capítulo se abordarán los proyectos similares relacionados con este. Contendrá una reflexión sobre la inspiración que haya podido obtener de estos proyectos y qué cosas serían interesantes aplicar.

El tercer capítulo, el más importante, será el de “Desarrollo” donde se detallará por objetivos la realización de este proyecto. En primer lugar, habrá un apartado de inicio del proyecto y las tareas que se han hecho, hasta finalizar con las pruebas del producto final – el modelo –.

El cuarto capítulo será el de “Evaluación” donde se analizarán todas las partes de este proyecto y si cumplen con su papel en el conjunto de este Trabajo de Fin de Grado.

El quinto capítulo constará de “Resultados y Conclusiones” donde se examinará si existe espacio de mejora, qué cosas se podrían haber hecho mejor, otras tecnologías que sería útil agregar al proyecto y una conclusión sobre si el proyecto en general cumple su objetivo marcado al principio de este documento.

Finalmente se incluirá una Bibliografía, además de un Anexo con los que finalizará este documento. En esta parte se incluirán los artículos de investigación, páginas web y demás información consultada para la realización de este Trabajo de Fin de Grado, además de figuras, gráficas e información que por su extensión o por otro motivo no se incluyen en alguna de las partes anteriores.

2 Estado del Arte

2.1 Fundamentos Teóricos

2.1.1 Inteligencia Artificial

2.1.1.1 Introducción

Hay muchas definiciones de la inteligencia artificial (IA) en función de con qué se esté tratando de comparar [14]. Algunas hablan de en términos de “*percepción del entorno y la complejidad del mundo real*”, otras de “*procesamiento de la información*” y otras del “*proceso de decisión*”. Es por ello por lo que en 2020 surge un programa de la *Comisión Europea* para reunir más de cincuenta y cinco definiciones provenientes de diferentes instituciones o de publicaciones relevantes de los últimos cincuenta años con el objetivo de estandarizar una definición que sintetice lo mejor posible su significado. Este programa decide seguir la definición propuesta por el *High-Level Expert Group (HLEG)*, que define así la Inteligencia Artificial:

“Los sistemas de inteligencia artificial (IA) son sistemas de software (y posiblemente también de hardware) diseñados por humanos que, dado un objetivo complejo, actúan en la dimensión física o digital percibiendo su entorno mediante la adquisición de datos, interpretando los datos estructurados o no estructurados recopilados, razonando sobre el conocimiento, o procesando la información, derivada de estos datos y decidiendo la(s) mejor(es) acción(es) a tomar para alcanzar el objetivo dado. Además, pueden utilizar reglas simbólicas o aprender un modelo numérico, y también pueden adaptar su comportamiento analizando cómo se ve afectado el entorno por sus acciones anteriores.” [12]

Para tratar a IA de una forma más clara y concisa se puede decir que es un campo de la informática que consiste en la creación de sistemas que a través de las matemáticas, sean capaces de realizar tareas que requieren inteligencia como puede ser el razonamiento o el aprendizaje [15][16][17]. Estos sistemas son y serán fundamentales en sectores como la industria, los servicios, y las comunicaciones, impulsando cambios profundos en la automatización, eficiencia y análisis predictivo. Será capaz de optimizar tareas en áreas como logística, manufactura, atención al cliente y servicios de salud, donde su capacidad para procesar grandes volúmenes de datos mejorará la toma de decisiones y la personalización de servicios. Este avance transformará el mercado laboral, creando nuevos roles en tecnología y análisis de datos, pero también desplazando algunos empleos tradicionales[3][18].

El funcionamiento de un sistema de inteligencia artificial está dividido en tres piezas que interactúan entre sí. La primera es la recepción de un estímulo por parte de sus sensores, que son los que recogen la información. Aquí es donde llega la segunda pieza encargada de procesar esa información y tomar una decisión en base a ella. Decisión que llevará a cabo la tercera pieza llamada “actuador”. Nótese que un actuador puede ser hardware o software.[19]

En esta introducción de fundamentos teóricos únicamente se tratará la rama de la IA denominada machine learning(ML) ya que es la rama a la que pertenece el tipo de algoritmo que será empleado en este proyecto para predecir las ventas de un comercio.

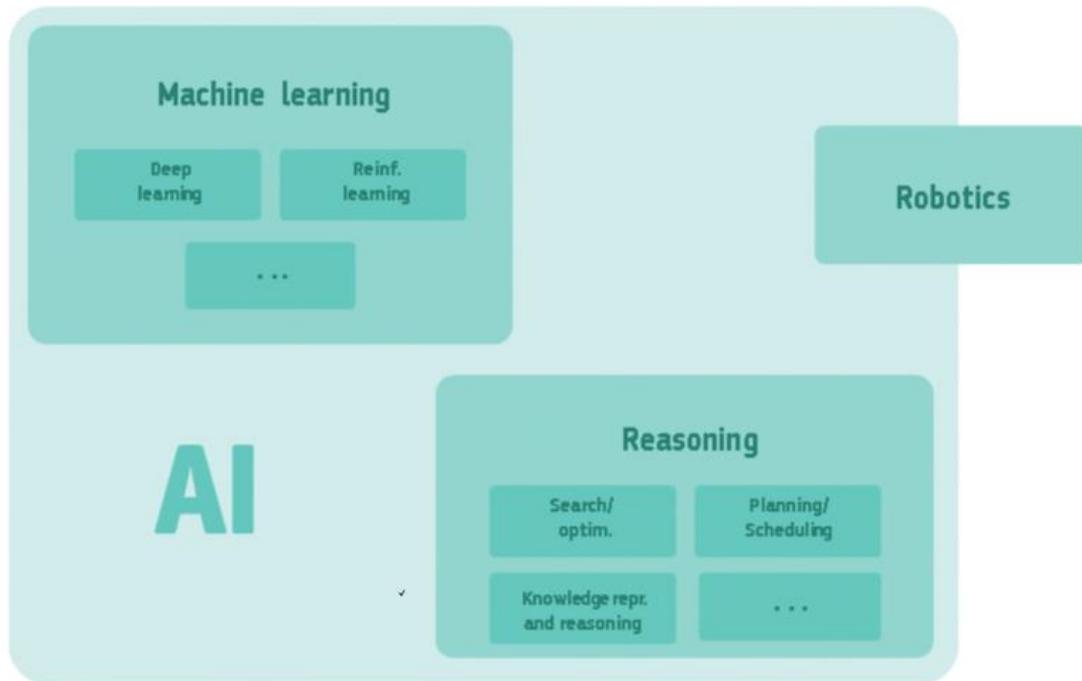


Figura 2.1 Una visión simplificada de las subdisciplinas de la IA y su relación²

2.1.1.2 Modelos

Machine learning (ML) se refiere a la rama de la IA que, basándose en un conjunto de datos iniciales, etiquetados o no, utilizan algoritmos que generan una predicción o una clasificación. Una parte relevante de estos algoritmos es que se corrigen ellos mismos, de tal forma que el error cometido en sucesivas iteraciones va reduciéndose -aumentando la precisión-. Existen diferentes técnicas de optimización de estos algoritmos como por ejemplo aumentar o disminuir el conjunto de datos de entrenamiento[19][20]. Hay dos tipos principales: El aprendizaje supervisado y el aprendizaje no supervisado [21].

Un clasificador supervisado parte de unos datos etiquetados. Esos datos, en función de la política tomada se dividen en dos conjuntos. Uno de entrenamiento y otro de test. Además, estos datos están clasificados en clases, de tal forma que un dato x pertenece a la clase C_i . El algoritmo aprende de estas relaciones a través del conjunto de datos de entrenamiento y más adelante medirá su precisión utilizando el conjunto de test. Esas muestras del conjunto

² Fuente : High-Level Expert Group on Artificial Intelligence

de test serán asignadas a una clase siguiendo el criterio de la función de decisión $f(x)$ tal que:

$$f(x) = \operatorname{argmax}_i f_i(x)$$

donde $f(x)$ maximiza el valor. La función varía dependiendo del algoritmo.

Los algoritmos supervisados más empleados son los clasificadores lineales, el *Support Vector Machine (SVM)* y las redes de neuronas[22].

A diferencia de los clasificadores supervisados, los clasificadores no supervisados parten de un conjunto de datos sin clasificar, es decir sin etiquetas, y tratan de identificar patrones en los datos. Al final del proceso, los datos estarán divididos en clusters o grupos. Uno de los algoritmos más comunes es el de *K-Means* donde los datos de entrada se dividirán en K grupos. Cada dato se clasificará en un grupo calculando la distancia al centroide de cada cluster K. Con cada iteración, cada cluster formado por cada dato junto con su centroide μ_k se actualizarán.

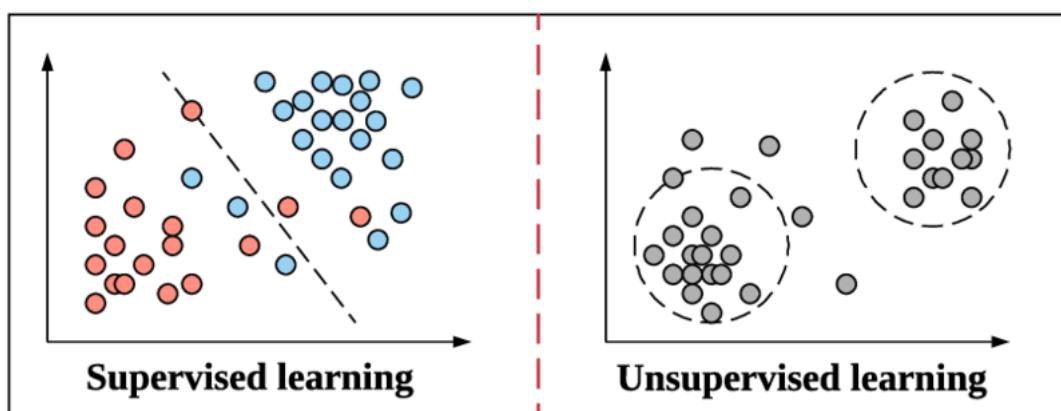


Figura 2.2 Aprendizaje supervisado vs Aprendizaje no supervisado³

Para concluir este apartado, es necesario definir lo que son las redes de neuronas, así como sus partes principales y su funcionamiento. Las redes neuronales o redes de neuronas (**RN**) son modelos de aprendizaje automático inspirados en el cerebro, que procesan datos a través de neuronas artificiales conectadas en capas: una de entrada, una o más capas ocultas, y una capa de salida. Cada neurona recibe señales ponderadas de otras neuronas, aplica una función de activación y transmite el resultado solo si supera un umbral. Esta arquitectura permite aprender patrones complejos en los datos mediante la adaptación de los pesos de conexión entre las neuronas a través del entrenamiento. Las redes pueden identificar características jerárquicas en datos, como imágenes o texto, y son ampliamente aplicadas en reconocimiento de patrones y predicción de secuencias[23].

³ Fuente: https://www.researchgate.net/figure/Examples-of-Supervised-Learning-Linear-Regression-and-Unsupervised-Learning_fig3_342761996

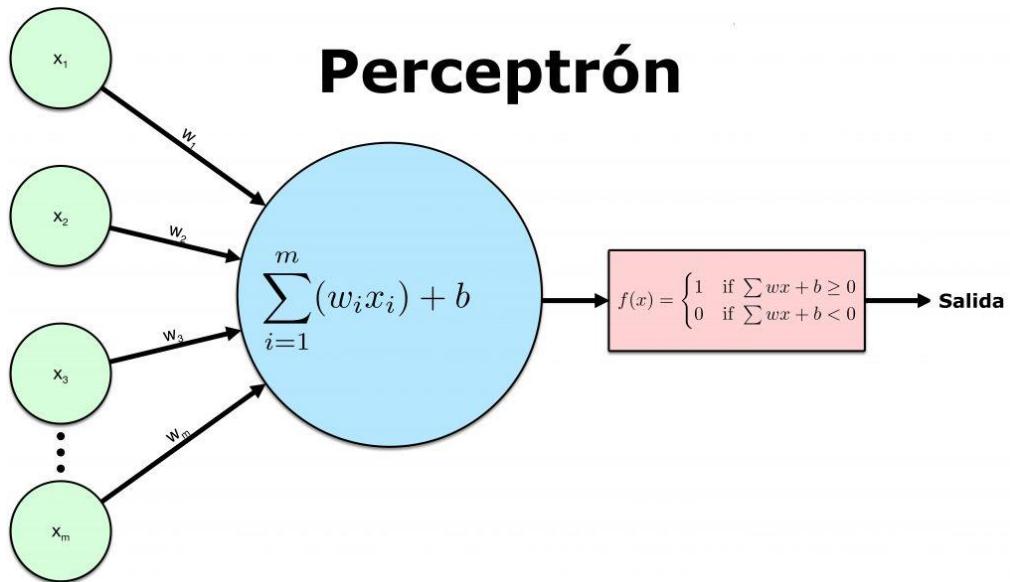


Figura 2.3 Modelo del perceptrón. Fuente: José Mariano Álvarez

Dentro del concepto de redes de neuronas, tenemos las redes de neuronas recurrentes (RNN), especializadas en procesar datos secuenciales o series temporales. Según el libro "Introducción al análisis de series temporales" de José Alberto Mauricio, una serie temporal se define como:

"Una serie temporal es una secuencia de observaciones de una variable recogidas en momentos sucesivos, generalmente con intervalos de tiempo constantes entre ellas." [24]

Las RNN nacieron con el objetivo de extender la capacidad de las redes neuronales tradicionales permitiendo que el modelo no solo procese cada dato de forma aislada, sino que también tenga en cuenta el contexto previo. Su origen se remonta a los años 80 y 90, cuando se propusieron los primeros modelos capaces de incorporar una memoria interna que se actualiza en cada paso de tiempo, haciendo posible el aprendizaje de dependencias temporales. Están compuestas de 3 tipos de capas: la capa de entrada, la capa oculta y la capa de salida.

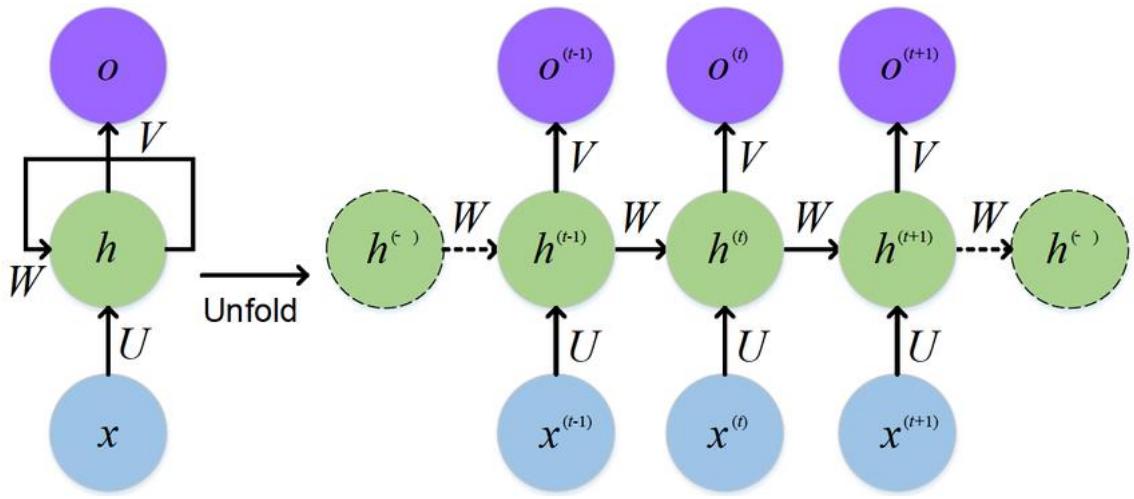


Figura 2.4 Representación de una red neuronal recurrente (RNN)

En la Figura 2.4 Representación de una red neuronal recurrente (RNN), se muestra la estructura compacta de la RNN con retroalimentación en el estado oculto h , mientras que a la derecha se observa su versión desplegada en el tiempo, donde cada entrada $x(t)$ se procesa junto con el estado oculto anterior $h^{(t-1)}$, para generar un nuevo estado oculto $h^{(t)}$, que a su vez produce una salida $o^{(t)}$. [25]. Entrando brevemente en su funcionamiento y como se ha mencionado antes el estado oculto $h^{(t)}$ se calcula de la siguiente forma:

$$h^{(t)} = f(W_h h^{(t-1)} + W_x x^{(t)} + b)$$

Posteriormente, genera la salida correspondiente $y(t)$ aplicando una transformación lineal sobre el estado oculto:

$$y^{(t)} = g(W_{hy} h^{(t)} + c)$$

En estas expresiones, W_x , W_h , W_y , son matrices de pesos que determinan cómo influyen respectivamente la entrada actual, el estado anterior y el estado oculto en la salida; b y c son los vectores de sesgo (bias); f es una función de activación que introduce no linealidad; y $x^{(t)}$, $x^{(t)}$, $x^{(t)}$ representan, respectivamente, la entrada, el estado oculto y la salida en el instante ' t '. Este mecanismo permite que la red mantenga una memoria a lo largo del tiempo, adaptando su comportamiento en función de lo aprendido de secuencias previas. [26]

Gracias a ello, las RNN son capaces de modelar fenómenos donde el orden y la evolución de los datos en el tiempo son fundamentales, como en:

- El lenguaje natural. [27]
- La predicción financiera, de la demanda y del clima. [28]
- Análisis de series temporales en general.

Sin embargo, las RNN tradicionales presentan limitaciones cuando se trata de secuencias largas, debido a problemas como la desaparición o explosión del gradiente durante el entrenamiento. La desaparición del gradiente ocurre cuando los valores de los gradientes se vuelven tan pequeños que dejan de tener impacto en la actualización de los pesos, haciendo que la red "olvide" la información anterior. Por otro lado, la explosión del gradiente se produce cuando estos valores crecen de forma descontrolada, lo que puede provocar inestabilidad numérica y resultados erráticos.

Por ejemplo, si una red neuronal intenta aprender la relación entre el número de horas estudiadas y la nota obtenida, y predice una calificación incorrecta (por ejemplo, predice un 6 cuando el valor real es 8), la red calcula ese error y lo propaga hacia atrás usando el algoritmo "Backpropagation Through Time" (BPTT). Una forma típica de calcular el error es a través del error cuadrático medio (MSE) cuya fórmula es la siguiente:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Luego, mediante el ajuste del gradiente, modifica sus pesos internos para mejorar su rendimiento. Sin embargo, si la secuencia de datos es muy larga, como en un historial educativo extenso, los gradientes pueden desaparecer o explotar, haciendo que la red no aprenda correctamente.

Para superar estas dificultades, en 1997 se propone una alternativa. La arquitectura "Long Short-Term Memory" (LSTM), que introduce mecanismos de control internos —como puertas de entrada, olvido y salida— para regular qué información se conserva y cuál se descarta, permitiendo un aprendizaje más estable en secuencias largas.[29][30]. A continuación se ofrece una breve explicación sobre su funcionamiento:

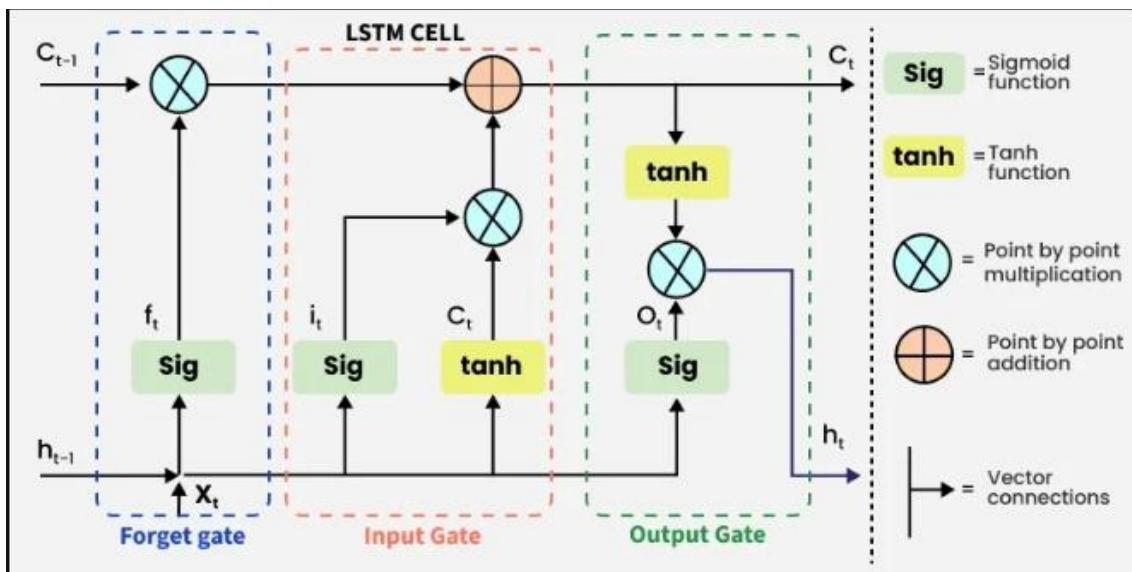


Figura 2.5 Representación esquemática de una celda LSTM, mostrando las puertas de olvido, entrada y salida, así como el flujo de información interna⁴.

- **Partes principales y fórmulas.**

Como se ha mencionado anteriormente, existen 3 tipos de puertas. Las puertas de entrada, las de olvido y las de salida; y se relacionan de la siguiente forma para formar una celda:

1. **Puerta de olvido**

$$ft = \sigma(Wf \cdot [ht - 1, xt] + bf)$$

2. **Puerta de entrada**

$$it = \sigma(Wi \cdot [ht - 1, xt] + bi)$$

3. **Información útil**

$$\hat{C}t = \tanh(WC \cdot [ht - 1, xt] + bc)$$

4. **Candidato**

$$Ct = ft \cdot Ct - 1 + it \cdot \hat{C}t$$

⁴ Fuente: [GeeksforGeeks](#)

5. Puerta de salida

$$ot = \sigma(Wo \cdot [ht - 1, xt] + bo)$$

6. Salida de la celda

$$ht = ot \cdot \tanh(Ct)$$

- **Explicación.**

En una celda LSTM, las puertas de olvido, entrada y salida son cruciales para regular el flujo de información y el estado de la memoria. La puerta de olvido, ft , determina qué parte de la información previa se debe descartar, calculándose mediante una función sigmoide σ . La puerta de entrada, it , decide qué nueva información debe almacenarse en la celda, mientras que el candidato a nuevo estado o información útil, \hat{Ct} , define la nueva información que será añadida al estado de la celda. El estado de la celda Ct se actualiza como la combinación ponderada de la información anterior y la nueva información (controlada por la puerta de entrada y la información útil). Finalmente, la puerta de salida, ot , decide qué parte del estado de la celda se usará para generar la salida ht de la celda.[29][30]

Nótese que en las fórmulas 1) 2) 3) y 5) aparece un término “ b_{indice} ”. Éste se refiere a un sesgo o “bias”; y en función del objetivo puede tomar un valor u otro.

En cuanto a los valores que pueden tomar las distintas variables, las puertas ft , it y ot devuelven valores entre 0 y 1 gracias a la función sigmoide, mientras que \hat{Ct} y ht devuelven valores entre -1 y 1 por usar la tangente hiperbólica.

2.1.1.3 Evaluación de modelos

A continuación, se introducen cuatro conceptos —MAE, MSE, overfitting y underfitting— que serán fundamentales para la evaluación del rendimiento del modelo y comprender por qué un modelo es mejor que otro al evaluar unos datos u otros.

El Error Absoluto Medio (MAE) y el Error Cuadrático Medio (MSE) son dos de las métricas más comunes utilizadas para evaluar el rendimiento de modelos predictivos. El MAE mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección (es decir, sin importar si el valor predicho es mayor o menor que el real). Un valor de MAE más bajo indica una mayor precisión del modelo en términos absolutos. Por otro lado, el MSE penaliza más fuertemente los errores grandes debido a que estos errores se elevan al cuadrado en su cálculo. Este enfoque hace que el MSE sea más sensible a los valores atípicos, lo que puede ser útil cuando se desea que el

modelo sea más sensible a grandes errores, pero también puede llevar a que el modelo se ajuste excesivamente a los mismos. En resumen, el MAE es más robusto frente a valores atípicos, mientras que el MSE proporciona un enfoque más sensible a ellos.**[31]**

El overfitting y el underfitting son dos problemas comunes que pueden afectar a los modelos y son muy útiles para comprender la dinámica de las métricas como el MAE y el MSE. El overfitting ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento, lo que resulta en una excelente precisión en esos datos, pero en una pobre capacidad para generalizar a nuevos datos. Esto sucede cuando el modelo es demasiado complejo, con demasiados parámetros, lo que lleva a una alta varianza. Un modelo sobreajustado tiene un bajo error en el conjunto de entrenamiento, pero un alto error en el conjunto de prueba, ya que no es capaz de capturar las tendencias generales de los datos y se ajusta a las fluctuaciones aleatorias de este conjunto. Para evitar el overfitting, se pueden usar técnicas como la regularización o reducir la complejidad del modelo.

El underfitting, en cambio, ocurre cuando un modelo es demasiado simple para capturar la estructura subyacente de los datos, lo que provoca un rendimiento deficiente tanto en los datos de entrenamiento como en los de prueba. Esto puede suceder si el modelo tiene demasiados pocos parámetros o si no se ha entrenado adecuadamente, lo que lleva a un alto sesgo y baja varianza. En estos casos, el modelo no logra aprender patrones importantes y, por lo tanto, su rendimiento es deficiente. La clave para evitar el underfitting es asegurarse de que el modelo tenga suficiente capacidad para aprender los patrones en los datos, sin caer en el riesgo del overfitting. **[32] [33]**

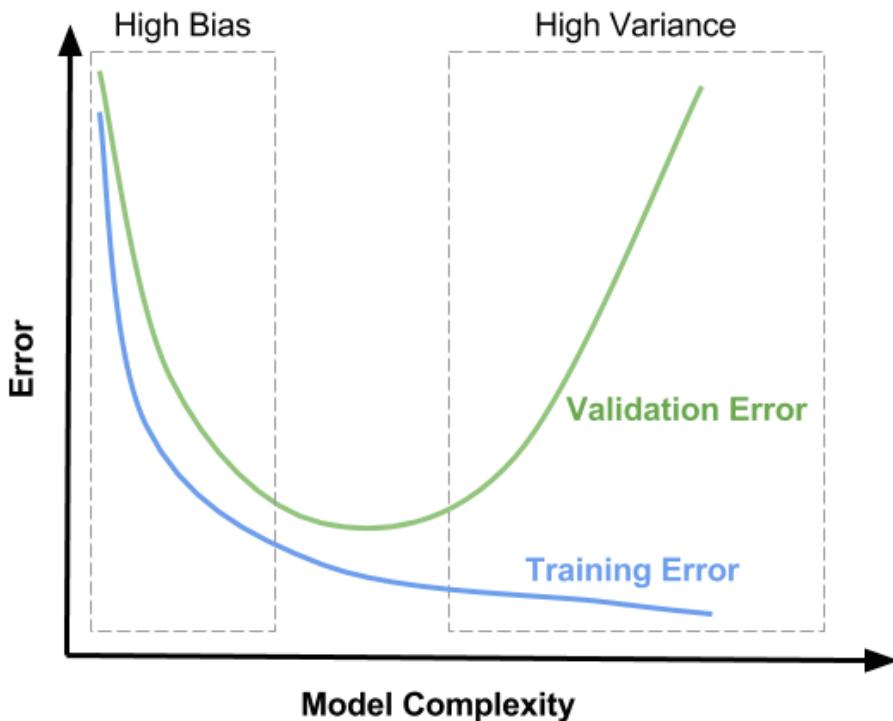


Figura 2.6 Curva de relación entre complejidad y error⁵.

2.1.1.4 Otras técnicas de evaluación

Para finalizar este apartado de “Inteligencia Artificial” y para dar soporte a las métricas descritas en el subapartado de “Evaluación de modelos”, se presentan tres métricas; media móvil exponencial, mapas de calor y matrices de correlación.

Las medias móviles exponenciales son un método de suavizado de series temporales que otorga mayor peso a los datos recientes, permitiendo detectar tendencias a corto plazo en series irregulares. Se calculan recursivamente como:

$$zt = \lambda xt + (1 - \lambda)zt - 1, \text{ donde } 0 < \lambda \leq 1$$

Esta técnica es útil para comparar las curvas reales y predichas, ya que muestra las diferencias en la tendencia

Los heatmaps o mapas de calor, se utilizan para representar visualmente el rendimiento (por ejemplo, MAE o MSE) en el espacio de hiperparámetros, facilitando la identificación de configuraciones con buen desempeño y descartar las que tengan un mal desempeño.

⁵ Fuente: "LearnOpenCV"

Finalmente, las matrices de correlación cuantifican la relación entre valores de hiperparámetros (batch size, épocas, etc.) y métricas de evaluación. Un coeficiente Pearson (r) alto indica que un parámetro influye directamente en el desempeño.

Estas métricas se utilizan con frecuencia en la literatura científica para apoyar el análisis y la validación de modelos en distintos contextos de investigación.**[34][35][36]**

2.1.2 Open Data

El Open Data o "datos abiertos" se refiere a aquellos datos accesibles al público sin restricciones de uso, distribución o modificación. Deben estar en formatos reutilizables, y que cualquier persona pueda acceder a ellos, usarlos, analizarlos y redistribuirlos de forma gratuita**[37]**. Los datos abiertos pueden provenir de diferentes fuentes, como gobiernos**[2]** u organizaciones internacionales**[38]** y pueden incluir desde estadísticas económicas y datos geográficos hasta datos de investigación científica o datos meteorológicos.

El movimiento de Open Data surgió en la década de 2000 como parte de un esfuerzo por promover la transparencia en la gestión pública y la democratización del acceso a la información. La idea central es que el gobierno y otras entidades públicas deberían proporcionar los datos que generan con los fondos públicos de forma abierta, lo que facilita el acceso a la información y permite a los ciudadanos, organizaciones y empresas aprovecharla para la creación de nuevas soluciones y servicios **[39]**.

Es importante mencionar algunas de estas iniciativas nacidas en la Unión Europea, Reino Unido y Estados Unidos. Por ejemplo, la *Open Knowledge Foundation*, fundada en 2004 por Rufus Pollock, que desarrolla herramientas para que otras organizaciones compartan sus datos y los mantengan actualizados **[1]**.

Los objetivos del Open Data son promover la transparencia, la rendición de cuentas, la innovación y la eficiencia. Estos objetivos se cumplirán cuando cualquier persona pueda acceder a los datos y utilizarlos para crear nuevos productos, servicios o investigaciones que beneficien a la sociedad en general. Entre las metas clave del Open Data se incluyen:

- **Promover la transparencia:** Permitiendo a los ciudadanos comprobar si los recursos públicos se están empleando bien.
- **Fomentar la innovación:** Promover la creación de nuevas soluciones tecnológicas.
- **Impulsar la colaboración:** Facilitar el trabajo entre instituciones, investigadores y empresas para abordar retos globales.

Aplicaciones del Open Data Las aplicaciones del Open Data son vastas y variadas **[40]**. Abarcan áreas como:

1. **Gobernanza y Transparencia:** Los ciudadanos pueden acceder a datos gubernamentales, como presupuestos, gastos y políticas públicas, lo que fomenta la participación y la rendición de cuentas.
2. **Investigación y Ciencia:** Los investigadores pueden aprovechar grandes volúmenes de datos para avanzar en sus estudios, desde biomedicina hasta investigaciones sobre el cambio climático.
3. **Desarrollo de Nuevas Tecnologías:** Las empresas pueden utilizar datos abiertos para desarrollar nuevas aplicaciones y tomar decisiones más acertadas.
4. **Servicios Públicos:** Los gobiernos pueden usar los datos abiertos para mejorar los servicios a los ciudadanos, como el transporte público, la planificación urbana, la salud pública, etc.

España, la iniciativa de Open Data ha crecido, pero aún está en desarrollo. A nivel gubernamental, el portal **datos.gob.es** se ha establecido como el principal punto de acceso a los datos abiertos del sector público. Sin embargo, la accesibilidad y la reutilización de estos datos no son siempre sencillas. Muchos datos están dispersos entre distintas administraciones, y aunque el portal ha mejorado, aún no existe una completa integración y estandarización de la información. Además, la ciudadanía en general aún no hace uso de la gran mayoría de los datos disponibles debido a la falta de conocimiento o de herramientas para aprovechar esos recursos de manera efectiva.

No obstante, en los últimos cinco años, la evolución de las empresas y aplicaciones que reutilizan los datos públicos españoles ha experimentado un notable aumento, con un crecimiento que ha llegado a duplicarse. Este auge en el uso de datos abiertos se debe al creciente interés por parte de las empresas, administraciones públicas y ciudadanos por aprovechar la información disponible. Sin embargo, como mencionábamos anteriormente, los datos públicos en España siguen estando fragmentados y almacenados en diversos formatos, lo que dificulta su utilización de forma efectiva. Por ejemplo, el 45% de los datos disponibles en **datos.gob.es** provienen de administraciones autonómicas. A continuación, se ofrece un desglose de estos datos por formato, destacando los diferentes tipos de archivos. Esto refleja la necesidad de una estandarización y unificación de los datos.

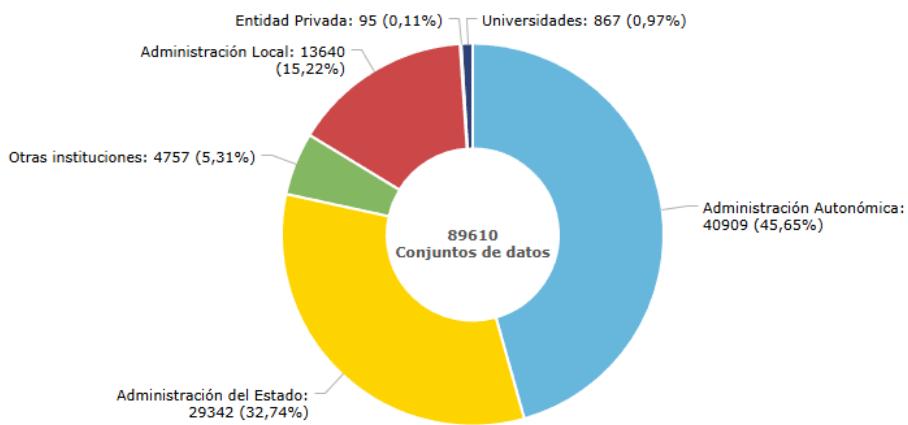


Figura 2.7 Conjuntos de datos por nivel de administración⁶.

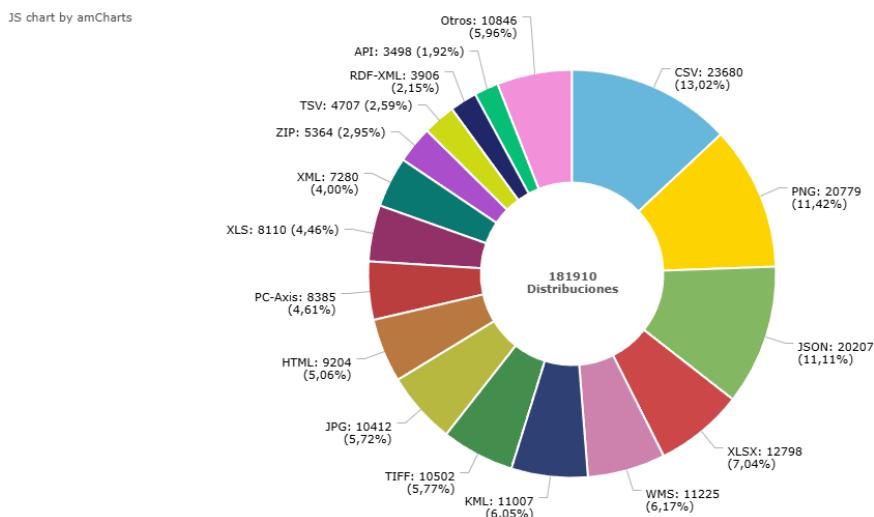


Figura 2.8 Porcentaje de distribuciones por formato por nivel de administración⁷.

El Open Data se rige por varios principios fundamentales [41] que son los siguientes:

1. **Accesibilidad:** Los datos deben ser fácilmente accesibles a través de internet, preferiblemente sin restricciones técnicas.

⁶ Fuente: <https://datos.gob.es/es>

⁷ Fuente : <https://datos.gob.es/es/dashboard>

2. **Transparencia:** Los datos deben reflejar la realidad de manera clara y sin manipulaciones.
3. **Interoperabilidad:** Los datos deben estar en formatos que puedan ser fácilmente combinados y utilizados con otros conjuntos de datos.
4. **Licencia abierta:** Los datos deben ser reutilizables bajo licencias que permitan su modificación y distribución sin restricciones comerciales.
5. **Completitud:** Los conjuntos de datos deben ser completos, incluyendo todos los valores relevantes sin omitir información significativa.

Varias organizaciones están a la vanguardia del movimiento Open Data:

- **Open Knowledge Foundation:** Una de las organizaciones pioneras en la promoción de la cultura del Open Data. Ofrece recursos, estándares y plataformas que facilitan el acceso y la reutilización de los datos.
- **Open Data Charter:** Un conjunto de principios internacionales promovido por gobiernos y organizaciones globales para establecer normas en la implementación de políticas de datos abiertos.
- **The World Bank:** A través de su portal de Open Data, esta organización ofrece acceso a datos globales sobre desarrollo económico, social y ambiental.
- **European Data Portal:** Impulsado por la Unión Europea, este portal proporciona acceso a un gran volumen de datos abiertos de más de treinta países.

2.2 Tecnologías

A continuación, se describirán las diferentes tecnologías y herramientas empleadas en el desarrollo de este proyecto⁸, que cumplirán con distintos roles como cubrir la necesidad de un lenguaje de programación, manejar el tratamiento de datos y gestionarlos o desplegar el servicio web.

2.2.1 Generales

Java es un lenguaje de programación de alto nivel, orientado a objetos. Fue desarrollado en 1995 por *Sun Microsystems* con la visión de crear un lenguaje que pudiera ejecutarse en cualquier dispositivo mediante la famosa filosofía WORA, “Write Once, Run Anywhere”. Java se destaca por su capacidad para ejecutarse en diversas plataformas sin necesidad de recompilación [42]. En este

⁸ En el capítulo de Desarrollo se proporciona un diagrama de arquitectura con las diferentes tecnologías empleadas y la relación entre ellas.

proyecto, Java será el lenguaje principal para la implementación tanto del backend como del acceso a bases de datos y manejo de datos.

Según el índice TIOBE, que clasifica los lenguajes de programación más populares en función de su frecuencia de búsqueda en Internet, Java sigue siendo uno de los lenguajes más utilizados a nivel mundial [43]. Según una encuesta de *Stack Overflow* (2024), más del 30% de los desarrolladores usan Java, especialmente en aplicaciones empresariales, desarrollo Android y sistemas integrados. Java se utiliza principalmente en empresas grandes debido a su estabilidad, seguridad y ecosistema maduro, especialmente en sectores como la banca, seguros, telecomunicaciones y desarrollo de software [44].

Es un lenguaje que ofrece varias ventajas clave para el desarrollo de software. En primer lugar, su robustez se debe en gran parte a un manejo de memoria eficiente y la ausencia de punteros, lo cual reduce vulnerabilidades de seguridad. Su portabilidad es otra gran ventaja: al compilarse en bytecode, el código Java puede ejecutarse en cualquier sistema que soporte la Java Virtual Machine (JVM), lo que facilita su implementación en múltiples plataformas. Finalmente, la capacidad multihilo permite que Java ejecute varias tareas de manera concurrente, optimizando el uso de recursos y mejorando la eficiencia en aplicaciones de gran escala [45].

Python es un lenguaje de programación interpretado, de alto nivel y propósito general. Fue creado por Guido van Rossum y se publicó por primera vez en 1991. Es conocido por su sintaxis clara y legible, lo que facilita su aprendizaje y uso en una amplia variedad de campos como el desarrollo web, la automatización, la ciencia de datos y la inteligencia artificial. Su gran ecosistema de bibliotecas lo convierte en una herramienta muy versátil para el análisis y procesamiento de datos.[46]

Draw.io es otra herramienta empleada en este proyecto. Se utilizará para la creación de diagramas de la arquitectura del sistema, diagramas de clases UML e incluso el modelo entidad-relación necesario para desarrollar la base de datos.

2.2.2 Tratamiento y Almacenamiento de Datos

Este apartado está dedicado a las tecnologías que se utilizarán para gestionar y procesar los datos del proyecto, desde su manipulación y conversión entre diferentes formatos hasta su almacenamiento en bases de datos.

MySQL es el sistema de gestión de bases de datos relacional (RDBMS) que se empleará para almacenar la información obtenida de las APIs de AEMET e INE. MySQL es uno de los RDBMS más populares [44][47] y utilizados en el mundo. MySQL facilita la organización y consulta de grandes volúmenes de datos a través de SQL, proporcionando una alta eficiencia en el manejo de datos estructurados.

Gson es una librería de Google diseñada para simplificar el proceso de conversión entre JSON y objetos Java. Este proceso se le conoce como deserialización o serialización [48]. El motivo principal por el cual será empleada es la de cubrir la necesidad del proyecto de manejar un alto volumen de datos provenientes de APIs.

Apache POI es la librería que se utiliza para la lectura y escritura de diferentes archivos de Microsoft como Word, Excel o PowerPoint. Esta herramienta facilita la lectura, escritura y modificación de documentos Excel desde Java. Dado que algunos de los datos relevantes para el proyecto se proporcionan en archivos Excel, Apache POI será fundamental para extraer la información necesaria de manera eficiente y ordenada. La unidad básica con la que trabaja Apache POI es el “Workbook”, que representa el archivo Excel completo. Dentro de un Workbook, los datos se encuentran organizados en “Sheets”, que contienen filas y celdas. Cada Sheet se compone de un conjunto de “Rows”, y cada Row a su vez está formado por celdas, donde se almacena la información específica.[49]

Java Database Connectivity (JDBC) es la API estándar de Java que permite conectar las aplicaciones con bases de datos. A través de JDBC, se puede establecer una conexión con la base de datos MySQL, enviar consultas SQL y procesar los resultados devueltos. El uso de JDBC en este proyecto es esencial para interactuar con la base de datos de manera eficiente, permitiendo realizar operaciones como la inserción, actualización y eliminación de registros, así como la ejecución de consultas complejas sobre los datos almacenados [50].

<i>Driver</i>	Responsable de gestionar la conexión con la base de datos.
<i>Connection</i>	La conexión con la base de datos que permite la creación de objetos.
<i>Statement</i>	Permite ejecutar consultas SQL. Hay dos tipos principales: Statement y PreparedStatement
<i>ResultSet</i>	Objeto con los resultados de la consulta que permite recorrerlos por filas.
<i>SQLException</i>	Maneja las excepciones principales.

Tabla 2.1 Unidades básicas de JDBC y su función

GeoTools es una biblioteca de código abierto para Java que proporciona herramientas para el manejo y procesamiento de datos geoespaciales. Su principal utilidad radica en ofrecer una serie de funciones y métodos para trabajar con datos geográficos, como coordenadas, mapas, y proyecciones cartográficas. La biblioteca está diseñada para facilitar la manipulación de información geoespacial en diversos formatos, como Shapefiles, GeoJSON, KML, GML, entre otros.

2.2.3 Servicio web: Servidor

En este apartado se abordarán las tecnologías necesarias para crear un servicio web que permita la consulta de los datos integrados en el sistema.

Apache Tomcat es el servidor web y contenedor de servlets que se utilizará para desplegar el servicio web. Tomcat es uno de los servidores más populares y de código abierto para ejecutar aplicaciones Java basadas en servlets y JavaServer Pages (JSP).

Spring Framework es un marco de trabajo de código abierto para Java diseñado para que desarrolladores implementen aplicaciones autónomas. Además, con una configuración mínima es posible desplegar la aplicación. Nos permite simplificar el desarrollo de API REST mediante anotaciones como las que se describen a continuación:

<code>@Service</code>	Indica que la clase contiene lógica de negocio.
<code>@Repository</code>	Se utiliza para definir componentes de acceso a datos.
<code>@Controller</code>	Marca una clase como un controlador para manejar solicitudes web.
<code>@RestController</code>	Combina <code>@Controller</code> y <code>@ResponseBody</code> , utilizada para controladores REST.

Tabla 2.2 Anotaciones generales Spring

<code>@GetMapping</code>	Maneja solicitudes HTTP GET.
<code>@PostMapping</code>	Maneja solicitudes HTTP POST.
<code>@PutMapping</code>	Maneja solicitudes HTTP PUT.
<code>@DeleteMapping</code>	Maneja solicitudes HTTP DELETE.
<code>@RequestParam</code>	Vincula parámetros de consulta (query parameters) a argumentos de métodos.
<code>@RequestBody</code>	Vincula el cuerpo de una solicitud HTTP a un objeto Java.
<code>@ResponseBody</code>	Indica que el valor de retorno de un método debe convertirse en JSON o XML. Mejor utilizar <code>@RestController</code>

Tabla 2.3 Anotaciones de gestión web y rest Spring

<code>@Entity</code>	Define una clase como una entidad.
<code>@Id</code>	Marca el campo que será la clave primaria de la entidad.
<code>@GeneratedValue</code>	Define cómo se genera el valor de la clave primaria

<code>@Table</code> <code>@ManyToOne, @OneToMany,</code> <code>@ManyToMany, @OneToOne</code>	Especifica la tabla a la que está asociada una entidad. Configuran relaciones entre entidades.
--	---

Tabla 2.4 Anotaciones de acceso a datos Spring

2.2.4 Servicio web: Cliente y LSTM

En este apartado se explicarán de forma muy breve todas las librerías empleadas en el desarrollo del modelo y procesamiento de datos. Estas librerías permiten desde la recolección de datos hasta la construcción, evaluación y visualización de modelos de predicción. Las librerías utilizadas son las siguientes:

- **requests:** Permite realizar peticiones HTTP para obtener datos desde APIs o servicios web.
- **pandas:** Facilita la manipulación y análisis de datos estructurados en forma de tablas (dataframes).
- **numpy:** Proporciona herramientas para realizar operaciones matemáticas y trabajar con arrays de forma eficiente.
- **holidays:** Permite identificar días festivos oficiales según país o región, útil en análisis temporales.
- **sklearn.preprocessing:** Contiene herramientas para escalar y transformar datos. MinMaxScaler ajusta los valores a un rango específico (por ejemplo, [0, 1]), mientras que StandardScaler normaliza los datos con media 0 y desviación estándar 1.
- **tensorflow.keras:** Este módulo permite construir y entrenar redes neuronales. Dentro de él, Sequential se utiliza para crear modelos secuenciales capa por capa; LSTM añade capas de memoria a largo corto plazo, útiles en series temporales; Dense incorpora capas totalmente conectadas, y Dropout aplica regularización para evitar el sobreajuste desconectando aleatoriamente algunas neuronas durante el entrenamiento.
- **matplotlib.pyplot:** Utilizada para crear gráficos como líneas, barras o diagramas de dispersión.
- **sklearn.metrics:** Métricas para evaluar el rendimiento de modelos.
- **itertools:** Contiene herramientas para construir iteradores complejos como combinaciones o productos cartesianos.
- **seaborn:** Librería para visualización estadística avanzada basada en Matplotlib, que permite generar gráficos.

2.3 Proyectos Similares

En los últimos años, han surgido diversas iniciativas con el objetivo de utilizar los datos abiertos que proporcionan los gobiernos. Por otro lado, también hay multitud de iniciativas para optimizar los negocios mediante la implementación de sistemas basados en IA. Y es que la IA proporciona ventajas notables en el ámbito empresarial, tales como la eficiencia operativa, la mejora en la personalización de servicios, y el análisis de grandes volúmenes de información para predecir patrones y tendencias de mercado.

Estas iniciativas responden a la necesidad de adaptar las organizaciones al entorno digital y globalizado, en el que los datos se convierten en un recurso fundamental. Los sistemas de IA pueden aprovechar estos datos abiertos para automatizar procesos y desarrollar estrategias comerciales personalizadas y competitivas. Sin embargo, el éxito de la IA en la mejora de decisiones empresariales depende en gran medida de la disponibilidad y calidad de los datos, lo cual subraya la importancia del acceso a datos abiertos y que estos sigan determinadas normas[51][52].

Dicho esto, en este apartado se introducen algunos trabajos íntimamente relacionados que exploran la extracción de datos de fuentes abiertas o implementaciones de IA.

En primer lugar, hablaremos sobre proyecto *Madrid Green Data Spaces* (MGDS), financiado en 2020 por EIT Digital, tiene como objetivo desplegar un espacio de datos que permita analizar y visualizar la infraestructura verde de Madrid mediante indicadores relevantes. Para ello, se diseñó un Producto Mínimo Viable (MVP) que incluye un catálogo de datos y un dashboard interactivo que presenta métricas como el *Green Space Index*, que muestra el porcentaje de área verde por persona en cada sección censal, la densidad de espacio verde, y la proximidad de estas zonas para los habitantes. Este sistema se basa en datos abiertos del Ayuntamiento de Madrid, el Instituto Geográfico Nacional y Copernicus, además de datos LIDAR y satelitales. Este proyecto, desarrollado por el Grupo de Ingeniería Ontológica y el GIIS de la Universidad Politécnica de Madrid, destaca por la integración de datos abiertos junto con su utilización y visualización, que permiten a usuarios evaluar la calidad de la infraestructura verde urbana en la ciudad de Madrid.

El almacenamiento de los datos se realiza en PostgreSQL, y el acceso a algunos conjuntos se facilita mediante una API REST. Estos datos están en formatos como CSV, XLSX o XLS y se despliegan en una aplicación web accesible públicamente a través de una arquitectura IDS (International Data Spaces) [53].

Otro proyecto parecido al primero que acabamos de comentar es el TFM titulado "*Integración y Visualización de Datos Abiertos Medioambientales*". Este proyecto tiene como objetivo integrar datos abiertos medioambientales de diversas fuentes en España, como la Comunidad de Madrid, la Junta de Andalucía y el País Vasco, y visualizarlos en tiempo real. Utiliza datos de calidad del aire,

temperatura, radiación solar y otros indicadores climáticos. Las herramientas clave incluyen el lenguaje de programación R, la librería Shiny para la interfaz gráfica y scripts para automatizar la integración de datos en formatos como CSV. Este trabajo aborda la disparidad de formatos en los datos, destacando que "no existe un estándar internacional", lo que crea dificultades en la integración de los datos. Para optimizar el análisis comparativo y la toma de decisiones informadas en tiempo real, el TFM enfatiza la importancia del preprocesado, limpieza e integración de las fuentes de datos, así como evitar la supervisión humana durante el manejo de estos datos. La integración de datos heterogéneos es crucial para la optimización de la visualización y el uso de los datos ambientales públicos.

Este TFM afronta el mismo reto que el presente proyecto: la integración de datos pertenecientes a distintas fuentes en una única base de datos, con el objetivo de diseñar una aplicación con la que luego utilizar esos datos unificados. Del conocimiento obtenido de este TFM, puedo decir que sería interesante usar alguna herramienta estadística para añadir un valor extra a los datos obtenidos en crudo de mi proyecto, los cuales luego serán la entrada de mi red de neuronas. De esta forma, se podría mejorar la calidad de las predicciones y facilitar la toma de decisiones más precisas[54].

El tercer proyecto del que hablaremos es una iniciativa personal del Senior Data Scientist Ander Fernández cuyo trabajo se titula "*Análisis de la España despoblada*". Este trabajo tiene como objetivo principal presentar la evolución de la despoblación en España desde 1996 hasta 2018, tratando además la escasez de proyectos que presenten un análisis de este problema. Para ello, crea un dashboard con datos procedentes de fuentes como el Instituto Nacional de Estadística (INE) y el Centro Nacional de Información Geográfica (CNIG), y se centraliza la información sobre la población y la topografía de los municipios en formatos como shapefile o geoJSON.

Durante el desarrollo, el autor se enfrentó a problemas de coherencia en los nombres de los municipios y a su homogeneización a lo largo de los años, además de a las limitaciones de Excel para gestionar datos amplios. La solución consistió en una limpieza de datos para unificar nombres y, en algunos casos, en el uso de R para manejar grandes cantidades de datos y enriquecerlos con análisis de la variación de la población por comunidad autónoma y provincia. Tableau se utilizó para plasmar estos datos en un cuadro de mandos interactivo que permite al usuario explorar la evolución demográfica de manera visual y detallada[55].

Ander Fernández, el autor previamente mencionado, estuvo involucrado en otro proyecto similar titulado «*BasqueMinder*», desarrollado para el concurso de datos abiertos de Open Data Euskadi en 2020. El proyecto busca centralizar datos socioeconómicos del País Vasco para proporcionar a los técnicos y trabajadores públicos de Euskadi una herramienta que les permita tomar decisiones apoyadas en datos. Estos datos se obtienen de más de 210 fuentes públicas y van desde indicadores medioambientales hasta sostenibilidad. Los datos se extraen usando Python y se visualizan en mapas mediante la API REST

de “CartoCiudad”, una iniciativa del Ministerio de Transportes, Movilidad y Agenda Urbana que utiliza información del Centro Nacional de Información Geográfica.

Además de Python, el proyecto emplea Shiny, junto con JavaScript y CSS, para la visualización en la interfaz web. «*BasqueMinder*» ha logrado facilitar a los trabajadores públicos el acceso y uso de estos datos en una plataforma accesible y sencilla de usar en cuanto al formato o la búsqueda de datos[56].

El proyecto de Ander comparte aspectos clave con el proyecto presentado en este documento, como la extracción y tratamiento de datos públicos y su visualización en un servicio web. Inspirado en «*BasqueMinder*», se contemplará la posible integración futura de la API de “CartoCiudad”.

Si hablamos de proyectos empresariales o de administraciones públicas, el grado de complejidad aumenta notablemente. Es el caso de plataformas como *Salesforce Einstein* es la plataforma de inteligencia artificial de *Salesforce* que utiliza machine learning, procesamiento de lenguaje natural (NLP) y análisis predictivo para transformar la forma en que las empresas interactúan con sus clientes. Esta plataforma se integra con los datos de CRM de *Salesforce*, proporcionando insights clave sobre las ventas, el servicio al cliente y el marketing[57], permitiendo personalizar las recomendaciones de productos, optimizar los flujos de trabajo y mejorar las estrategias de ventas, algo que se alinea con el objetivo de mi TFG de mejorar las predicciones de ventas mediante IA. Dado que la complejidad de esta herramienta solo es replicable por otras empresas, la potencial utilidad de su revisión se ubica en los casos de uso que *Salesforce Einstein* presenta y en la forma de enfocar la red de neuronas que se aplicará en este Trabajo de Fin de Grado. Por ejemplo, sería útil basar el algoritmo en crear promociones de ventas en función del tiempo que hará esa semana y de la renta de ese municipio.

Las administraciones públicas no solo están ofreciendo sus datos, sino que también promocionan su uso, dando a conocer a empresas cuyo objetivo de negocio sea aplicar estos datos abiertos. Iniciativas como *datos.gob.es* y *Open Data Euskadi*, permiten que organizaciones tanto públicas como privadas aprovechen esta información para desarrollar nuevos proyectos. Ofrecen acceso a miles de conjuntos de datos, desde información sobre el clima hasta indicadores sociales. Ejemplos de proyectos nacidos gracias a estas iniciativas incluyen:

1. *Civio*: Un proyecto de periodismo que utiliza datos abiertos para promover la transparencia gubernamental. Esta iniciativa ha permitido crear plataformas que informan al público sobre la asignación de fondos públicos y la contratación del gobierno.
2. *Meteogrid*: Aplicación orientada al sector meteorológico que utiliza datos abiertos para generar pronósticos más precisos. Esta plataforma procesa información climática pública para ofrecer predicciones mejoradas para diferentes áreas, contribuyendo a la prevención de desastres naturales.

3. *TerceroB*: Un proyecto en el sector inmobiliario que usa datos abiertos para analizar la disponibilidad y precios de propiedades, ayudando a los usuarios a encontrar mejores opciones en el mercado.
4. *EuroAlert*: Una plataforma que monitoriza la contratación pública utilizando datos abiertos, permitiendo a las empresas y ciudadanos seguir la evolución de licitaciones y contratos gubernamentales.

Estas iniciativas demuestran cómo el acceso y la reutilización de datos abiertos pueden ser herramientas poderosas para transformar sectores, desde el medioambiente hasta la economía.

Si hablamos de proyectos cuyo tema principal sea nuestro objetivo 4, tenemos *LSTM Neural Network for Time Series Prediction* [58], un repositorio desarrollado por Jakob Aungiers⁹ que implementa redes neuronales LSTM con Keras y TensorFlow para abordar problemas de predicción en series temporales. El proyecto incluye ejemplos tanto con datos sintéticos (como funciones seno) como con datos reales del mercado bursátil (SP 500), y cubre todas las etapas del pipeline: desde la preparación de datos hasta el entrenamiento y evaluación del modelo. Resulta especialmente útil para entender cómo aplicar modelos a datos temporales multivariantes, algo directamente relacionado con nuestro objetivo. De este trabajo extraigo la capacidad de los modelos LSTM para predecir datos reales con ruido y complejidad, como los del mercado financiero. Además, valoro especialmente la importancia del preprocesado en el resultado final del modelo y su capacidad para capturar patrones largoplacistas.

En esta misma línea, está el notebook de Kaggle titulado *Step by Step Guide for Sales Data Prediction [LSTM]*, elaborado por Sanjayar[59]. Este trabajo está estrechamente vinculado de nuevo con una parte fundamental de mi proyecto: el uso de LSTM para la predicción de ventas. A lo largo del notebook se abordan tres fases clave: la carga y exploración de datos históricos de ventas, la limpieza para eliminar valores atípicos o no representativos, y la transformación en secuencias temporales mensuales que se puedan utilizar en el modelo LSTM. Me resultó especialmente útil la estructura clara del código previo al entrenamiento, así como el enfoque práctico para construir un modelo completo desde cero, destacando la importancia de la preparación de los datos y de la configuración del modelo para obtener resultados fiables.

⁹ Jaungiers. "LSTM Neural Networks for Time Series Prediction - IoT Data Science Conference - Jakob Aungiers

" YouTube, Enero 2017, <https://www.youtube.com/watch?v=2np77NOdnwk>. En este video, el autor explica paso a paso cómo implementar un modelo LSTM para la predicción de series temporales.

3 Desarrollo

Antes de entrar en detalle sobre cada parte del desarrollo del Trabajo de Fin de Grado, conviene explicar primero el enfoque general que se ha seguido y ofrecer una visión global de cómo se ha planificado y llevado a cabo el proyecto.

Como punto de partida, se hace referencia a los diagramas de Gantt, incluidos en el Anexo 8.1, que recogen la planificación temporal del proyecto y su división en tareas concretas agrupadas por objetivos. Estos diagramas se elaboraron en dos fases debido a una pausa en el desarrollo del trabajo: el primero ofrece un mayor nivel de detalle, desglosando actividades específicas dentro de cada objetivo; el segundo es más general y está enfocado a la revisión, mejora y finalización de los distintos bloques del TFG. Esta estructura ha sido fundamental para organizar y priorizar las tareas a lo largo del tiempo.

A continuación, se describe el diagrama de arquitectura general del sistema en la Figura 3.1 elaborado con Draw.io, que permite visualizar de forma estructurada los principales componentes del proyecto y sus relaciones. El sistema se divide en tres grandes bloques. El primero es un proyecto en Java centrado en la obtención y procesamiento de datos procedentes del INE y AEMET, dividido a su vez en dos fases: la recogida y tratamiento del dato, y su posterior almacenamiento. El segundo bloque lo constituye la base de datos relacional, que actúa como nexo entre los datos recopilados y el cliente-servidor. El tercer bloque corresponde al cliente-servidor REST, formado por un servidor en Java (basado en Spring y estructurado en capas repository, service y controller) y un cliente en Python encargado de realizar peticiones a la API, preprocesar los datos junto con los datos de ventas y aplicar el modelo LSTM. Esta parte del cliente también contempla el ajuste de hiperparámetros, el entrenamiento, la evaluación del modelo y la generación de predicciones. Finalmente, el sistema se complementa con scripts auxiliares dedicados a la visualización y análisis de resultados.

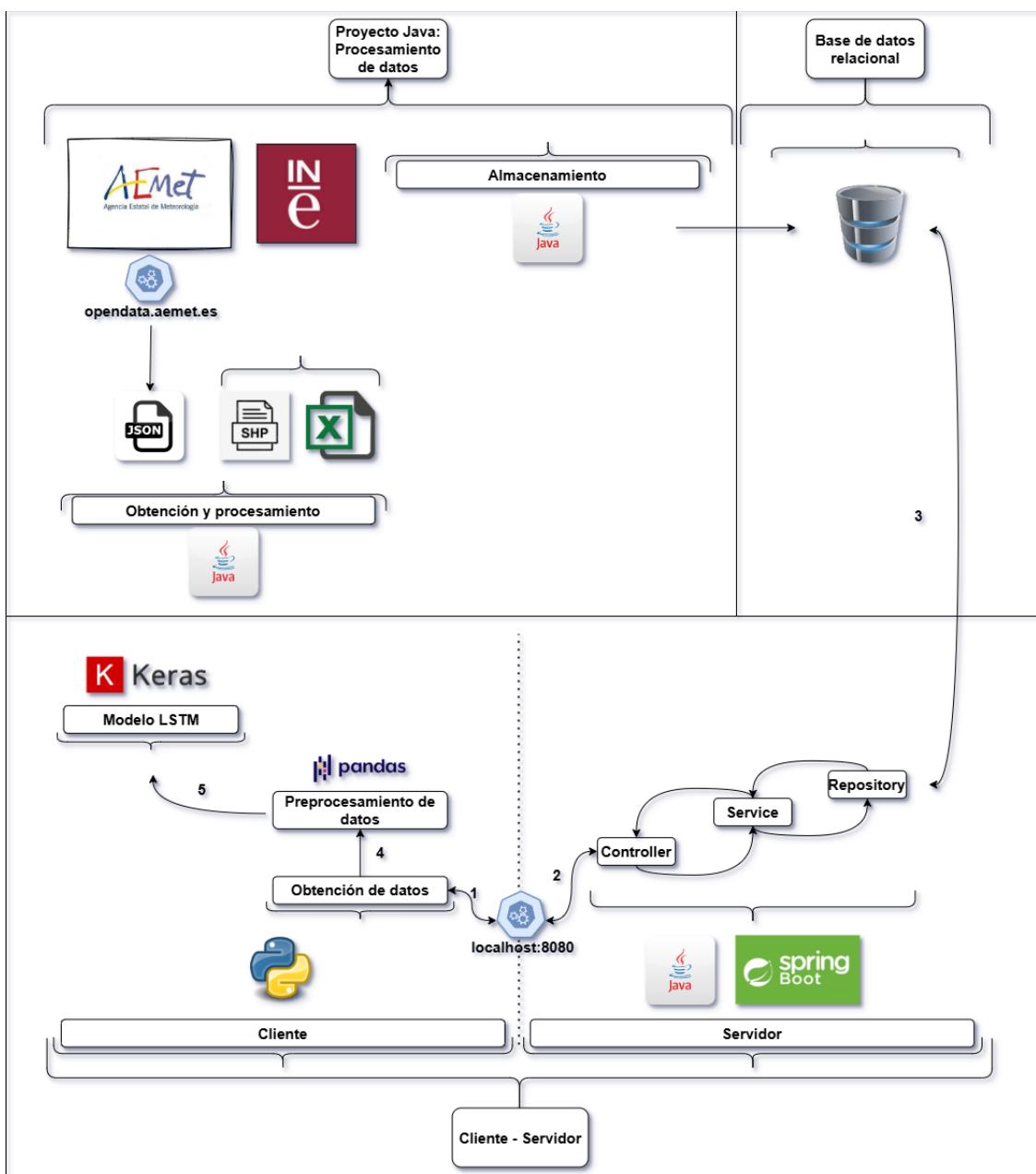


Figura 3.1 Diagrama de arquitectura general del sistema, dividido en 3 bloques: Procesamiento de datos, base de datos y cliente-servidor.

3.1 Obtención de los datos

Antes de iniciar el proceso de descarga e integración de los datos, fue necesario establecer una base conceptual sólida que permitiera unificar información proveniente de distintas fuentes. La premisa general de este Trabajo de Fin de Grado es evaluar si las variables meteorológicas y sociodemográficas — obtenidas de organismos oficiales como AEMET e INE — pueden contribuir de

forma significativa a la predicción de ventas de un comercio. Sin embargo, como se mencionó en el capítulo de Estado del Arte, el ecosistema de datos abiertos en España aún presenta una cierta fragmentación: distintas fuentes, formatos heterogéneos y niveles dispares de detalle.

Por tanto, antes de recopilar los datos, fue necesario plantearse: **¿qué elemento puede actuar como nexo común entre todas estas fuentes?** Es decir, se requería encontrar una unidad mínima territorial sobre la que poder construir toda la infraestructura de datos, tanto para el almacenamiento como para la consulta desde un servicio web.

En un primer momento se eligió el **municipio** como esa unidad de integración, ya que tanto el INE como AEMET ofrecen datos con un nivel suficientemente detallado a este nivel. De hecho, en el caso del INE, muchos de los indicadores sociodemográficos están incluso disponibles a niveles más específicos, como el **distrito** o la sección censal.

Una **sección censal** es la unidad territorial más pequeña utilizada por el Instituto Nacional de Estadística (INE) para la recopilación y difusión de datos estadísticos. Se trata de un área geográfica que agrupa un número reducido de viviendas y población, delimitada para facilitar el trabajo de los censos y encuestas. Cada sección pertenece de forma única a un distrito censal, y a su vez a un municipio, garantizando así una jerarquía clara en la organización del territorio. Esta granularidad permite analizar información con un elevado nivel de precisión territorial.**[60]**

Una vez definida esta idea de estructura jerárquica, se comenzó a planificar la forma de la futura base de datos, pieza clave del proyecto. En este punto, ya no bastaba con definir el municipio como “unidad”, sino que era imprescindible disponer de un identificador único para cada uno. Afortunadamente, tanto AEMET como INE utilizan una codificación común basada en concatenaciones numéricas. Por ejemplo, el identificador 28127 hace referencia al municipio de *Las Rozas de Madrid*, donde:

- 28 corresponde a la provincia de Madrid.
- 127 es el número del municipio dentro de esa provincia.

Esta codificación se repite también para provincias y comunidades autónomas, e incluso a nivel de sección censal, lo que facilita la relación entre variables procedentes de diferentes fuentes.

No obstante, esta lógica **no se aplica a las estaciones meteorológicas**. A diferencia de los municipios o secciones censales, las estaciones no están asociadas directamente a un identificador territorial. Lo único que sabemos de ellas es su ubicación mediante **coordenadas geográficas** (latitud y longitud). Esto plantea un problema si se quiere vincular una medición meteorológica con un municipio o sección censal concreta.

Además, el servicio REST que se plantea construir como parte del proyecto tiene como objetivo ofrecer una interfaz sencilla y flexible para consultar los datos integrados. En su diseño inicial, se contemplan **dos endpoints principales**:

- Un endpoint que reciba como parámetro un **código postal**, y devuelva la información meteorológica y sociodemográfica correspondiente.
- Un endpoint que acepte como entrada unas **coordenadas geográficas** (latitud y longitud), y devuelva los datos asociados a la ubicación proporcionada.

La idea es que el sistema sea capaz de resolver una consulta sobre el entorno (ya sea para alimentar un modelo de predicción o para ofrecer una visualización) sin necesidad de que el usuario conozca detalles administrativos como el municipio o la sección censal.

No obstante, esta decisión de diseño acarrea ciertos desafíos. Por ejemplo, **un mismo municipio puede tener múltiples códigos postales**. En el caso de Madrid, por ejemplo, existen 128 códigos postales distintos, lo que impide establecer una relación directa uno a uno entre municipio y código postal. Este problema se agrava cuando se requiere una consulta precisa. La solución pasa por descender al nivel de sección censal, ya que cada una de estas secciones pertenece de forma unívoca a un único código postal. De este modo, si se solicita información para un código postal, bastaría con agrupar las secciones censales correspondientes en la base de datos.

El inconveniente, sin embargo, es que no existe una fuente pública directa que relacione códigos postales con secciones censales, pese a que ambas unidades comparten jerarquía territorial dentro de un municipio.

Un obstáculo similar se presenta con las coordenadas geográficas. Aunque sepamos las coordenadas del usuario o de una estación meteorológica, no disponemos inicialmente de una fuente que indique a qué sección censal pertenece ese punto geográfico. Este tipo de asociación es imprescindible para conectar de manera coherente los datos meteorológicos (que dependen de la ubicación) con las variables sociodemográficas, que se almacenan estructuradas por secciones.

A raíz de estas dificultades, se identificaron tres problemas que debían resolverse para continuar con el desarrollo del sistema:

1. Encontrar un nexo común entre los datos meteorológicos y sociodemográficos.
2. Disponer de una fuente que relacione secciones censales con códigos postales.
3. Asignar coordenadas geográficas a cada sección censal y poder hacer consultas inversas (coordenadas → sección).

La solución a estos tres retos llegó con la incorporación de un recurso clave: el shapefile proporcionado por el INE.

Un shapefile es un formato digital de almacenamiento de información geográfica vectorial[61]. Contiene geometrías (puntos, líneas o polígonos) junto con

atributos descriptivos asociados. En este caso, cada polígono representa el contorno geográfico de una sección censal. Gracias a esta representación espacial, es posible:

1. Determinar a qué sección censal pertenece un par de coordenadas.
2. Obtener el centroide de una sección (su punto medio) para asociarla con un código postal.
3. Saber a qué municipio pertenece cada sección censal.

Esto permite vincular los datos meteorológicos, que inicialmente solo tienen coordenadas (en el caso de las estaciones), con unidades territoriales concretas que sí están presentes en los datos del INE.

Cabe señalar que antes de optar por esta solución se había considerado el uso de servicios como CartoCiudad, que permitía obtener información a partir de coordenadas (reverse geocoding). Sin embargo, CartoCiudad no ofrecía una solución completa para el cruce con secciones censales y códigos postales, por lo que finalmente se optó por trabajar directamente con el shapefile del INE.

3.1.1 AEMET

La Agencia Estatal de Meteorología (AEMET) es el organismo encargado de proporcionar datos meteorológicos en España. A través de su portal de datos abiertos¹⁰, AEMET permite el acceso a múltiples servicios mediante una API.

Para poder consumir esta API, es necesario registrarse como desarrollador en la propia web de AEMET, generando así una clave de acceso (API Key) que debe incluirse en todas las peticiones realizadas al servicio. Esta API Key actúa como identificador único del usuario y controla el número de peticiones por minuto, estableciendo un límite máximo de 150 peticiones por minuto.

El funcionamiento de la API de AEMET sigue una estructura basada en dos pasos:

1. **Primera petición:** Se accede a una URI principal con los parámetros necesarios (por ejemplo, un rango de fechas o un identificador de municipio). Esta petición devuelve un objeto JSON que contiene varios campos, entre ellos “estado”, “descripción”, “metadatos” y, lo más importante, una URL temporal contenida en el campo “datos”, desde donde se podrán descargar los datos reales.
2. **Segunda petición:** Se realiza una nueva petición HTTP al enlace proporcionado en “datos”, que apunta a un archivo JSON alojado temporalmente. Esta segunda petición es la que finalmente devuelve los datos meteorológicos en sí.

¹⁰ El portal se puede encontrar en la siguiente dirección : <https://opendata.aemet.es>

Response body

```
{  
  "descripcion": "exito",  
  "estado": 200,  
  "datos": "https://opendata.aemet.es/opendata/sh/2bd33091",  
  "metadatos": "https://opendata.aemet.es/opendata/sh/dfd88b22"  
}
```

Figura 3.2 Objeto JSON con la respuesta inicial del servidor de AEMET

Para gestionar esta lógica se implementó una clase Java llamada PrimeraRespuesta, que encapsula el JSON recibido en la primera petición. Su estructura incluye los atributos mencionados anteriormente y permite identificar con rapidez si la petición ha sido exitosa y cuál es la URL de descarga efectiva.

AEMET proporciona múltiples endpoints o URIs para acceder a los datos¹¹, entre los que destacan:

- **/api/prediccion/nacional/hoy:** Predicción meteorológica a nivel nacional en formato texto.
- **/api/maestro/municipios:** Información básica sobre todos los municipios, como ID, latitud, longitud y número de habitantes.
- **/api/prediccion/especifica/municipio/diaria/{municipio}:** Predicción diaria detallada para un municipio concreto.
- **/api/observacion/convencional/todas:** Observaciones meteorológicas en tiempo real para todas las estaciones.
- **/api/valores/climatologicos/diarios/datos/fechaini/{fechaIni}/fechaFin/{fechaFin}/todasestaciones:** Histórico diario de datos meteorológicos de todas las estaciones entre dos fechas.

Este abanico de URIs ofrece muchas posibilidades, pero también impone una estructura clara sobre cómo deben organizarse las peticiones y cómo deben tratarse las respuestas. Como veremos en los siguientes apartados, se han desarrollado dos líneas principales para explotar esta API: una centrada en las predicciones meteorológicas por municipios y otra en la recopilación de datos históricos diarios por estaciones.

¹¹ Estos endpoints se describen en un Swagger en la siguiente dirección:
<https://opendata.aemet.es/dist/index.html#/>

3.1.1.1 Municipios

En un primer momento, se planteó como objetivo obtener predicciones meteorológicas diarias a nivel municipal usando el endpoint:

`/api/prediccion/especifica/municipio/diaria/{id_municipio}`

Esta URI permite solicitar una predicción meteorológica específica para un municipio concreto, utilizando como parámetro el identificador de este (por ejemplo, como mencionábamos en la introducción del apartado **Obtención de los datos**, *Las Rozas de Madrid* tiene el ID 28127).

Esas predicciones cubren los siete días siguientes y se actualizan de forma continua. Incluyen datos como temperatura, viento, estado del cielo, probabilidad de lluvia y radiación ultravioleta máxima.

Para adaptarse a este endpoint, se diseñó una jerarquía de clases en Java que permitiera representar adecuadamente la estructura de las respuestas JSON. En este diseño se utilizó la librería Gson para convertir automáticamente las respuestas JSON en objetos Java. Por motivos de simplicidad, se optó por utilizar HttpURLConnection como método para consumir el servicio REST.

La organización se muestra claramente en el siguiente diagrama:

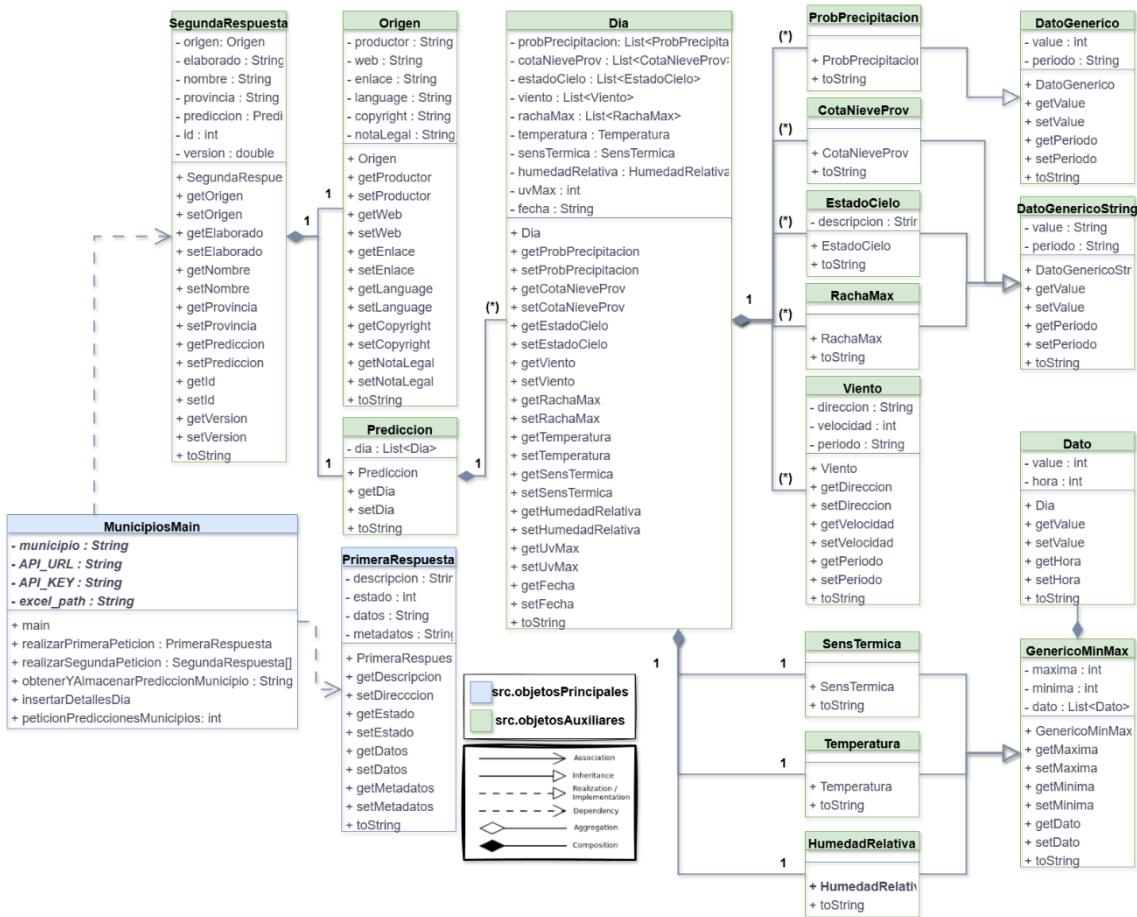


Figura 3.3 Diagrama UML de clases para predicción por municipio

Sin embargo, a medida que avanzó el desarrollo se identificaron dos problemas clave que llevaron a abandonar este enfoque en la práctica:

1. Las predicciones sólo abarcan una semana vista, lo que limita su utilidad para un análisis histórico o comparativo con datos de ventas pasadas.
2. Su uso efectivo requeriría que el servicio REST que se desea construir haga una llamada a AEMET en tiempo real cada vez que reciba una petición, lo cual introduce una complejidad adicional y dependencia temporal.

Por estas razones, aunque se haya desarrollado toda la lógica y estructura para obtener y procesar estas predicciones, finalmente se optó por no utilizarlas en la fase final del proyecto. Pese a esta decisión queda abierta la posibilidad de en un futuro integrarlos con los demás datos.

3.1.1.2 Estaciones

Otra de las opciones para obtener datos meteorológicos se basó en utilizar el endpoint:

/api/observacion/convencional/todas

Este endpoint proporciona observaciones en tiempo real (últimas 12 horas) de todas las estaciones meteorológicas “automáticas”. Un ejemplo completo de los campos disponibles puede encontrarse en el **Anexo 8.3.1**.

Para hacernos una idea del tipo de datos que se ofrece, estos son algunos de los más representativos:

- lon: Longitud de la estación meteorológica (grados).
- lat: Latitud de la estación meteorológica (grados)
- prec : Precipitación acumulada, medida por el pluviómetro, durante los 60 minutos anteriores a la hora indicada por el período de observación
- vmax : Velocidad máxima del viento, valor máximo del viento mantenido 3 segundos.

Para trabajar con este endpoint, se diseñó una estructura de clases Java específica, que permite mapear las observaciones recogidas en tiempo real a objetos reutilizables. Este diseño se refleja en la **Figura 3.4**:

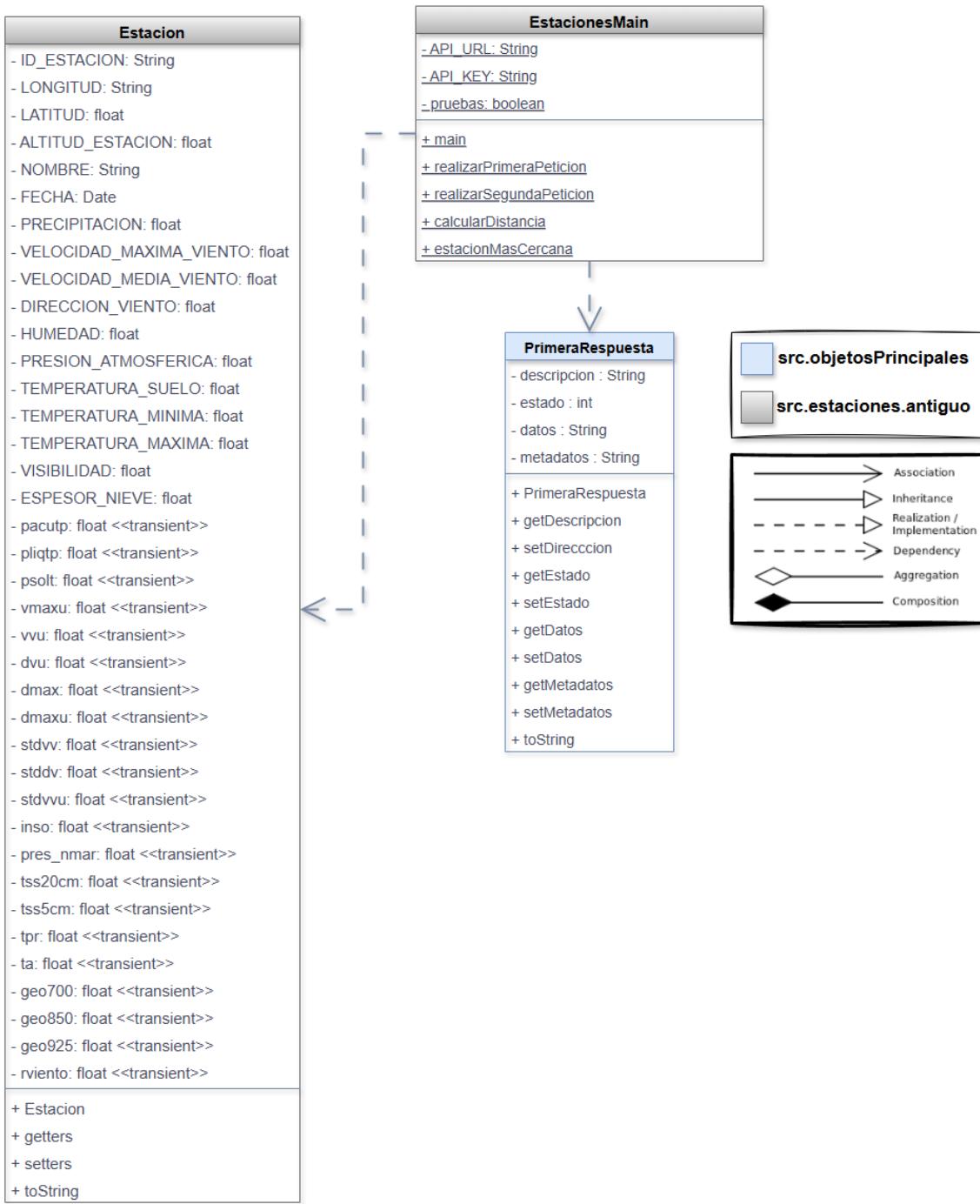


Figura 3.4 Diagrama UML del modelo de estaciones antiguas

Aunque inicialmente se consideró este endpoint como fuente principal de datos, se comprobó que no era viable construir un histórico a partir de él debido a la temporalidad del dato (observaciones de las últimas 12 horas, que se actualizan cada hora). Sin embargo, nos ha servido en parte para insertar en la BBDD los

campos: **latitud, longitud, altitud y nombre**. Esta información sirve para luego asociar los datos históricos descargados del nuevo endpoint.

Para crear un histórico completo, se adoptó el uso del endpoint:

```
/api/valores/climatologicos/diarios/datos/fechaini/{fechaIni}/fechafin/{fechaFin}/todasestaciones
```

Permite obtener datos meteorológicos diarios desde 2022¹² hasta la fecha actual. Entre otras variables¹³, contaremos con **temperatura media, precipitaciones, humedad relativa e insolación**, que más tarde nos serán de mucha utilidad.

La descarga se automatiza a través de la clase HistoricoEstaciones, que divide el rango de fechas total en bloques de 15 días (límite impuesto por AEMET), realiza las peticiones de forma secuencial, gestiona los errores y transforma los datos recibidos en objetos Medicion. Durante este proceso, se han tenido en cuenta distintas casuísticas que pueden aparecer en los datos, como valores nulos o strings como "Ip" o "Acum". Estos casos se gestionan de forma específica para asegurar que la información sea transformada de forma correcta.

El diseño de clases que permite esta lógica se resume en el siguiente diagrama UML:

¹² El endpoint permite consultar datos anteriores. No obstante, para este Trabajo, la fecha inicial es 2022 puesto que los datos que tenemos de ventas de un negocio se inician en dicha fecha.

¹³ Se pueden consultar todas las variables en el **Anexo 8.3.2**

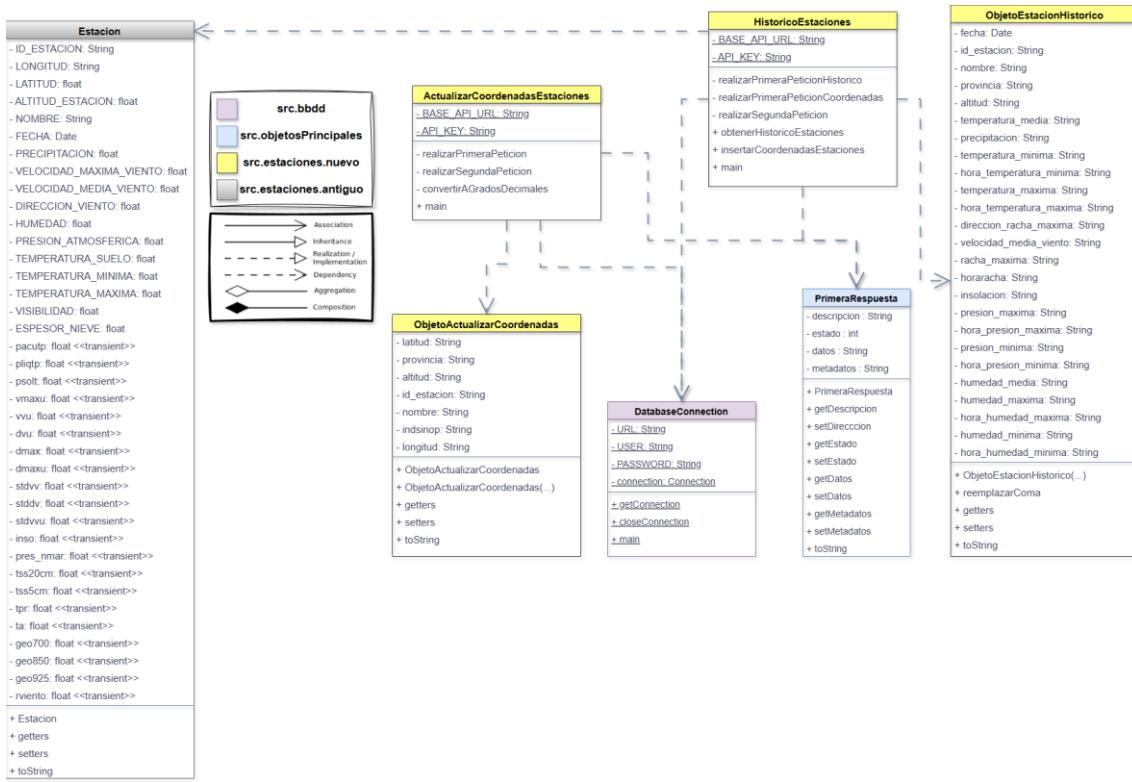


Figura 3.5 Diagrama UML para la descarga del histórico diario

A continuación se muestra un **fragmento del código** encargado de realizar las dos peticiones necesarias para **obtener el histórico** desde la API de AEMET. En él, se emplea la clase PrimeraRespuesta para deserializar la respuesta inicial y extraer la URL con los datos reales. Posteriormente, estos datos se vuelven a deserializar como un array de objetos ObjetoEstacionHistorico:

```

PrimeraRespuesta aux = null;
ObjetoEstacionHistorico[] respuesta = null;

GsonBuilder gsonBuilder = new GsonBuilder();
gsonBuilder.setDateFormat("dd/MM/yyyy hh:mm a");
Gson gson = gsonBuilder.create();

String response = realizarPrimeraPeticionHistorico(fechaIniStr,
fechaFinStr);

String urlDatosAemet = null;

```

```

if (response != null) {
    aux = gson.fromJson(response, PrimeraRespuesta.class);
    urlDatosAemet = aux.getDatos();
}
String res = realizarSegundaPeticion(urlDatosAemet);

respuesta = gson.fromJson(res, ObjetoEstacionHistorico[].class);

```

Tabla 3.1 Doble petición y deserialización con Gson

3.1.2 INE

El Instituto Nacional de Estadística (INE) es una fuente extensa y diversa de datos sociodemográficos y económicos en España. Para este proyecto, su papel resulta fundamental, ya que nos proporciona datos estructurados a diferentes niveles territoriales que complementan la información meteorológica de AEMET. Sin embargo, acceder y trabajar con estos datos no es tan trivial como pudiera parecer en un principio.

Existen literalmente miles de variables publicadas en el INE, cubriendo áreas como salud, trabajo, educación, entorno, medioambiente, uso de tecnologías, calidad de vida y más. Sin embargo, no todas estas variables están disponibles con la misma granularidad territorial ni con la misma frecuencia temporal. De hecho, muchas de ellas solo se encuentran a nivel nacional o autonómico, lo que limita su utilidad para un análisis a nivel local. Para poder hacer uso riguroso de estos datos en un modelo predictivo de ventas, necesitamos información con un nivel de desagregación mucho más fino.

Tal y como se detalló en el apartado **Obtención de los datos**, la unidad de análisis seleccionada es la **sección censal**, ya que se trata de la división territorial más pequeña utilizada por el INE para difundir datos estadísticos.

3.1.2.1 Datos

Dado el volumen y la diversidad de los datos que publica el INE resulta necesario hacer una selección. A modo de contexto, a continuación se presentan algunos ejemplos interesantes de datos disponibles, organizados por temática. Reflejan el enorme potencial de análisis que ofrece el INE, pero también muestran la dispersión y heterogeneidad de los datos. Explorar todos estos conjuntos de datos requeriría por sí solo un trabajo de investigación completo:

- **Salud:**
 - Tasa bruta de natalidad por municipio (aunque no todos los municipios están representados), desde 2014 hasta 2023.
 - Esperanza de vida a nivel nacional, desde 1975 hasta 2023.
 - Proporción de personas mayores por provincia, desde 1975 hasta 2024.
 - Tasa de natalidad por nacionalidad, por comunidad autónoma, desde 2002 hasta 2023.
- **Trabajo:**
 - Población activa, ocupada e inactiva por provincia y comunidad, desde 2007 hasta 2025. Estos datos están disponibles por trimestre.
 - Datos de empleo por edad, nivel educativo, sexo o nacionalidad, a nivel nacional, desde 2006 hasta 2023.
- **Educación:**
 - Niveles de estudios alcanzados (analfabetismo, estudios primarios completos o incompletos, educación superior, etc.), por comunidad autónoma, desde 2014 hasta 2025, con frecuencia trimestral.
- **Medioambiente y entorno:**
 - Encuestas de satisfacción o bienestar, generalmente disponibles solo para años puntuales.
- **Tecnología (TIC):**
 - Porcentaje de hogares con teléfono móvil, fijo, ordenador o acceso a Internet, disponibles por comunidad autónoma y a nivel nacional, desde 2006 hasta 2024.
 - Porcentaje de menores de 15 años con acceso a tecnologías digitales, también disponible desde 2006.
- **Encuestas sociales:**
 - Porcentaje de población en riesgo de pobreza por comunidad, desde 2008 hasta 2024.

Las variables finalmente seleccionadas provienen de tres bloques temáticos ofrecidos por el INE[62]:

- **Indicadores de renta** (media y mediana)
- **Distribución por fuente de ingresos**
- **Indicadores demográficos**

Dentro de estas categorías, se obtendrán variables como:

- Renta neta media por persona
- Renta bruta media por hogar

- Porcentaje de ingresos por salario
- Porcentaje de ingresos por pensión
- Porcentaje de ingresos por prestaciones de desempleo u otras ayudas
- Población total
- Porcentaje de población española
- Tamaño medio del hogar
- Edad media de la población

En cuanto al acceso y descarga de los datos, el INE ofrece una interfaz web que permite seleccionar provincia, unidad territorial (municipio, distrito o sección censal), año, y variables de interés. Estos datos pueden descargarse en distintos formatos: .xlsx, .csv o .json. Aunque los formatos .csv y .json podrían parecer inicialmente más manejables para un tratamiento automatizado, se descartaron por su complejidad estructural y menor claridad en la presentación de la información. En cambio, el formato .xlsx fue el elegido por su facilidad de lectura, tanto para el desarrollo como para su posible revisión por parte del tribunal.

Resultados por municipios, distritos y secciones censales						
Madrid						
Indicadores de renta media y mediana						
Unidades: €						
	Renta neta media por hogar 2021	Renta neta media por hogar 2021	Media de la renta por hogar 2021	Mediana de la renta 2021	Renta bruta media por hogar 2021	Renta bruta media por hogar 2021
28001 Acebeda, La	15.245	27.228			18.151	32.418
2800101 Acebeda, La distrito 01	15.245	27.228			18.151	32.418
2800101001 Acebeda, La sección 01001	15.245	27.228			18.151	32.418
28002 Ajalvir	14.285	40.309	22.021	19.250	17.510	49.410
2800201 Ajalvir distrito 01	14.285	40.309	22.021	19.250	17.510	49.410
2800201001 Ajalvir sección 01001	15.276	43.101	23.461	20.650	18.812	53.079
2800201002 Ajalvir sección 01002	13.480	38.040	20.848	18.550	16.453	46.429
28003 Alameda del Valle	17.962	31.144	23.319	20.650	22.296	38.659
2800301 Alameda del Valle distrito 01	17.962	31.144	23.319	20.650	22.296	38.659
2800301001 Alameda del Valle sección 0100	17.962	31.144	23.319	20.650	22.296	38.659
28004 Álamo, El	11.097	32.321	17.271	15.750	13.081	38.098
2800401 Álamo, El distrito 01	11.097	32.321	17.271	15.750	13.081	38.098
2800401001 Álamo, El sección 01001	10.494	30.566	16.419	15.050	12.215	35.577
2800401002 Álamo, El sección 01002	11.814	32.706	18.078	16.450	14.033	38.849
2800401003 Álamo, El sección 01003	10.207	31.420	16.167	15.050	11.972	36.855
2800401004 Álamo, El sección 01004	11.883	36.072	18.627	17.150	14.136	42.912
28005 Alcalá de Henares	13.809	37.487	20.924	18.550	16.741	45.446

Figura 3.6 Fragmento de un Excel con datos del INE

Para cada provincia se descargan tres ficheros .xlsx, correspondientes a cada uno de los bloques mencionados anteriormente. Considerando las 50 provincias y las dos ciudades autónomas (Ceuta y Melilla), esto supone un total de 156 ficheros Excel y **millones de filas de información censal**. Cabe destacar que, aunque la mayoría de las secciones contienen datos completos, existe un porcentaje pequeño pero significativo de registros con valores nulos, que se tendrán en cuenta en la fase de Obtención.

Además de los ficheros estadísticos en formato Excel, el INE también proporciona cartografía censal en formato digital[63], concretamente mediante un archivo **shapefile**. Este fichero recoge la geometría de todas las secciones censales del territorio español, representadas como polígonos geográficos. Cada sección incluye una serie de atributos clave, como su identificador único (CUSEC), el código del municipio al que pertenece (CUMUN), el distrito censal, la provincia y otros datos administrativos relevantes. Esta representación espacial permite vincular eficazmente información estadística con la localización geográfica correspondiente.

En el contexto de este proyecto, el shapefile cumple un papel fundamental. Permite **traducir coordenadas geográficas a identificadores de sección censal** y viceversa, facilitando tanto la integración como la consulta de los datos. Además, ofrece acceso a información adicional sobre cada sección censal, como el **código postal** al que pertenece. Con estas funcionalidades, el shapefile actúa como nexo entre los distintos conjuntos de datos utilizados, y **resuelve** de forma directa los tres **problemas clave planteados** anteriormente: la asociación entre coordenadas, secciones censales y códigos postales.

3.1.2.2 Obtención

El proceso de obtención de los datos del INE se ha estructurado cuidadosamente para poder tratar tanto los ficheros estadísticos .xlsx como la cartografía censal en formato shapefile. Cada tipo de fuente se ha abordado con tecnologías distintas: por un lado, **Apache POI** para el tratamiento de los Excel; por otro, la librería **GeoTools** para trabajar con información geoespacial. Este enfoque ha permitido construir un sistema robusto y modular que facilita la automatización de todo el proceso.

La lectura de los ficheros estadísticos se ha dividido por provincias y por tipo de información. Cada conjunto de datos (indicadores de renta, distribución por fuente de ingresos, e indicadores demográficos) se procesa mediante métodos específicos:

- procesarIndicadoresRentaMedia
- procesarDistribucionPorFuenteIngresos
- procesarIndicadoresDemograficos

Esta organización ha permitido separar la lógica de carga y adaptación de los datos según su naturaleza, favoreciendo la claridad del código y su mantenimiento a largo plazo.

El método “principal” de la clase encargada de esta parte del sistema actúa como núcleo integrador de todo el flujo de trabajo. En primer lugar, recorre automáticamente todos los subdirectorios correspondientes a las provincias utilizando utilidades propias como UtilsDirectorios, y lee cada archivo Excel con ayuda de la clase LeerExcel. A partir del contenido de cada hoja, identifica el tipo de información contenida y llama al método correspondiente para almacenar los datos en la base de datos. Una vez completada esta fase, se lanza el proceso para asociar cada sección censal con su código postal. Esta etapa,

dividida en dos pasos aprovecha las capacidades del shapefile para extraer esta información geográfica adicional.

- obtenerCPostalesSeccionesCensalesTxt
- almacenarCPostalesSeccionesCensalesBBDD

En cuanto al tratamiento del shapefile, se ha desarrollado una clase específica (ShapefileUtils) que permite resolver varias de las necesidades planteadas en la fase inicial del proyecto. Entre otras funcionalidades, permite identificar a qué sección censal pertenece un punto dado en el mapa a partir de sus coordenadas geográficas, calcular el centroide de una sección o municipio concreto, y consultar atributos como el código postal, la provincia, el identificador de sección censal y otros campos asociados. Como base para el desarrollo de estas funcionalidades y su implementación técnica, se ha tomado como referencia la guía oficial de GeoTools[64].

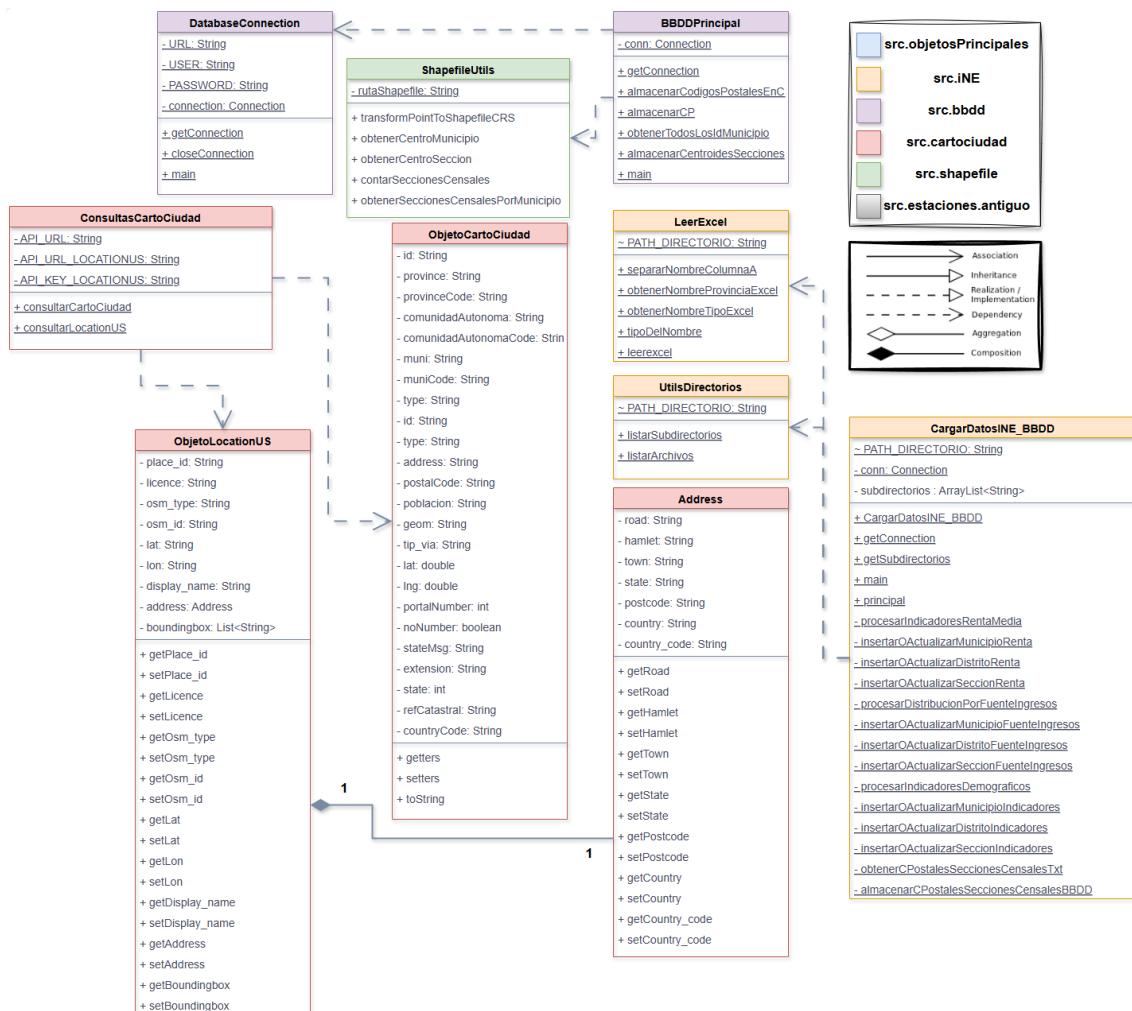


Figura 3.7 Diagrama UML de las clases de obtención y carga de datos del INE

Cabe señalar que antes de llegar a esta solución se exploraron alternativas. Inicialmente se empleó **CartoCiudad** para obtener el código postal a partir de coordenadas. Sin embargo, se observó que una parte considerable de las consultas no devolvía resultados, por lo que se optó por descartarla. También se realizó una prueba con la API extranjera **LocationUS**, pero resultó igualmente inviable. Estos intentos fallidos forman parte del camino seguido hasta la solución definitiva y se incluyen en este Trabajo para reflejar el carácter **iterativo e incremental** del desarrollo.

3.2 BBDD

La base de datos actúa como punto central de integración los otros dos bloques principales de este Trabajo de Fin de Grado. Por un lado, almacena los datos recogidos por el proyecto desarrollado en Java, encargado de obtener y estructurar la información meteorológica y sociodemográfica. Por otro, sirve como fuente de consulta para el servicio REST, que se abordará en el siguiente apartado.

El modelo de datos sigue una estructura relacional, implementada sobre **MySQL**, y está dividido en tres grandes bloques: datos meteorológicos por estación, predicciones meteorológicas por municipio y datos sociodemográficos por municipio, distrito y sección censal. El diseño se ha plasmado gráficamente en un **diagrama entidad-relación**, hecho con la herramienta Draw.io. Recoge todas las entidades, sus claves primarias y foráneas, así como los atributos y las relaciones.

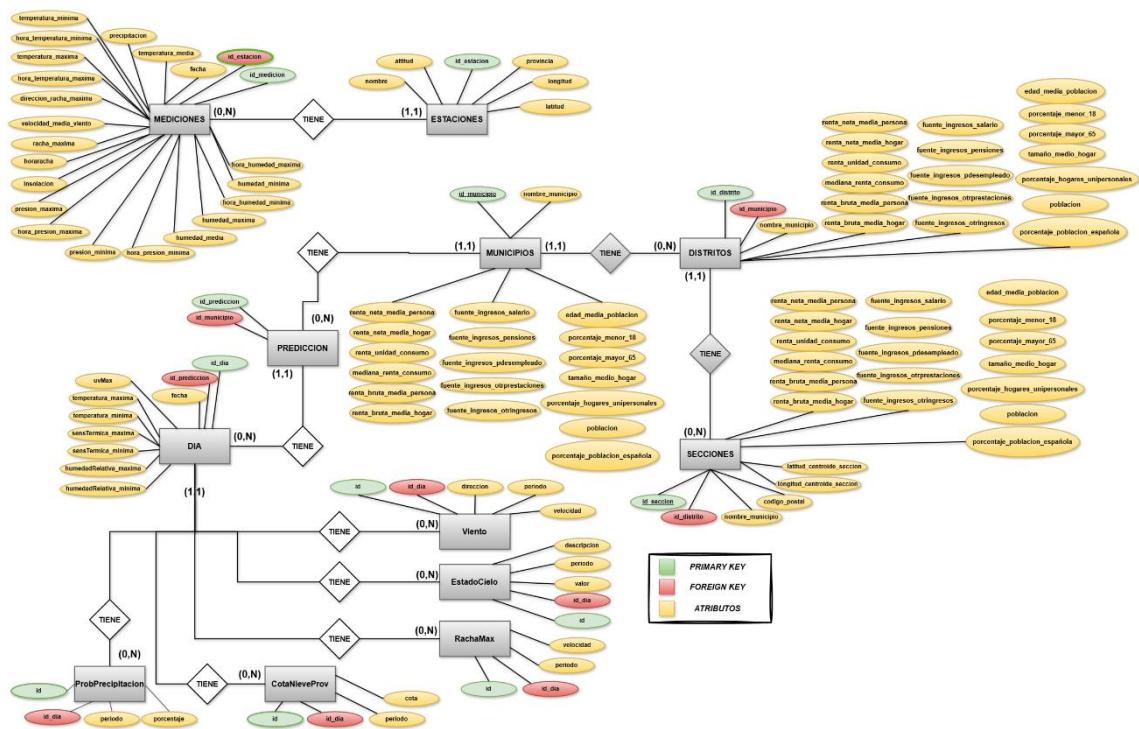


Figura 3.8 Diagrama entidad - relación

Por un lado, los datos históricos de estaciones meteorológicas, como se ha explicado en el apartado **¡Error! No se encuentra el origen de la referencia.**, se obtienen mediante peticiones a la API de AEMET y se almacenan en las tablas Estaciones y Mediciones, que están relacionadas mediante una clave primaria compuesta formada por ***id_estacion*** y ***fecha***.

A continuación se muestra el script SQL utilizado para crear la tabla Mediciones:

```
CREATE TABLE Mediciones (
    id_estacion VARCHAR(10) NOT NULL,
    fecha DATE NOT NULL,
    temperatura_media FLOAT,
    precipitacion FLOAT,
    temperatura_minima FLOAT,
    hora_temperatura_minima VARCHAR(10),
    temperatura_maxima FLOAT,
    hora_temperatura_maxima VARCHAR(10),
    direccion_racha_maxima FLOAT,
    velocidad_media_viento FLOAT,
```

```

racha_maxima FLOAT,
horaracha VARCHAR(10),
insolacion FLOAT,
presion_maxima FLOAT,
hora_presion_maxima VARCHAR(10),
presion_minima FLOAT,
hora_presion_minima VARCHAR(10),
humedad_media FLOAT,
humedad_maxima FLOAT,
hora_humedad_maxima VARCHAR(10),
humedad_minima FLOAT,
hora_humedad_minima VARCHAR(10),
PRIMARY KEY(id_estacion, fecha),
FOREIGN KEY (id_estacion) REFERENCES Estaciones(id_estacion) ON DELETE CASCADE
);

```

Tabla 3.2 Código SQL de creación de la tabla Mediciones

En paralelo, se desarrolló también una estructura para almacenar predicciones meteorológicas a nivel municipal. Esta parte incluye las tablas Prediccion y Dia, además de tablas auxiliares como ProbPrecipitacion, CotaNieveProv, EstadoCielo, Viento y RachaMax. Sin embargo, como se ha explicado en el apartado 3.1.1.1, esta parte carece de utilidad en el proyecto final y se mantiene nada más para dar peso al argumento de que **ha sido un desarrollo iterativo e incremental** (también podría ser utilizado en un futuro, de continuarse el proyecto).

En cuanto a los datos del INE, se almacenan en las tablas Municipios, Distritos y Secciones. Estas últimas incluyen no solo las variables sociodemográficas, sino también los atributos geográficos necesarios, como la latitud y longitud del centroide de cada sección censal, así como el código postal asociado. Este diseño permite realizar consultas precisas a partir de diferentes niveles territoriales o filtros como código postal, comunidad autónoma o coordenadas.

Durante el desarrollo del proyecto en Java se han creado varias clases que se encargan de insertar los datos en la base de datos, cada una adaptada a una fuente distinta. Algunas de las principales son:

- CargarDatosINE_BBDD
- HistoricoEstaciones
- BBDDPrincipal
- ActualizarCoordenadasEstaciones

Todas ellas utilizan sentencias SQL a través de PreparedStatement, haciendo uso de métodos como executeUpdate(), addBatch() y executeBatch(). Además, se

han incluido controles para evitar duplicidades o gestionar correctamente los valores nulos.

La clase DatabaseConnection contiene las credenciales de la BBDD y permite abrir o cerrar la conexión, simplificando la gestión del acceso en todo el proyecto.

A continuación se ofrece un ejemplo de un código de inserción, perteneciente a la clase ActualizarCoordenadasEstaciones:

```
String res = realizarSegundaPeticion(urlDatosEstaciones);

        respuesta = gson.fromJson(res, ObjetoActualizarCoordenadas[].class);

        String updateQuery = "UPDATE estaciones SET latitud = ?, longitud = ? WHERE
id_estacion = ? AND latitud IS NULL";

        try (PreparedStatement stmt = conn.prepareStatement(updateQuery)) {
            System.out.println("Actualizando Estaciones...");
            for (ObjetoActualizarCoordenadas estacion : respuesta) {
                String latitudOriginal = estacion.getLatitud();
                String longitudOriginal = estacion.getLongitud();

                double latitudDecimal = convertirAGradosDecimales(latitudOriginal);
                double longitudDecimal = convertirAGradosDecimales(longitudOriginal);

                estacion.setLatitud(latitudDecimal + "");
                estacion.setLongitud(longitudDecimal + "");

                stmt.setDouble(1, latitudDecimal);
                stmt.setDouble(2, longitudDecimal);
                stmt.setString(3, estacion.getIdEstacion());

                int rowsAffected = stmt.executeUpdate();
                if (rowsAffected > 0) {
                    System.out.println("Actualizada estación con indicativo: " +
estacion.getIdEstacion());
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

Tabla 3.3 Código inserción en la BBDD

Por último quiero mencionar que se han realizado múltiples consultas de prueba. Por ejemplo, se han ejecutado consultas de agregación sobre la tabla Mediciones para calcular promedios diarios y búsquedas específicas sobre la tabla Secciones para encontrar aquellas que carecen de código postal o latitud.

```

SELECT
    fecha,
    AVG(temperatura_media) AS temperatura_media,
    AVG(precipitacion) AS precipitacion,
    AVG(temperatura_minima) AS temperatura_minima,
    AVG(temperatura_maxima) AS temperatura_maxima,
    AVG(direccion_racha_maxima) AS direccion_racha_maxima,
    AVG(velocidad_media_viento) AS velocidad_media_viento,
    AVG(racha_maxima) AS racha_maxima,
    AVG(insolacion) AS insolacion,
    AVG(presion_maxima) AS presion_maxima,
    AVG(presion_minima) AS presion_minima,
    AVG(humedad_media) AS humedad_media,
    AVG(humedad_maxima) AS humedad_maxima,
    AVG(humedad_minima) AS humedad_minima
FROM Mediciones
WHERE id_estacion IN ("3194Y", "3343Y") AND fecha < '2022-01-15'
GROUP BY fecha;

select *
from estaciones est
inner join mediciones med
on est.id_estacion = med.id_estacion
where est.id_estacion = "1347T" AND fecha > "2024-11-15";
select * from secciones where latitud_centroide_seccion IS NULL;

```

Tabla 3.4 Ejemplos de querys SQL para testing

3.3 Servicio Web – Servidor

Antes de empezar a hablar sobre la estructura del servicio, sus dependencias, clases, diagramas y demás información técnica, me parece conveniente hacer una introducción sobre cómo ha ido evolucionando el desarrollo de esta parte del proyecto.

La idea originaria era crear un servicio utilizando herramientas vistas durante la carrera. Con esto me refiero a JAX-RS, definición de endpoints mediante anotaciones, empleo de Tomcat para el despliegue del servidor, y conexión con una base de datos utilizando InitialContext, PreparedStatement, etc. No obstante, varias reuniones después se establece la premisa de que JAX-RS no

es la opción a seguir. Se plantea en su lugar utilizar Spring, un framework mucho más potente y moderno. Su funcionamiento es similar al de JAX-RS (ambos utilizan anotaciones, verbos HTTP, y permiten la serialización/deserialización de objetos), pero con muchas más posibilidades y mejor soporte.

Esta decisión resultó positiva porque el cambio no suponía una ruptura total: ya se conocía la lógica general de construcción de un servicio REST. A partir de ahí, se empezó a estudiar todo el manual de Spring Boot a través de su página web para entender cómo se organiza un proyecto, qué partes lo componen, y qué ventajas ofrece.

Con todo esto se procede al desarrollo del servicio. El primer paso fue utilizar **Spring Boot Initializr**, una funcionalidad que ofrece Spring para generar el esqueleto del proyecto. A través de una interfaz web se pueden definir parámetros como el tipo de proyecto, el lenguaje y versión, la versión de Spring Boot y las dependencias necesarias, así como el nombre del grupo o del paquete principal. Para este proceso de aprendizaje, también se consultaron recursos complementarios como Baeldung, GeeksforGeeks, documentación oficial de Spring Data JPA, tutoriales disponibles en la plataforma Spring y otros artículos especializados publicados en plataformas como DEV o GitHub por la comunidad de Spring. [65][66][67][68][69][70][71]

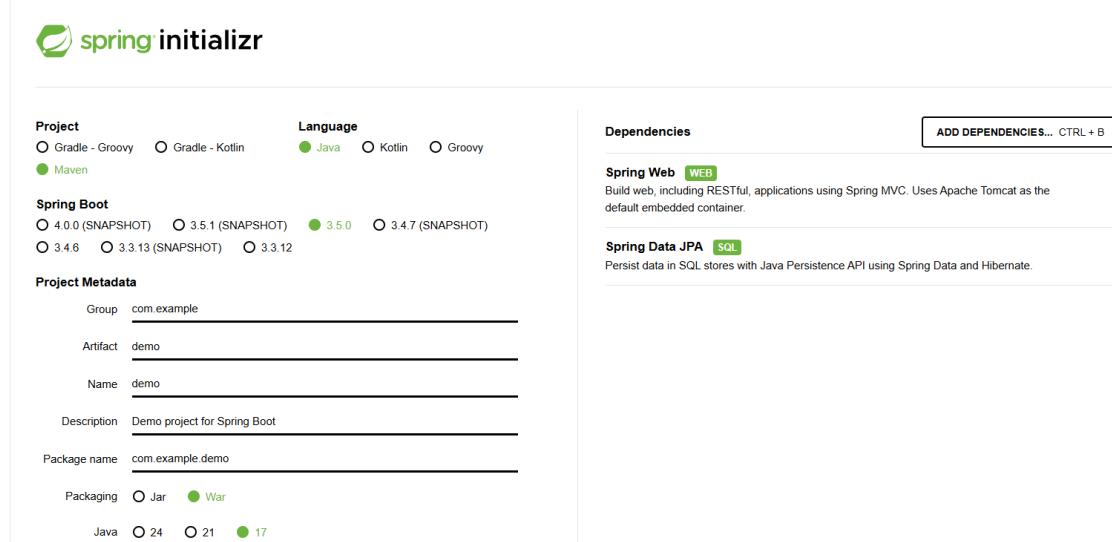


Figura 3.9 Pantalla principal Spring Initializr

Como podemos observar, se permite introducir variables como el tipo de proyecto, el lenguaje y la versión que se van a utilizar, la propia versión de Spring así como diversos datos adicionales como el nombre del grupo o del paquete principal o las dependencias que quieras utilizar teniendo en cuenta el tipo de proyecto a desarrollar.

Una vez desarrolladas ciertas funcionalidades básicas, se detectaron varios problemas. No se seguían buenas prácticas recomendadas por Spring. Únicamente se contaba con una clase `ServiceApplication` para iniciar el servidor, un Controller que manejaba todas las peticiones, algunos objetos

model, y una clase utils a medio desarrollar. No se había separado la lógica en capas Service ni Repository, ni se estaban utilizando DTOs. La conexión a base de datos se realizaba directamente desde el Controller, y este también se encargaba de todo el procesamiento. Esta estructura dificultaba el mantenimiento y complicaba mucho la depuración.

Tampoco existía una herramienta visual que permitiera documentar los endpoints, mostrar qué parámetros acepta cada uno, qué errores pueden producirse o qué formato de respuesta se espera. Además, no estaban bien definidos los mensajes de error personalizados.

Con todo esto en mente, se decidió reestructurar el servicio por completo. Los objetivos eran claros: resolver los problemas antes mencionados y construir una base de código clara, escalable y robusta. Además de este rediseño, fue necesario ampliar el servicio, pasando de dos endpoints iniciales a tres:

- /codigopostal
- /coordenadas
- /comunidad

Esta ampliación se produjo en paralelo al desarrollo del modelo LSTM final, cuyos resultados se verán reflejados en el **Objetivo 4**.

Dada esta pequeña introducción, pasamos a describir el producto final desarrollado. En los siguientes apartados se detallará partes como la estructura general del servicio, el funcionamiento de los distintos endpoints, las tecnologías y dependencias utilizadas, o el diseño del código siguiendo el patrón controlador-servicio-repositorio.

3.3.1 Estructura del proyecto. Organización en paquetes

La estructura principal del servicio se concentra en cuatro paquetes principales. El primero y más importante es el **controller**, que se encarga de recibir las peticiones HTTP, interpretar los parámetros y devolver una respuesta al cliente. Puede incluir cálculos menores, pero la lógica central del servicio no se encuentra en esta parte. Se puede identificar fácilmente con la anotación @RestController. En este proyecto se ha definido un único controller, que agrupa los tres endpoints disponibles, explicados con más detalle en el apartado **Endpoints desarrollados**.

Por otro lado, tenemos el paquete **service**. Este paquete agrupa toda la lógica de negocio. Se comunica con todas las demás partes del servicio, excepto con la base de datos directamente. Las clases en este paquete se identifican por usar las anotaciones @Service y @Component. Esta última no define una clase de tipo servicio como tal, pero sirve para marcar clases auxiliares que contienen funciones reutilizables. Aquí es también donde se utilizan los métodos definidos en las clases @Repository, que veremos a continuación.

En este proyecto se han definido tres clases de servicio:

- EstacionService
- MedicionService

- SeccionService

El paquete **repository** es el encargado de interactuar directamente con la base de datos. Las clases aquí están anotadas con `@Repository`. Aunque no se definan métodos explícitos, siempre estarán disponibles métodos CRUD básicos gracias a Spring JPA. Esto es lo que se conoce como *Query Method Naming*, y permite que Spring genere automáticamente las consultas en función del nombre del método. Algunos ejemplos de métodos disponibles son:

- `findAll()`
- `findById(String id)`
- `save(T objetoEntidad)`
- `deleteById(String id)`

Esta estrategia detecta palabras clave[53] como DISTINCT, BETWEEN, LIKE directamente en el nombre del método, sin necesidad de definir consultas SQL manuales. En los casos donde fue necesario definir consultas más complejas, se usó la anotación `@Query`.

Las clases repository implementadas en este servicio son:

- EstacionRepository
- MedicionRepository
- SeccionRepository

Por último, vamos a hablar de los **modelos** y los **DTOs**. Los modelos son clases que definen las entidades de la base de datos. Estas clases son utilizadas por JPA para mapear automáticamente los resultados de una consulta SQL a objetos Java. Por otro lado, los DTOs (Data Transfer Objects) son clases diseñadas para definir el tipo de objeto que se devuelve como respuesta en una petición HTTP.

Los modelos definidos en este servicio son:

- Estacion
- Medicion
- MedicionId
- Seccion

Y los DTOs utilizados son:

- MedicionDTO
- CoordenadasGeograficas
- ObjetoRespuesta

Para concluir la explicación de la estructura, se aporta un diagrama UML con todos los atributos, métodos y relaciones entre las clases:

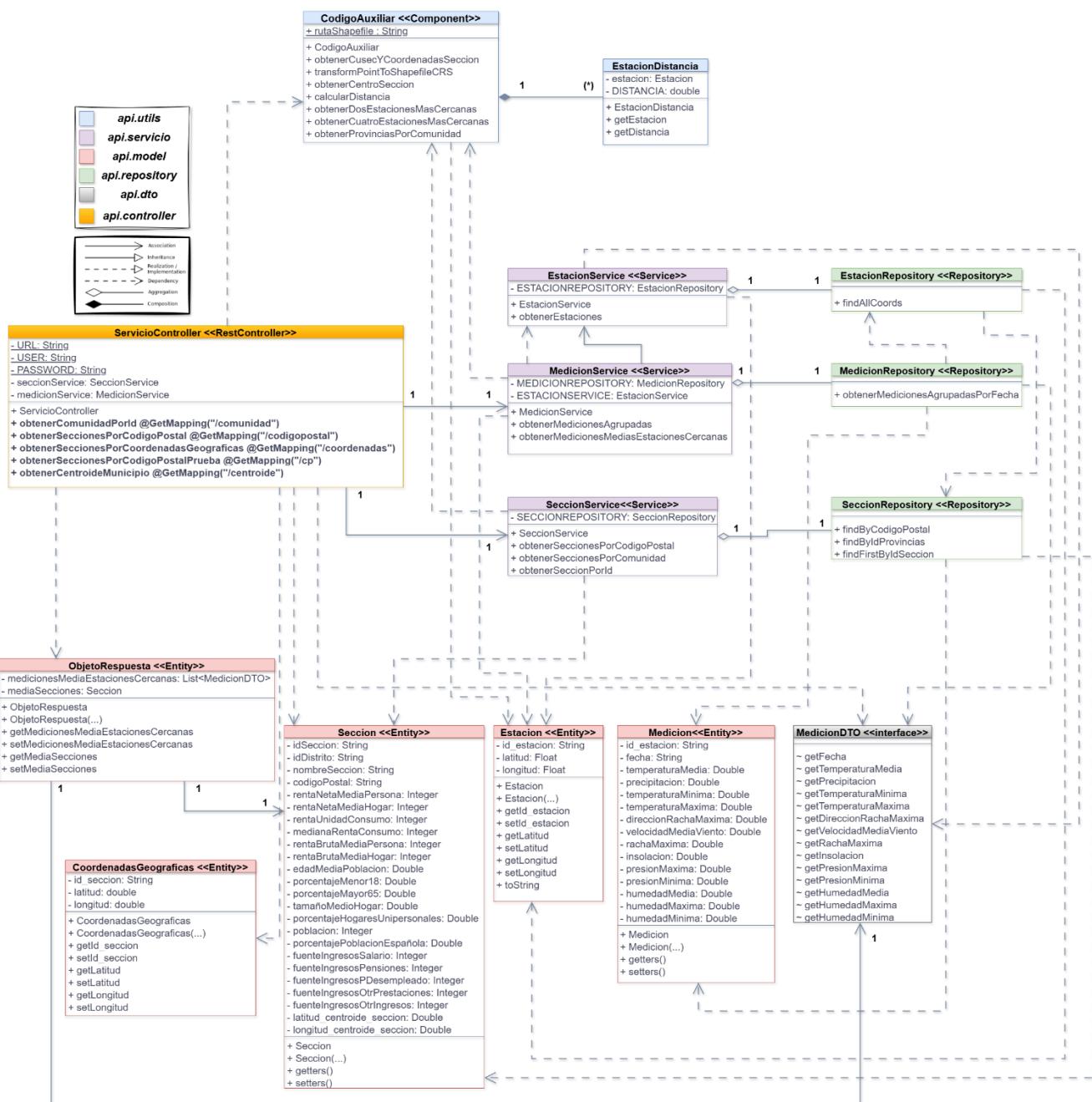


Figura 3.10 Diagrama UML del servicio

3.3.2 Endpoints desarrollados

En este apartado se detallarán los endpoints ofrecidos por el servicio. Su objetivo, como se ha mencionado en la **Introducción**, es la de dar acceso a los datos agrupados de las fuentes AEMET e INE de tal forma que un cliente pueda

consultarlos introduciendo datos sencillos como el código postal, coordenadas geográficas o el id de una comunidad autónoma. Los endpoints son los siguientes: **/códigopostal**, **/coordenadas** y **/comunidad**.

Antes de entrar en detalle, es muy importante establecer, que los datos meteorológicos obtenidos de las estaciones de la AEMET están agrupados **DIARIAMENTE**, para todos los endpoints. Los datos del INE en cambio son **ANUALES**.

3.3.2.1 Código postal

El funcionamiento básico de este endpoint consiste en recibir un código postal español válido como parámetro de entrada y devolver dos tipos de información:

- Los datos sociodemográficos medios de todas las secciones censales asociadas a dicho código postal.
- Los datos meteorológicos medios de las dos estaciones meteorológicas más cercanas al centroide de esas secciones censales, siempre y cuando estén ubicadas a menos de 30 kilómetros de distancia.

Para apoyar a este proceso, se comprueba si el código postal introducido tiene únicamente dígitos y formato postal (cinco dígitos)

A continuación se presenta una tabla con las posibles respuestas del servidor junto con una explicación más en detalle sobre el funcionamiento interno:

Código HTTP	Estado	Contenido de la respuesta
200	OK	ObjetoRespuesta (para ver el contenido, dirigirse al Anexo 8.4)
204	No Content	No se encontraron secciones censales para dicho código postal.
400	Bad Request	El código postal tiene un formato inválido o no existen estaciones cercanas al mismo.
		El código postal debe tener exactamente 5 caracteres.
		Falta el/los parámetro/s requerido.
404	Not Found	No se encuentra el endpoint. O se ha introducido un método no permitido.
500	Internal Server Error	Error interno del servidor.

Tabla 3.5 Tabla con las respuestas HTTP para código postal

El proceso comienza con obtener todas las secciones censales asociadas al código postal introducido. Para ello se emplean tanto “*SeccionService*” como “*SeccionRepository*”.

A continuación, se calcula el centroide de todas las secciones obtenidas, es decir, el punto medio a partir de sus latitudes y longitudes. Este punto sirve como referencia para buscar las **dos estaciones meteorológicas más cercanas**. Internamente este proceso no es trivial; son necesarios los siguientes pasos:

1. Obtener todas las estaciones, su distancia al centroide y filtrar las más cercanas en un radio de 30 kilómetros. Para ello se utiliza la fórmula de Haversine¹⁴. Esta fórmula permite obtener una estimación¹⁵ de la distancia entre dos puntos sobre la superficie terrestre usando latitud y longitud.
2. Ordenar por la distancia calculada y se seleccionan dos.
3. Obtener las mediciones históricas medias para las estaciones más cercanas. Para ello se emplea el siguiente método:

```
@Query("SELECT " +
        "m.id.fecha AS fecha, " +
        "AVG(m.temperaturaMedia) AS temperaturaMedia, " +
        "AVG(m.precipitacion) AS precipitacion, " +
        "AVG(m.temperaturaMinima) AS temperaturaMinima, " +
        "AVG(m.temperaturaMaxima) AS temperaturaMaxima, " +
        "AVG(m.direccionRachaMaxima) AS direccionRachaMaxima, " +
        "AVG(m.velocidadMediaViento) AS velocidadMediaViento, " +
        "AVG(m.rachaMaxima) AS rachaMaxima, " +
        "AVG(m.insolacion) AS insolacion, " +
        "AVG(m.presionMaxima) AS presionMaxima, " +
        "AVG(m.presionMinima) AS presionMinima, " +
        "AVG(m.humedadMedia) AS humedadMedia, " +
        "AVG(m.humedadMaxima) AS humedadMaxima, " +
        "AVG(m.humedadMinima) AS humedadMinima " +
        "FROM Medicion m " +
        "WHERE m.id.id_estacion IN :ids " +
        "GROUP BY m.id.fecha")
List<MedicionDTO> obtenerMedicionesAgrupadasPorFecha(@Param("ids")
List<String> ids);
```

Tabla 3.6 Función que consulta la BBDD usando JPA.

Finalmente se devuelve la respuesta a la petición. Esta respuesta está mapeada como “*ObjetoRespuesta*”.

¹⁴ La fórmula se ha extraído de la siguiente página:
<https://www.genbeta.com/desarrollo/como-calcular-la-distancia-entre-dos-puntos-geograficos-en-c-formula-de-haversine>

¹⁵ Se trata de una estimación puesto que la fórmula supone que La Tierra es perfectamente redonda.

3.3.2.2 Coordenadas geográficas

El funcionamiento de este endpoint consiste en introducir coordenadas geográficas (latitud y longitud) y se obtienen los datos de la sección censal que contiene dicho punto geográfico; junto con los datos históricos medios de las dos estaciones más cercanas al punto, siempre y cuando estén en un radio de 30 kilómetros.

De la misma forma que el apartado anterior, se presenta una tabla con las posibles respuestas del servidor:

Código HTTP	Estado	Contenido de la respuesta
200	OK	ObjetoRespuesta (para ver el contenido, dirigirse al Anexo 8.4)
400	Bad Request	Las coordenadas tienen un formato inválido o no existen estaciones cercanas a ellas.
		Los parámetros latitud y longitud no deben contener comas (,). Debes emplear la siguiente notación : latitud=43.5 o longitud=-2.546
		Falta el/los parámetro/s requerido.
404	Not Found	No se encuentra el endpoint. O se ha introducido un método no permitido.
500	Internal Server Error	Error interno del servidor.

Tabla 3.7 Tabla con las respuestas HTTP para coordenadas geográficas

Este proceso, más sencillo que el proceso del apartado anterior, comienza con consultar los datos¹⁶ del shapefile obtenido del INE, utilizando la clase *CodigoAuxiliar*.

Más concretamente, el método **obtenerIdSeccionDesdeCoordenadas**, utilizando la librería **GeoTools**, obtiene el id de la sección cuyo polígono contiene el punto geográfico introducido como parámetro en la petición. Una vez identificada la sección censal, se procede a obtener sus datos de la base de datos, mediante las clases “*SeccionService*” y “*SeccionRepository*”.

¹⁶ Por criterios de tamaño y dificultad de almacenamiento, el shapefile del INE habita fuera de la base de datos donde sí que están los demás datos cargados.

A continuación, de la misma forma que el endpoint anterior, se buscan los datos meteorológicos de las dos estaciones más cercanas, con la única diferencia de que en este caso no es necesario calcular ningún centroide –el propio punto geográfico introducido en la petición hace de centroide–. Por tanto, se obtienen todas las estaciones, se ordenan por la distancia al punto geográfico, se seleccionan las dos más cercanas y se calculan los datos meteorológicos medios.

Finalmente se devuelve un “*ObjetoRespuesta*” como respuesta a la petición.

3.3.2.3 Comunidad

Además de los dos endpoints originales, se propuso desarrollar un tercero que permitiera obtener información agregada a nivel de comunidad autónoma, con el fin de alimentar un modelo de predicción basado en redes LSTM (del cual se hablará más adelante en este Trabajo).

Aunque a simple vista esta ampliación pueda parecer menor, supuso una serie de cambios estructurales en el servicio. Fue precisamente en este momento cuando, como ya se comentó en la introducción del apartado **Servicio Web - Servidor**, se reorganizó el proyecto siguiendo buenas prácticas de desarrollo en Spring Boot. Hasta entonces, el servicio no estaba correctamente separado en capas ni diseñado para escalar fácilmente.

El principal reto al desarrollar este endpoint fue que una sección censal se asocia a una provincia, pero no directamente a una comunidad autónoma. Es decir, no se conocía un identificador explícito que asociara una sección censal con una comunidad. Por tanto, se necesitaba una forma de saber qué provincias pertenecen a una comunidad autónoma concreta. Para ello se utilizó un Excel proporcionado por la AEMET, que permite obtener las provincias asociadas a cada comunidad a partir del identificador CODAUTO (código de comunidad autónoma).

Relación de municipios y códigos por comunidades autónomas

CODAUTO	CPRO	CMUN	DC	NOMBRE
16	01	051	3	Agurain/Salvatierra
16	01	001	4	Alegría-Dulantzi
16	01	002	9	Amurrio
16	01	049	3	Añana
16	01	003	5	Aramaio
16	01	006	6	Armiñón

Figura 3.11 Tabla de códigos AEMET

Por tanto se puede decir que este endpoint consiste en introducir el id de la comunidad (CODAUTO) y se obtienen los datos sociodemográficos medios de **todas las secciones censales de todas las provincias que conforman dicha comunidad**; junto con los datos históricos medios de las cuatro estaciones meteorológicas más cercanas al centroide de la comunidad (aproximadamente el centroide medio de todas las secciones censales), siempre y cuando estén a menos de 70 kilómetros de él.

A continuación se ofrece una tabla con las posibles respuestas del servidor.

Código HTTP	Estado	Contenido de la respuesta
200	OK	ObjetoRespuesta (para ver el contenido, dirigirse al Anexo 8.4)
204	No Content	No se encontró la comunidad autónoma con el id introducido.
400	Bad Request	El id_comunidad tiene un formato incorrecto. El formato correcto es '\dd\', donde d es un digito entre 0 y 9.
		El id_comunidad debe estar en el rango de 01 a 17.
		Falta el/los parámetro/s requerido.
404	Not Found	No se encuentra el endpoint. O se ha introducido un método no permitido.
500	Internal Server Error	Error interno del servidor.

Tabla 3.8 Tabla con las respuestas HTTP para comunidad

3.3.3 Documentación, dependencias y tratamiento de errores

Todo servicio REST utiliza Swagger para presentar los endpoints accesibles junto con toda su información relevante. Para este proyecto también se ha generado un Swagger gracias a la dependencia springdoc-openapi-starter-webmvc-ui. Esta interfaz permite consultar los parámetros que acepta cada endpoint, los posibles códigos de respuesta y realizar peticiones reales desde el navegador.

Para su generación automática se han utilizado anotaciones como `@Operation` (descripción del endpoint), `@ApiResponses` y `@ApiResponse` (códigos de estado), `@Parameter` (parámetros de entrada) y `@Schema` junto con `@Content` para definir la estructura de la respuesta.

A modo de ejemplo se presentan las anotaciones Swagger definidas para el endpoint `/comunidad`, junto con una captura de la interfaz:

```

@GetMapping("/comunidad")
    @Operation(
        summary = "Obtener la información relativa a la comunidad y a las estaciones más cercanas a ella en función del id de la comunidad.",
        description = "El id debe pertenecer a España. Estar en un rango entre 01, 02... y 17"
    )
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "OK",
            content = @Content(mediaType = "application/json",
                schema = @Schema(implementation = ObjetoRespuesta.class))),
        @ApiResponse(responseCode = "204", description = "No se encontró la comunidad autónoma con el id empleado"),
        @ApiResponse(responseCode = "400", description = "El id_comunidad tiene un formato incorrecto. El formato correcto es '\dd\' , donde d es un digito entre 0 y 9",
            content = @Content(mediaType = "application/json")),
        @ApiResponse(responseCode = "500", description = "Error interno del servidor."),
        content = @Content(mediaType = "application/json"))
    })
    public ObjetoRespuesta obtenerComunidadPorId(@Parameter(name = "id_comunidad",
description = "Identificador de comunidad (01 a 17)", required = true, example = "01")
        @RequestParam(name = "id_comunidad") String idComunidad) {

```

Tabla 3.9 Anotaciones Swagger

The screenshot shows the Swagger UI interface for the **servicio-controller**. It displays two main sections: **servicio-controller** and **Schemas**.

servicio-controller section:

- GET /api/demografico/cp**: Obtener la media de los datos de las secciones censales que pertenecen a dicho código postal.
- GET /api/demografico/coordenadas**: Obtener la información relativa a las secciones y a las estaciones más cercanas en función de las coordenadas geográficas.
- GET /api/demografico/comunidad**: Obtener la información relativa a la comunidad y a las estaciones más cercanas a ella en función del id de la comunidad.
- GET /api/demografico/codigopostal**: Obtener la información relativa a las secciones y a las estaciones más cercanas en función del código postal.
- GET /api/demografico/centroide**: Obtener el centroide de cada una de las secciones de un municipio.

Schemas section:

- Sección >
- MedicionDTO >
- ObjetoRespuesta >
- CoordenadasGeograficas >

Figura 3.12 Captura del Swagger ofrecido por el servidor

3.3.4 Diagrama de secuencia

Para finalizar este apartado 3.3 se proporciona un diagrama de secuencia para una petición REST satisfactoria al endpoint `/comunidad`.

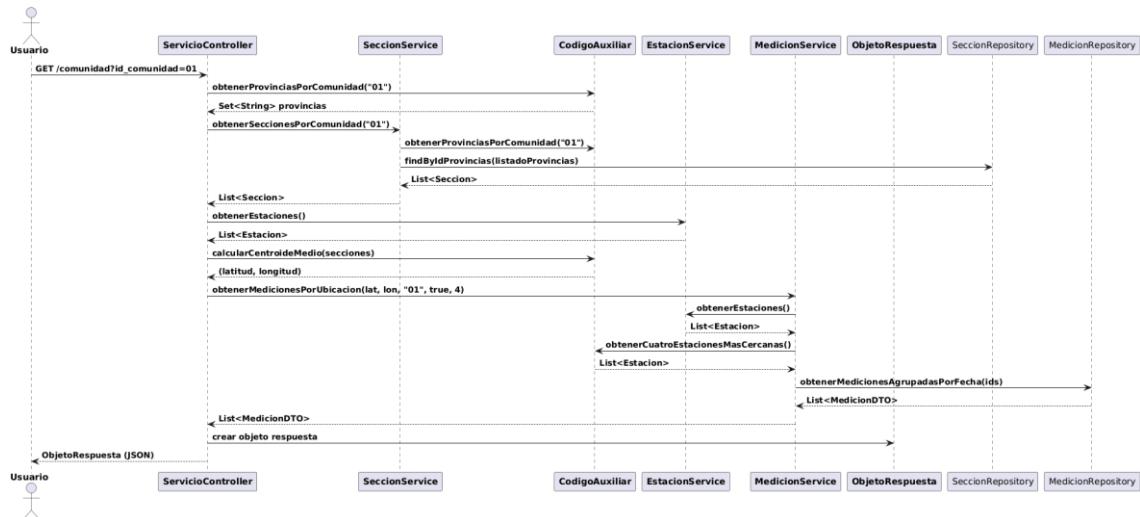


Figura 3.13 Diagrama de secuencia

3.4 Servicio Web – Cliente y LSTM

3.4.1 Introducción

En este cuarto apartado del capítulo de desarrollo se presenta la última parte de este Trabajo de Fin de Grado. Como se mostraba en la **Figura 3.1**, el cliente está compuesto por tres piezas clave:

1. Obtención de datos
2. Preprocesamiento de los datos
3. Modelo LSTM

A diferencia de los apartados **3.1**, **3.2** y **3.3**, donde el lenguaje principal era Java, en esta sección el desarrollo se ha realizado íntegramente en Python.

El objetivo principal es predecir las ventas de un negocio utilizando un modelo LSTM (Long Short-Term Memory), un tipo de red neuronal que, como ya se explicó en el apartado de tecnologías, forma parte de la familia de las redes recurrentes (RNN). Para ello, el modelo se alimenta con los datos proporcionados por el servicio desarrollado en el apartado **3.3**, actuando este bloque como cliente.

Encontrar un modelo adecuado como proporcionarle los datos correctos no es una tarea sencilla. En los siguientes apartados se explicará cómo se han obtenido y combinado los datos de ventas y los datos sociodemográficos y

meteorológicos procedentes del servicio, cómo se ha construido el modelo, qué técnicas de validación se han aplicado, y cómo se visualizan los resultados gráficamente. Este apartado no se centrará en analizar los resultados, sino en describir las herramientas, decisiones y el código desarrollado.

Además para la comparación final, se desarrollará un código que entrene al mismo modelo, pero que utilice únicamente las ventas para predecir ventas. Esto es necesario para dar una respuesta a la pregunta de si mejora la predicción de ventas o no.

3.4.2 Preparación de los datos

El primer paso antes de alimentar el modelo LSTM consiste en preparar correctamente los datos. Para ello, se han creado tres métodos, cada uno encargado de manejar la comunicación con un endpoint. En la siguiente tabla se puede ver la asociación entre el método y el endpoint al que apunta.

Nombre	Endpoint
<i>obtainDataFromPostalCode</i>	codigopostal
<i>obtainDataFromCoordinates</i>	coordenadas
<i>obtainDataFromComunidad</i>	comunidad

Tabla 3.10 Asociación entre los métodos creados y el endpoint al que apuntan

```
def obtainDataFromPostalCode(codigo_postal = None):
    if codigo_postal is None:
        codigo_postal = 28290
    params = {"codigo_postal" : codigo_postal}
    call = requests.get(entrypoint + endpoint_codigopostal, params=params)
    if(call.status_code == 400):
        return None, None
    respuesta = call.json()
    df_mediciones = pd.DataFrame(respuesta["medicionesMediaEstacionesCercanas"])
    df_mediaSecciones = pd.DataFrame(respuesta["mediaSecciones"], index = [0])
    return df_mediciones,df_mediaSecciones
```

Tabla 3.11 Método para obtener los datos del endpoint / codigopostal

Utilicemos el método que utilicemos, se transformará automáticamente la respuesta JSON en dos DataFrame: uno con las mediciones meteorológicas y otro con los indicadores sociodemográficos. Ambos se combinan posteriormente en un único DataFrame común, realizando cambios menores en el formato de las fechas.

Acto seguido, se realiza la carga y depuración del fichero Excel que contiene los datos de ventas. Se filtran únicamente los pedidos válidos cuyo estado sea alguno de los que se muestra en la siguiente lista:

```
tipos_de_orden_aceptados = ["Servido", "Entregado", "Enviado",
                            "Preparación en curso", "Enviado directo proveedor",
                            "Enviado parcialmente", "Pedido Recibido",
                            "CETELEM - Crédito Preprobado",
                            "Pedido listo para recoger en tienda", "Pago recibido"]
```

Tabla 3.12 Tipos de pedidos aceptados

Además se corrige el formato de los importes pagados y se agrupan las ventas por código postal.

Pero en caso de que obtengamos los datos para una comunidad autónoma es **necesario un paso extra**. Si recordamos en el apartado **3.3.2.3 Comunidad**, se mencionaba el uso de un Excel que contenía para cada código de comunidad, todas las provincias contenidas (y sus ids, claro). En este caso, para relacionar el código postal con la comunidad se utilizará también.

```
provincias =
df_cod_mun_esp[df_cod_mun_esp["CODAUTO"] == id_comunidad]["CPRO"].unique().tolist()
-----
df_cero_cp =
df_cero[df_cero["postcode"].astype(str).str.startswith(tuple(list_provincias_comunidad))]
```

Tabla 3.13 Código para filtrar códigos postales dado un id de comunidad

Para garantizar que se incluyen todos los días y códigos postales posibles, se genera un índice con todas las combinaciones (día-código postal) y se rellenan los valores faltantes mediante dos estrategias: la mediana por código postal y el valor cero. Estos dos enfoques dan lugar a dos variantes de los datos (df_cero y df_mediana). En el primero, se rellenan las ventas con 0 si el par fecha – código postal no existía en las ventas originales mientras que en el segundo, se calcula la mediana de las ventas y se inserta ese valor.

Hay que destacar que este último enfoque fue utilizado en una primera versión del código pero quedó descartado después en favor de df_cero. No obstante se mantiene en el código final, siguiendo el mismo enfoque que por ejemplo el apartado **3.1.1.1 Municipios**: demostrar que el **proceso de construcción fue iterativo e incremental donde se aprendía de los errores**.

Una vez ambos bloques de datos están preparados, se procede a la fusión mediante la fecha como campo común. Esto produce una tabla final donde, para cada día, se tienen tanto las ventas totales como las variables del entorno. Antes de servir este conjunto al modelo, se generan variables adicionales que se pensó podían mejorar la predicción:

- el día de la semana.
- la estación del año.
- columna booleana que indica si el día corresponde a un festivo autonómico, calculado usando la librería holidays.

Los dos primeros conjuntos de variables son categóricos, un dato que no es compatible con los modelos LSTM – que necesitan variables numéricas para entrenar-. Por ello, se utiliza una función de Pandas llamada *get_dummies()* que convierte variables categóricas a numéricas.

Pero cómo lo hace:

1. Se especifican los valores de la columna categórica. Para “día de la semana” esos valores son “lunes” “martes” “miércoles”...
2. Genera una columna nueva para cada valor único y asigna un valor booleano True o False en función del valor original de la celda del DataFrame.

Tabla original			
fecha	ventas_totales	dia_semana	
2022-01-01	566.41	Sábado	
2022-01-02	1192.98	Domingo	
2022-01-03	1507.48	Lunes	
2022-01-04	2398.82	Martes	
2022-01-05	820.80	Miércoles	
Después de aplicar pd.get_dummies			
fecha	ventas_totales	dia_Domingo	dia_Lunes
2022-01-01	566.41	False	False
2022-01-02	1192.98	True	False
2022-01-03	1507.48	False	True
2022-01-04	2398.82	False	False
2022-01-05	820.80	False	False

Tabla 3.14 Comparación para el método *get_dummies*

Finalmente, se eliminan columnas sin variabilidad y se ajusta el formato de los campos temporales para adecuarlos al modelo.

3.4.3 Diseño del modelo LSTM

Para la predicción de ventas se ha optado por el uso de una red neuronal tipo LSTM (Long Short-Term Memory), ya que este tipo de arquitectura está

especialmente diseñada para trabajar con datos secuenciales, como es el caso de las series temporales. A diferencia de otros modelos más clásicos, las LSTM permiten capturar dependencias a largo plazo en los datos, lo cual resulta clave en contextos como la evolución de las ventas, donde pueden existir patrones que se repiten de forma semanal, mensual o incluso estacional. Esta elección ya fue anticipada en el apartado dedicado a tecnologías utilizadas.

Las variables introducidas al modelo incluyen, entre otras, las ventas pasadas, indicadores meteorológicos, variables sociodemográficas, el día de la semana, la estación del año y si un día es festivo o no. En total, se podrían llegar a introducir hasta **44 variables** distintas por cada día. No obstante, se adelanta que en el apartado de evaluación se demuestra cómo incluir todas estas variables no mejora el rendimiento del modelo, y que una selección adecuada es fundamental.

Antes de alimentar el modelo, se crean secuencias de longitud fija para que el modelo pueda aprender patrones dentro de esas ventanas. Además, los datos se escalan usando “MinMaxScaler”, pero no todos juntos. Por un lado, se aplica un escalado general a todas las variables excepto a la de ventas. Por otro, se usa un segundo “MinMaxScaler” solo para la columna ventas_totales. Esto permite interpretar mejor las predicciones y evita que el modelo se vea afectado por mezclar escalas diferentes.

```
def create_sequences_multivariable(data, target_index, window_size):
    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i + window_size])
        y.append(data[i + window_size, target_index])
    return np.array(X), np.array(y)
```

Tabla 3.15 Método para crear secuencias usadas por el LSTM

En su versión final, el modelo está definido como una arquitectura simple compuesta por **una única capa LSTM**, seguida de una capa de **“Dropout”** y una capa **“Dense”** de salida. La capa LSTM tiene un número de neuronas configurable (default_neurons). La capa Dropout, con un valor configurable de default_dropout, ayuda a mitigar el sobreajuste. Finalmente, la capa Dense con una única unidad produce la predicción final de ventas. El modelo se compila utilizando el optimizador **“Adam”** y la función de pérdida **“mean squared error” (MSE)**.

```
model = Sequential([
    LSTM(default_neurons, return_sequences=False,
          input_shape=(X_train.shape[1], X_train.shape[2])),
    Dropout(default_dropout),
    Dense(1)
])
```

Tabla 3.16 Modelo LSTM final

Es importante destacar que **el diseño final del modelo y del código en general no fue estático desde el principio**, sino que ha ido evolucionando en función de los resultados obtenidos en la fase de evaluación. A lo largo del desarrollo se realizaron varias revisiones, especialmente cuando se observaban problemas de rendimiento o sobreajuste. De hecho, al menos **dos modificaciones importantes** en el diseño y preparación del modelo surgieron directamente a raíz de los resultados obtenidos durante la evaluación. Esto implica que el desarrollo del modelo LSTM y del propio servicio web estuvieron estrechamente condicionados¹⁷ por el análisis de errores y por la búsqueda de un mejor comportamiento predictivo.

3.4.4 Entrenamiento y experimentación

Una vez entrenado el modelo, se realiza la predicción sobre el conjunto de test (el 30% final de los datos) mediante “***model.predict***”. Tanto las predicciones como los valores reales se desescalan utilizando el ventas_scaler, con el objetivo de compararlos correctamente en unidades reales de venta. Finalmente ambos resultados se almacenan en un DataFrame.

```
predicted_ventas = ventas_scaler.inverse_transform(predicted)
real_ventas = ventas_scaler.inverse_transform(y_test.reshape(-1, 1))

resultados = pd.DataFrame({
    "Real": real_ventas.flatten(),
    "Predicción": predicted_ventas.flatten()
})
```

Tabla 3.17 Desescalado y almacenamiento de las ventas predichas y reales

Para encontrar la mejor configuración, se desarrolló un script que implementa una búsqueda *grid search* manual. Se basa en definir una lista de valores posibles para los hiperparámetros (número de neuronas, tamaño de ventana, *dropout*, número de épocas y batch) e ir probando combinaciones de forma sistemática. Cada modelo se evalúa y se almacena junto a sus métricas de rendimiento y observaciones personales.

Las métricas empleadas para comparar los modelos son:

- MSE (Mean Squared Error)
- MAE (Mean Absolute Error)
- MAE sobre la media móvil de las curvas de predicción y valores reales
- Diferencia entre pérdida de entrenamiento y validación, para medir el sobreajuste o subajuste

¹⁷ Como se ha explicado en la introducción del apartado 3.3 del capítulo de Desarrollo.

Con estas métricas se generó un ranking de los modelos entrenados. En total se entrenaron más de 700 configuraciones distintas¹⁸, lo que supuso un tiempo de entrenamiento superior a las 15 horas¹⁹.

3.4.5 Visualización de los datos

Para interpretar los resultados de los entrenamientos no basta con quedarse con la métrica más baja y afirmar que es el mejor modelo únicamente por eso. Es importante analizar cómo se comporta el modelo, cómo evoluciona su pérdida y qué relación tienen las variables con las ventas. Un modelo puede tener un MAE más bajo pero obtener una diferencia mucho más grande entre la pérdida de entrenamiento y validación, lo que indica que ese modelo está sobreajustándose a los datos de entrenamiento; otro modelo puede obtener un MAE bajo pero ser incapaz de captar la tendencia general de las ventas (obtiene un MAE en las medias móviles alto). Para ello se ha desarrollado un conjunto de herramientas de visualización y análisis.

Una primera visualización directa es la comparación entre las ventas reales y las predichas por el modelo. Esto permite observar visualmente si el modelo consigue predecir los picos de ventas. También se representa la evolución de la pérdida durante el entrenamiento (tanto en el conjunto de entrenamiento como en validación) para detectar posibles problemas de sobreajuste o infraajuste.

Además, se ha utilizado la técnica de **media móvil exponencial** sobre ambas curvas (real y predicha), con el objetivo de comparar mejor las tendencias en las ventas. Esta comparación también se ha evaluado mediante el MAE correspondiente.

Por otro lado, para entender mejor la importancia de las variables de entrada, se han analizado las correlaciones entre todas las variables y la variable objetivo ventas_totales. Se representará como una barra horizontal ordenada por coeficiente de correlación, destacando cuáles son las variables que más influyen positiva o negativamente en las ventas.

Finalmente, se podrán generar mapas de calor para analizar cómo afectan diferentes combinaciones de hiperparámetros (como el tamaño de ventana, el número de épocas, el batch size o el dropout) sobre el MAE medio. Estos mapas se han generado agrupando los datos experimentales y ayudan a detectar patrones de rendimiento según la configuración.

Estas visualizaciones han sido claves para interpretar de forma clara el comportamiento del modelo y para tomar decisiones sobre qué arquitectura y configuración final utilizar.

¹⁸ En el Anexo B: Figuras evaluación se proporciona una tabla con algunos ejemplos.

¹⁹ El tiempo total es una estimación. Por dar un ejemplo : un modelo con 1024 neuronas, ventana 30 y 100 épocas tardó 23 minutos.

4 Evaluación

4.1 Objetivo 1 y 2

La evaluación de ambos objetivos es sencilla por la siguiente razón: el desarrollo del proyecto ha dejado claro que se han cumplido casi en su totalidad. En lo referente al **Objetivo 1**, se han obtenido datos abiertos tanto meteorológicos (a través de la API de AEMET) como sociodemográficos (mediante archivos Excel del INE), tal como se planteaba inicialmente. La única excepción fue el almacenamiento de las predicciones meteorológicas, que se decidió no incluir por las limitaciones explicadas en el apartado **3.1.1.1 Municipios**. Aun así, se desarrolló todo el código necesario para obtenerlas y se creó la estructura en la base de datos para almacenarlas.

Respecto al **Objetivo 2**, se ha diseñado y creado una base de datos relacional en MySQL, con un modelo de datos adecuado y funcional, y se han desarrollado múltiples clases Java para automatizar la inserción y peticiones sql para su consulta.

La mera existencia de los datos almacenados en la base de datos demuestra que ambos objetivos se han alcanzado con éxito, más allá de cualquier validación externa. No obstante, a continuación se muestran algunos ejemplos de los datos almacenados:

- Mediciones medias para las estaciones “3194Y” y “3343Y” agrupadas por fecha.

Fecha	2022-01-01	2022-01-02	2022-01-03	2022-01-04	2022-01-05	2022-01-06	2022-01-07
Temperatura Media	11	9.90	9.25	6.25	5.50	4.05	4.80
Precipitación	0	0	0	5.70	7.30	0.20	0
Temperatura Mínima	2.55	2.80	3.00	2.25	2.40	-1.50	-1.85
Temperatura Máxima	19.45	17	15.5	10.25	8.60	9.60	11.35
Dirección Racha Máxima	9	21	23	26	31	99	2
Velocidad Media Viento	0.80	0.80	0.80	1.90	1.90	0.80	3.30
Racha Máxima	4.20	3.10	4.20	16.40	10.60	3.60	9.20
Humedad Media	60.5	65	78	88	77.5	74	68.5
Humedad Máxima	81.5	87.5	94.5	99.5	95.5	98.5	88.5
Humedad Mínima	31	45	62	65.5	59	55	50.5

Tabla 4.1 Mediciones

- Mediciones para la estación “1347T” por fecha.

<i>ID Estación</i>	2024-11-16	2024-11-17	2024-11-18
<i>Nombre</i>	BURELA	BURELA	BURELA
<i>Provincia</i>	LUGO	LUGO	LUGO
<i>Altitud</i>	80	80	80
<i>Latitud</i>	43.651	43.651	43.651
<i>Longitud</i>	-7.35709	-7.35709	-7.35709
<i>ID Estación</i>	1347T	1347T	1347T
<i>Fecha</i>	2024-11-16	2024-11-17	2024-11-18
<i>Temperatura Media</i>	13.1	14.3	14.2
<i>Precipitación</i>	0	0	0
<i>Temperatura Mínima</i>	10.9	13.7	12.7
<i>Hora Temp. Mínima</i>	01:58	Varias	07:50
<i>Temperatura Máxima</i>	15.3	14.9	15.7
<i>Hora Temp. Máxima</i>	12:11	00:08	11:32
<i>Dirección Racha Máxima</i>			
<i>Velocidad Media Viento</i>			
<i>Racha Máxima</i>			
<i>Hora Racha</i>			
<i>Insolación</i>			
<i>Presión Máxima</i>			
<i>Hora Presión Máxima</i>			
<i>Presión Mínima</i>			
<i>Hora Presión Mínima</i>			
<i>Humedad Media</i>	94	94	87
<i>Humedad Máxima</i>	100	96	95
<i>Hora Humedad Máxima</i>	02:40	Varias	Varias
<i>Humedad Mínima</i>	88	90	75
<i>Hora Humedad Mínima</i>	11:40	Varias	12:40

Tabla 4.2 Mediciones

- Algunos datos de las secciones censales con código postal 28290

<i>ID Sección</i>	2812701008	2812701009	2812701017	2812701018
<i>ID Distrito</i>	2812701	2812701	2812701	2812701
<i>Nombre Sección</i>	Rozas de Madrid, Las sección 01008	Rozas de Madrid, Las sección 01009	Rozas de Madrid, Las sección 01017	Rozas de Madrid, Las sección 01018
<i>Código Postal</i>	28290	28290	28290	28290

Renta Neta Media Persona	20010	31232	21393	27306
Renta Neta Media Hogar	60671	98221	68709	94162
Renta Unidad Consumo	31415	50827	34448	44046
Mediana Renta Consumo	26950	37450	31150	36750
Porcentaje Hogares Unipersonales	20.00	15.90	18.10	11.00
Población	1414	2583	1348	2320
Porcentaje Población Española	89.90	92.40	93.70	94.60
Fuente Ingresos Salario	14902	27362	17596	25626
Latitud Centroide Sección	40.55582495191268	40.54233085691632	40.5633028682395	40.52594149637766
Longitud Centroide Sección	-3.8885929884867427	-3.91462515297818	-3.8937031193664984	-3.9092216458444256

Tabla 4.3 Datos secciones censales

- Número total de códigos postales existentes

Total Códigos Postales
8660

Tabla 4.4 Total códigos postales en la base de datos

- Número total de secciones existentes

Total Secciones Censales
36992

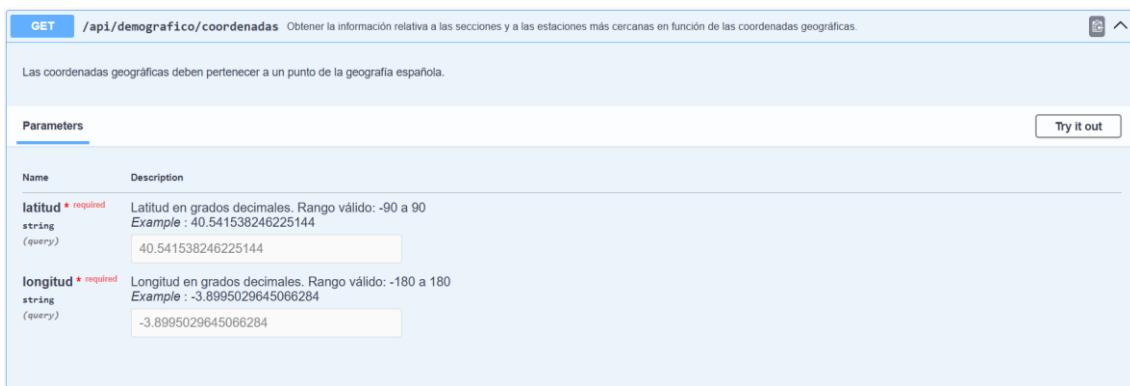
Tabla 4.5 Total secciones censales

4.2 Objetivo 3

La evaluación del servicio es directa. Basta con utilizar el propio Swagger que se ha creado y Postman para lanzar peticiones reales a los distintos endpoints, comprobar que devuelven los datos esperados y que el servidor responde correctamente ante distintos escenarios. A continuación se ofrecen algunas de las pruebas realizadas.

4.2.1 Coordenadas geográficas

4.2.1.1 Swagger



The screenshot shows the Swagger UI interface for a 'Geographic Coordinates' endpoint. At the top, it displays a 'GET' request to '/api/demografico/coordenadas'. The description is: 'Obtener la información relativa a las secciones y a las estaciones más cercanas en función de las coordenadas geográficas.' Below this, a note states: 'Las coordenadas geográficas deben pertenecer a un punto de la geografía española.' A 'Parameters' section lists two required query parameters: 'latitud' (Latitude) and 'longitud' (Longitude). Both parameters are of type 'string' and are marked as '(query)'. The 'latitud' parameter has a description: 'Latitud en grados decimales. Rango válido: -90 a 90' and an example value: '40.541538246225144'. The 'longitud' parameter has a description: 'Longitud en grados decimales. Rango válido: -180 a 180' and an example value: '-3.8995029645066284'. On the right side of the interface, there is a 'Try it out' button.

Figura 4.1 Captura del Swagger para coordenadas geográficas

4.2.1.2 Pruebas

The screenshot shows a POST request to `http://localhost:8080/api/demografico/coordenadas`. The 'Params' tab is selected, showing an empty table for query parameters. The 'Body' tab shows an empty JSON object. The response status is `400 Bad Request` with a message: "error": "Falta el parámetro requerido: 'latitud'".

Figura 4.2 Falta un parámetro de entrada

The screenshot shows a POST request to `http://localhost:8080/api/demografico/coordenadas?latitud=40&longitud=-3`. The 'Params' tab is selected, showing a table with two rows: 'latitud' with value '40' and 'longitud' with value '-3'. The 'Body' tab shows an empty JSON object. The response status is `200 OK`.

The screenshot shows a POST request to `http://localhost:8080/api/demografico/coordenadas?latitud=40&longitud=-3`. The 'Params' tab is selected, showing a table with two rows: 'latitud' with value '40' and 'longitud' with value '-3'. The 'Body' tab shows an empty JSON object. The response status is `200 OK`.

Figura 4.3 Petición correcta para coordenadas geográficas

GET | http://localhost:8080/api/demografico/coordenadas?latitud=40,3&longitud=-3,5 | Send

Params • Authorization Headers (7) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> latitud	40,3			
<input checked="" type="checkbox"/> longitud	-3,5			
Key	Value	Description		

Body Cookies Headers (4) Test Results | 400 Bad Request • 4 ms • 286 B • Save Response •

{ } JSON ▾ ▶ Preview ⚡ Visualize ▾

1 ERROR : Los parámetros latitud y longitud no deben contener comas (,).
2 Debes emplear la siguiente notación : latitud=43.5 o longitud=-2.546

Figura 4.4 Error en el formato de los parámetros

4.2.2 Código postal

4.2.2.1 Swagger

GET /api/demografico/codigopostal Obtener la información relativa a las secciones y a las estaciones más cercanas en función del código postal.

El código postal debe pertenecer a España.

Parameters

Name Description

codigopostal * required Código postal del municipio. Debe ser de 5 dígitos.
string
(query)
Example : 28290

Try it out

Figura 4.5 Captura Swagger código postal

4.2.2.2 Pruebas

The screenshot shows a POST request to `http://localhost:8080/api/demografico/codigopostal?`. The 'Params' tab is selected, showing two query parameters: 'Key' and 'Value'. The response status is 400 Bad Request, with a message: "error": "Falta el parámetro requerido: 'codigo_postal'".

Key	Value	Description
Key		
Value		Description

Body Cookies Headers (4) Test Results | 400 Bad Request • 4 ms • 202 B • Save Response ...

{ } JSON ▾ D Preview ⚡ Visualize ▾

```
1 {  
2 |   "error": "Falta el parámetro requerido: 'codigo_postal'"  
3 }
```

Figura 4.6 Faltan parámetros de entrada código postal

The screenshot shows a POST request to `http://localhost:8080/api/demografico/codigopostal?codigo_postal=0123123123`. The 'Params' tab is selected, showing three query parameters: 'Key', 'codigo_postal' with value '0123123123', and another 'Key' with empty 'Value' and 'Description'. The response status is 400 Bad Request, with a message: "ERROR : El código postal debe tener exactamente 5 caracteres.".

Key	Value	Description
Key		
codigo_postal	0123123123	
Key		Description

Body Cookies Headers (4) Test Results | 400 Bad Request • 4 ms • 206 B • Save Response ...

{ } JSON ▾ D Preview ⚡ Visualize ▾

```
1 ERROR : El código postal debe tener exactamente 5 caracteres.
```

Figura 4.7 Formato del código postal incorrecto pt1

The screenshot shows a POST request to `http://localhost:8080/api/demografico/codigopostal?codigo_postal=dddd`. The 'Params' tab is selected, showing two query parameters: 'codigo_postal' with value 'dddd'. The response status is 400 Bad Request, with the message '1 ERROR : Formato del código postal invalido.'

Figura 4.8 Formato del código postal incorrecto pt2

The screenshot shows a POST request to `http://localhost:8080/api/demografico/codigopostal?codigo_postal=28290`. The 'Params' tab is selected, showing one query parameter: 'codigo_postal' with value '28290'. The response status is 200 OK, and the JSON response is displayed, showing a nested object structure.

Figura 4.9 Petición correcta para código postal

4.2.3 Comunidad

4.2.3.1 Swagger

The screenshot shows the Swagger UI for a GET request to `/api/demografico/comunidad`. The description is: "Obtener la información relativa a la comunidad y a las estaciones más cercanas a ella en función del id de la comunidad." A note says: "El id debe pertenecer a España. Estar en un rango entre 01, 02... y 17". The 'Parameters' section shows a single parameter: `id_comunidad` (required, string, query) with value '01'. There is a 'Try it out' button.

Figura 4.10 Captura Swagger comunidad

4.2.3.2 Pruebas

The screenshot shows a Postman request to `http://localhost:8080/api/demografico/comunidad`. The 'Params' tab is selected. The 'Query Params' table has one row with 'Key' and 'Value' both empty. The 'Body' tab is selected, showing an empty JSON object. The response status is 400 Bad Request, with the error message: "error": "Falta el parámetro requerido: 'id_comunidad'".

Figura 4.11 Faltan parámetros de entrada comunidad

The screenshot shows a POST request to `http://localhost:8080/api/demografico/comunidad?id_comunidad=1`. The 'Params' tab is selected, showing a table with two rows:

Key	Value	Description
<input checked="" type="checkbox"/> id_comunidad	1	

The response status is 400 Bad Request, with the message: "El id_comunidad tiene un formato incorrecto. Debe tener exactamente dos dígitos numéricos."

Figura 4.12 Formato de id_comunidad incorrecto

The screenshot shows a POST request to `http://localhost:8080/api/demografico/comunidad?id_comunidad=11`. The 'Params' tab is selected, showing a table with three rows:

Key	Value	Description
<input checked="" type="checkbox"/> id_comunidad	11	
Key	Value	Description

The response status is 200 OK, with a JSON response body containing several nested objects and arrays, such as `medicionesMediaEstacionesCercanas` and `mediaSecciones`.

Figura 4.13 Petición correcta para comunidad

The screenshot shows a POST request to the endpoint `http://localhost:8080/api/demografico/comunidad?id_comunidad=1d`. The request includes two query parameters: `id_comunidad` with value `1d` and another unnamed parameter with value `Key`. The response is a 404 Not Found error, with the following JSON payload:

```

1 {
2   "timestamp": "2025-06-22T17:45:32.556+00:00",
3   "status": 404,
4   "error": "Not Found",
5   "path": "/api/demografico/comunidad"
6 }

```

Figura 4.14 No se encuentra el endpoint POST .../comunidad

4.3 Objetivo 4

A continuación, se presentará la evaluación de los resultados obtenidos en el desarrollo del objetivo 4. En este apartado, se parte de un modelo inicial que no logra ajustarse de manera precisa a los datos reales de ventas. Para optimizar las predicciones, se utilizarán diversas métricas de evaluación, como el error cuadrático medio (MSE) y el error absoluto medio (MAE), explicadas anteriormente. Éstas permitirán hacer una evaluación sobre la precisión del modelo (entre otras cosas). Además, se tendrá en cuenta la pérdida obtenida durante el proceso de entrenamiento y validación para evaluar la capacidad del modelo de generalizar sobre nuevos datos. Estas métricas serán clave para identificar posibles mejoras y para ajustar las variables de entrada del modelo.

En un primer momento, se optó por trabajar con datos diarios para la predicción de ventas. Sin embargo, durante las primeras pruebas se detectó una limitación significativa: al agrupar los datos por código postal, muchas fechas presentaban ventas igual a cero. Esto se debía a que la tienda analizada no tenía actividad o no había registros para determinados días, lo cual dificulta considerablemente el entrenamiento del modelo, especialmente en lo que respecta a la predicción de picos de ventas. Esta escasez de información diaria generaba un problema y afectaba directamente al rendimiento del modelo. Más tarde veremos qué problemas genera esta característica.

Ante esta situación, se decidió modificar la granularidad temporal de los datos y agruparlos semanalmente. Esta elección responde a una hipótesis razonable: es más probable que haya ventas registradas a lo largo de una semana completa

que en un único día. Con este cambio, en la segunda fase de evaluación, se logró una ligera reducción del error absoluto (también porque se decide simplificar el modelo), aunque los resultados seguían siendo insatisfactorios.

Durante esta primera fase, se utilizó una arquitectura compuesta por dos capas LSTM apiladas – una de 128 neuronas y otra de 64 neuronas –, seguidas de capas Dropout y Dense. Después se exploraron configuraciones de hiperparámetros para el tamaño de ventana, el número de épocas, el número de neuronas, el tamaño del batch y el dropout. A pesar de estos ajustes, el error absoluto medio (MAE) no bajaba de los 250 “euros”. Teniendo en cuenta que el rango de ventas en el municipio de Alcobendas oscilaba entre 0 y 2000 euros, este error suponía una desviación superior al 25%, tachándose de inaceptable.

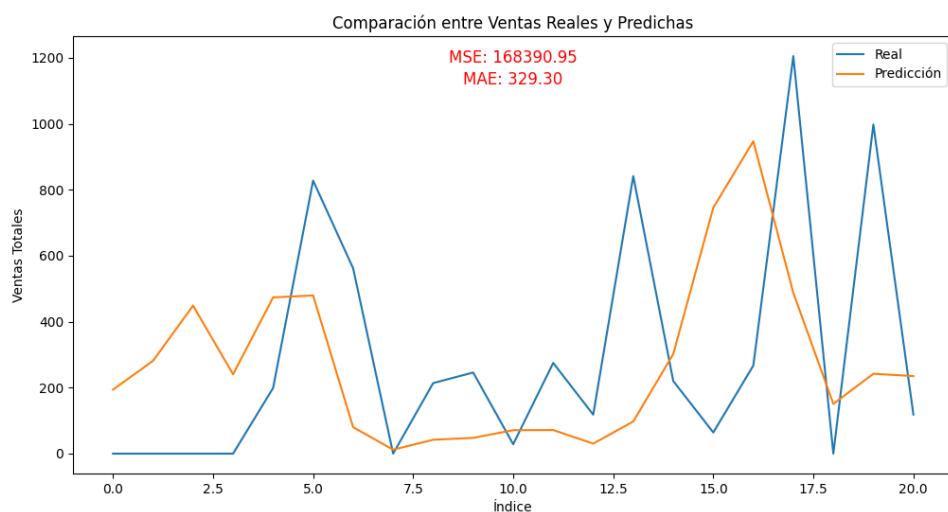


Figura 4.15 Comparación entre ventas reales y predichas utilizando una red LSTM compleja de 128 neuronas

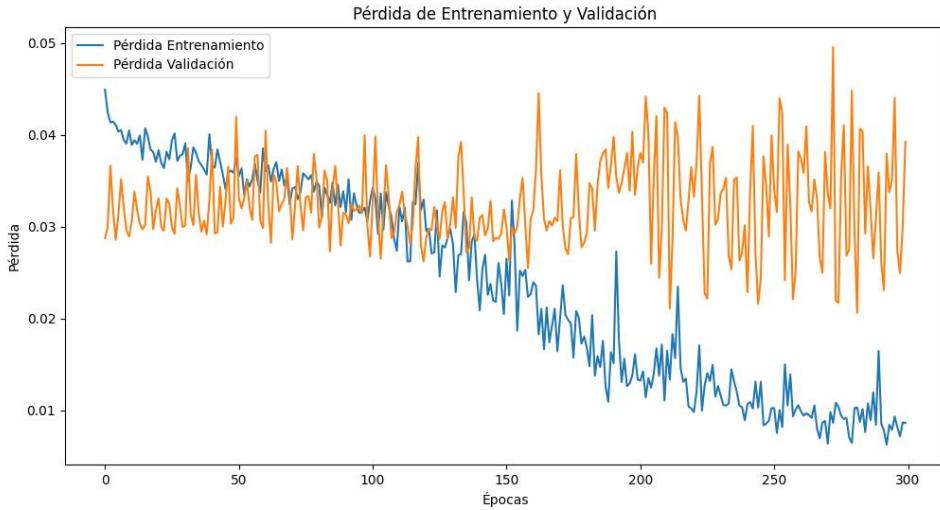


Figura 4.16 Evolución de la pérdida de entrenamiento y validación durante las 300 épocas.

Como podemos ver en las **dos figuras anteriores**, el rendimiento del modelo LSTM de 128 neuronas, ventana de 11 días, dropout de 0.3, batch size de 32 y 300 épocas es pobre. La **Figura 4.15** muestra una comparación entre las ventas reales y las predichas, devolviendo un error absoluto medio de más 300 unidades, lo que implica una falta muy grande de precisión. Por otro lado, la **Figura 4.16** refleja la evolución de la pérdida durante el entrenamiento y la validación donde se ve perfectamente un caso de sobreajuste (overfitting), en el que el modelo aprende demasiado bien los patrones del conjunto de entrenamiento pero pierde capacidad de generalizar. Además, los picos en ambas curvas podrían estar relacionados con un tamaño de batch reducido, que introduce alta varianza en los gradientes, o con un valor de dropout elevado, que afecta a la estabilidad del aprendizaje.

Con el objetivo de mejorar el rendimiento del modelo, se decide simplificar la arquitectura. Esta decisión se basó en los principios teóricos expuestos en el Estado del Arte sobre el sobreajuste y la varianza. En particular, se decide reducir la complejidad de la red, eliminando una de las capas LSTM y eliminando las capas “densas” intermedias. La nueva arquitectura se centra en una sola capa LSTM con `return_sequences=False`, seguida de una capa Dropout y una salida “densa”. Este enfoque busca paliar el sobreajuste que se ha visto antes.

Además, se estableció como objetivo primordial conseguir un MAE inferior a 150 unidades (Euros). Para alcanzar esta meta, se aplicó una búsqueda manual de hiperparámetros basada en la técnica conocida como Grid Search. Esta estrategia consiste en definir un espacio de combinaciones posibles de hiperparámetros y evaluar de forma iterativa cada combinación utilizando una métrica, en este caso el error cuadrático medio (MSE). Aunque en este trabajo la técnica se implementó de forma tradicional (bucles), permitió identificar

configuraciones óptimas del modelo. Los resultados de esta búsqueda se organizaron en un ranking con las cinco mejores configuraciones según la métrica mencionada antes. A continuación se muestra una figura con este ranking:

```
Top combinaciones:
{'window_size': 10, 'dropout': 0.1, 'epocas': 200, 'batch_size': 32, 'mse': 78914.79723729889, 'mae': 230.40281315970321}
{'window_size': 10, 'dropout': 0.1, 'epocas': 300, 'batch_size': 32, 'mse': 85565.21219557489, 'mae': 213.5753557994318}
{'window_size': 10, 'dropout': 0.3, 'epocas': 300, 'batch_size': 32, 'mse': 90085.00853245998, 'mae': 214.57831953567876}
{'window_size': 10, 'dropout': 0.5, 'epocas': 300, 'batch_size': 32, 'mse': 91762.47438613483, 'mae': 213.41957553475166}
{'window_size': 8, 'dropout': 0.5, 'epocas': 100, 'batch_size': 32, 'mse': 104158.37221223803, 'mae': 243.74700579771306}
```

Figura 4.17 Top 5 combinaciones de hiperparámetros (ventana, batch size, dropout, épocas) obtenidas para el modelo LSTM con 64 neuronas.

Tras observar la **Figura 4.17** —con las combinaciones seleccionadas—, se puede concluir que la reducción de la complejidad del modelo ha supuesto una ligera mejora en la capacidad de predecir del modelo. A pesar de que persisten problemas estructurales en los datos, como la ausencia de ventas en determinados días o la falta de información demográfica reciente (véase el apartado 3.1.2 del INE), el modelo simplificado (denominado en adelante *LSTM S*) consigue comportarse de forma más estable y predecir con mayor coherencia que su predecesor, el modelo más complejo (*LSTM C 128*).

En las gráficas de predicción y de evolución de la pérdida, se aprecia una tendencia más cercana a lo esperado: menos diferencia entre los valores reales y los predichos, y una curva de validación menos “caótica”, lo que indica una mejora en la generalización. Esta mejora se logró manteniendo una estructura simple compuesta por una sola capa LSTM, seguida de un dropout y una capadense de salida, todo acompañado de la selección de hiperparámetros.

La técnica de búsqueda manual de hiperparámetros, como se ha mencionado antes, consistió en ejecutar un código iterativo que explora sistemáticamente combinaciones posibles de valores. En este caso, se probaron ventanas temporales entre 8 a 12, valores de dropout de 0.1 a 0.5, tamaños de batch fijos en 32, y un rango de 100 a 300 épocas.

A pesar de este esfuerzo exhaustivo, no se consiguió reducir el error absoluto medio (MAE) por debajo del umbral objetivo de 100-150 unidades. Este valor se había establecido como referencia para considerar válido el modelo, teniendo en cuenta la magnitud de las ventas y el contexto del problema. La incapacidad para alcanzar dicho umbral puede explicarse por diversas limitaciones: la dispersión de las ventas por código postal —debido a que el comercio analizado es de tamaño reducido—, la falta de datos actualizados del INE (de nuevo, ver el apartado 3.1.2 del INE) para los años 2023 y 2024 (que obligó a reutilizar datos anteriores), y la propia dificultad del problema de predicción de ventas, especialmente en ausencia de datos que sean consistentes.

Como ejemplo de la mejora en la predicción del *LSTM S 64* respecto al *LSTM C 128*, se presenta la siguiente gráfica de predicción; donde el modelo —el *LSTM S*— ha sido entrenado con 10 ventanas, 0.2 de dropout, batch de 32 y 200 épocas y su correspondiente gráfica de pérdida.

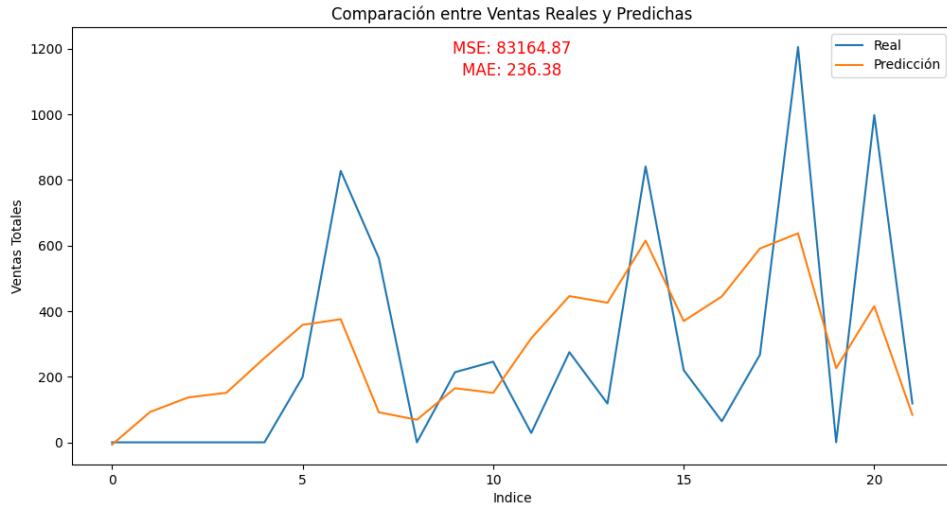


Figura 4.18 Comparación entre ventas reales y predichas utilizando el modelo LSTM S de 64 neuronas.

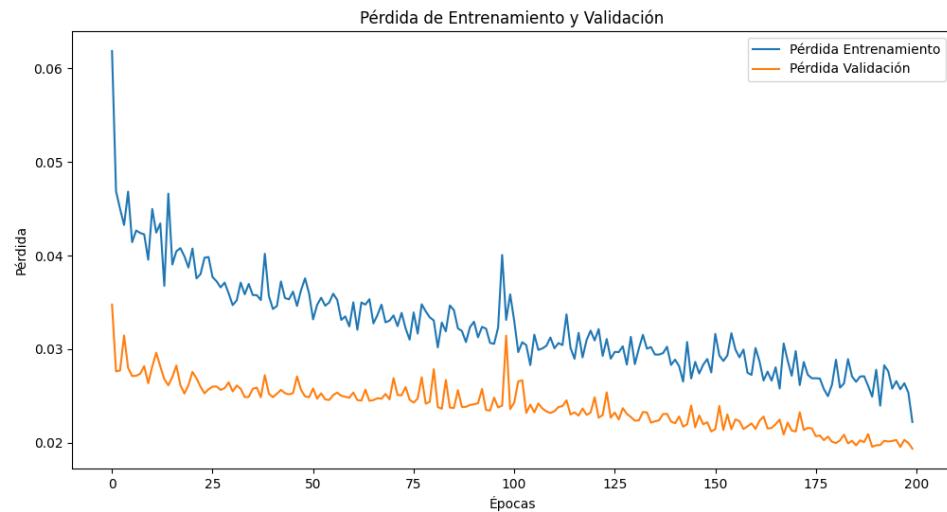


Figura 4.19 Evolución de la pérdida de entrenamiento y validación para el modelo LSTM S de 64 neuronas.

De la **Figura 4.4** y la **Figura 4.5**, sacamos lo siguiente: el modelo LSTM S de 64 neuronas alcanza una mejora notable en la precisión. Reduce el MAE, de 329 a 236 unidades, obteniendo un ajuste mejor a los datos reales. La cercanía entre las curvas de pérdida sugiere que el modelo conserva una buena habilidad para generalizar, sin sobreajustar.

Considerando que la experiencia de la etapa de evaluación pasada señala que simplificar el modelo es una estrategia efectiva para mejorar el rendimiento, decidimos repetir la búsqueda de hiperparámetros. En esta ocasión, se usa una

arquitectura aún más simple. Específicamente, se considera un modelo LSTM de solo 16 neuronas —LSTM S 16—, con el fin de minimizar la complejidad estructural del modelo.

Para esta prueba, se usan sólo las ventanas de 10 y 11 pasos de tiempo, pues estas han mostrado ser las más eficaces en ensayos previos. El resto de los hiperparámetros, como el dropout, la cantidad de épocas, y el tamaño del batch, permanecen dentro de rangos similares a los intentos anteriores.

```
Top combinaciones:
[{'window_size': 10, 'dropout': 0.2, 'epocas': 300, 'batch_size': 32, 'mse': 99976.87190575083, 'mae': 235.3778281192062},
 {'window_size': 10, 'dropout': 0.5, 'epocas': 300, 'batch_size': 32, 'mse': 106787.27243719407, 'mae': 242.44715626938944},
 {'window_size': 10, 'dropout': 0.2, 'epocas': 200, 'batch_size': 32, 'mse': 107571.86407476569, 'mae': 267.15359207924536},
 {'window_size': 10, 'dropout': 0.3, 'epocas': 300, 'batch_size': 32, 'mse': 109343.20461580367, 'mae': 265.0817852835374},
 {'window_size': 10, 'dropout': 0.5, 'epocas': 100, 'batch_size': 32, 'mse': 109657.31381055196, 'mae': 244.95411680904988}]
```

Figura 4.20 Top 5 combinaciones de hiperparámetros obtenidas para el modelo LSTM S con 16 neuronas.

En la **Figura 4.20** puede observarse el top 5 de combinaciones obtenidas con esta arquitectura simplificada. Sin embargo, los resultados no reflejan una mejora significativa. De hecho, en comparación con el modelo *LSTM S 64*, el rendimiento empeora, mostrando una mayor pérdida y menor capacidad de generalización. Esto lleva a concluir que, aunque reducir la complejidad puede ser útil hasta cierto punto, en este caso concreto, disminuir las neuronas a 16 ha implicado una pérdida de capacidad representativa que no compensa.

Antes de dar paso a una nueva etapa de evaluación centrada en una reestructuración de los datos de entrada —agrupándolos por comunidades autónomas en lugar de por códigos postales—, se plantean dos pruebas adicionales. La primera consiste en una búsqueda limitada de hiperparámetros en el modelo *LSTM S 64*, restringiendo las ventanas a 10 y 11, fijando el *dropout* en 0.2, y variando el número de épocas entre 200 y 300. Esta prueba busca afinar con mayor precisión los parámetros más prometedores observados hasta el momento.

En esta segunda búsqueda se explorarán (iterativamente) diferentes valores para todos los hiperparámetros clave: ventana temporal, dropout, tamaño del batch, número de épocas y número de neuronas. El objetivo de esta búsqueda es intentar encontrar una configuración óptima del modelo dentro del enfoque actual y si existe espacio de mejora o si es un camino sin salida; antes de cambiar a la otra estrategia mencionada.

```
Top combinaciones:
[{'window_size': 10, 'dropout': 0.2, 'epocas': 300, 'batch_size': 68, 'mse': 71244.75600203294, 'mae': 222.6183888997477},
 {'window_size': 10, 'dropout': 0.2, 'epocas': 200, 'batch_size': 60, 'mse': 80710.38576125588, 'mae': 232.36673271670767},
 {'window_size': 10, 'dropout': 0.2, 'epocas': 300, 'batch_size': 64, 'mse': 81275.76005137038, 'mae': 233.79950452289583},
 {'window_size': 10, 'dropout': 0.2, 'epocas': 200, 'batch_size': 32, 'mse': 86116.00283350462, 'mae': 226.86479224394918},
 {'window_size': 10, 'dropout': 0.2, 'epocas': 300, 'batch_size': 72, 'mse': 90597.07206071372, 'mae': 246.1977848202828}]
```

Figura 4.21 Top 5 combinaciones de hiperparámetros obtenidas para el modelo LSTM S con 64 neuronas.

En la **Figura 4.21** se muestra un modelo con una ventana de 10 días, dropout de 0.2, entre 200 y 300 épocas, y un batch variable. Aunque no se observaron mejoras relevantes en el rendimiento, lo que llevó a descartar este enfoque, la información recopilada resulta útil para orientar los siguientes experimentos. En esta prueba se plantea una búsqueda más exhaustiva que en las fases anteriores, realizando múltiples combinaciones de hiperparámetros con el objetivo de identificar la mejor configuración para el modelo. El número total de combinaciones posibles asciende inicialmente a unas 576. No obstante, gracias a la experiencia adquirida en este proceso de evaluación se pueden descartar de antemano una parte considerable de estas combinaciones, lo que permite reducir significativamente el tiempo total de entrenamiento.

Por ejemplo, se ha comprobado que ventanas distintas a 10 tienden a empeorar el rendimiento, por lo que se opta por fijar este parámetro. Del mismo modo, configuraciones con solo 16 neuronas o con valores de dropout superiores a 0.3 no han ofrecido buenos resultados, lo que justifica su exclusión. Además, los entrenamientos realizados con el modelo LSTM S 64 y distintos tamaños de batch sugieren que los valores intermedios no aportan mejoras sustanciales. Por tanto, también se elimina una parte de las posibles opciones de batch para optimizar aún más el tiempo de cómputo.

A partir de este filtro, se define un conjunto reducido de combinaciones, compuesto por los siguientes valores:

Hiperparámetro	Valores considerados
Ventana	10
Dropout	0.1, 0.2, 0.3
Batch size	32, 48, 64, 72
Épocas	200, 300
Neuronas	32, 64, 128

Tabla 4.6 Tabla con las combinaciones posibles

Este conjunto da lugar a un total de 72 combinaciones viables. Tras entrenar todos estos modelos, se seleccionarán las cinco configuraciones con menor MSE y serán analizadas para ver si son una opción viable.

```
Top combinaciones:
{'window_size': 10, 'dropout': 0.3, 'epocas': 300, 'batch_size': 32, 'neuron': 64, 'mse': 65572.69640593552, 'mae': 214.03077735518102}
{'window_size': 10, 'dropout': 0.2, 'epocas': 300, 'batch_size': 32, 'neuron': 64, 'mse': 69573.98983454418, 'mae': 206.19215886147222}
{'window_size': 10, 'dropout': 0.2, 'epocas': 300, 'batch_size': 64, 'neuron': 128, 'mse': 75864.4775324585, 'mae': 218.26308782667647}
{'window_size': 10, 'dropout': 0.3, 'epocas': 300, 'batch_size': 48, 'neuron': 64, 'mse': 76618.25128579044, 'mae': 214.05992984808393}
{'window_size': 10, 'dropout': 0.1, 'epocas': 300, 'batch_size': 64, 'neuron': 64, 'mse': 76732.91439180699, 'mae': 219.5333155116746}
{'window_size': 10, 'dropout': 0.1, 'epocas': 300, 'batch_size': 32, 'neuron': 64, 'mse': 77450.57234006951, 'mae': 219.5303350649554}
{'window_size': 10, 'dropout': 0.3, 'epocas': 300, 'batch_size': 64, 'neuron': 128, 'mse': 78355.31041518813, 'mae': 232.64526669102514}
{'window_size': 10, 'dropout': 0.3, 'epocas': 300, 'batch_size': 72, 'neuron': 128, 'mse': 78669.01327035324, 'mae': 217.15460761369556}
```

Figura 4.22 Segunda Prueba: Top combinaciones de hiperparámetros obtenidas.

En la figura anterior se aprecia un modelo que quizás pueda superar la barrera de las 200 unidades de MAE, lo que indica una mejora relevante respecto a iteraciones anteriores. Como resultado final de este proceso exhaustivo de búsqueda y ajuste, se puede concluir que, a pesar de no alcanzar el objetivo inicial fijado de entre 100 y 150 unidades de error medio absoluto (MAE), se ha logrado desarrollar un modelo con un rendimiento medianamente aceptable dadas las circunstancias del problema, mencionadas anteriormente. Entre todas las combinaciones probadas, destaca una configuración concreta por su rendimiento relativamente superior: el modelo LSTM S 64 con una ventana temporal de 10 días, un dropout de 0.2, batch size de 32 y un entrenamiento de 350 épocas. Esta configuración logra mantener un equilibrio razonable entre la capacidad de generalización y la estabilidad en el entrenamiento, evitando tanto el sobreajuste como el subajuste.

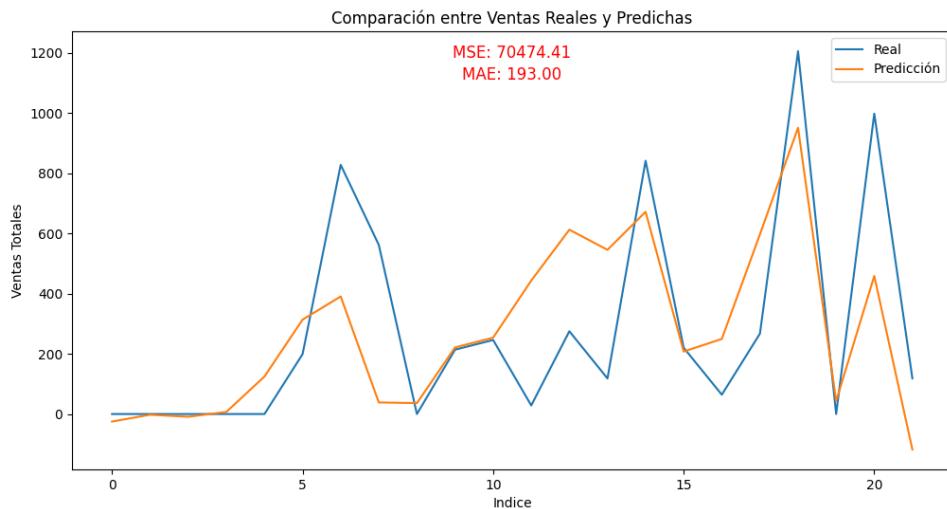


Figura 4.23 Gráfica comparación de la predicción vs real del modelo LSTM S 64 optimizado.

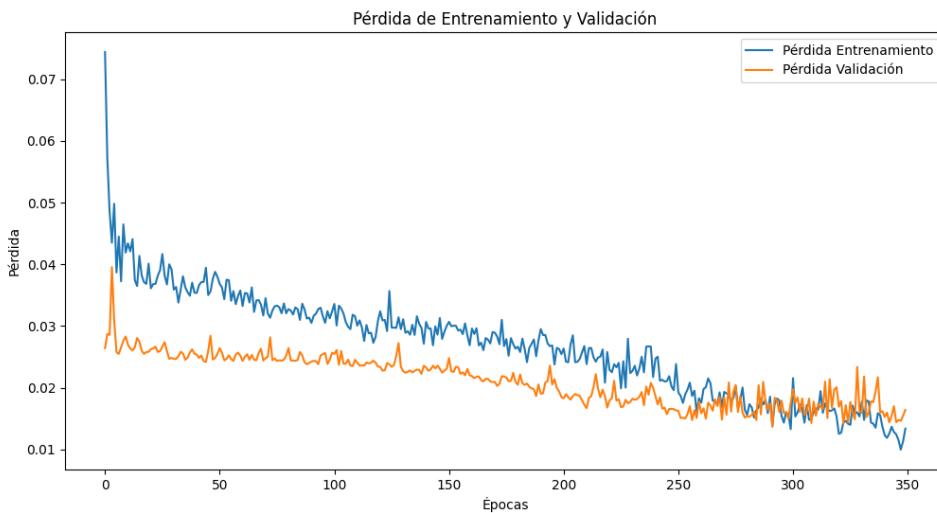


Figura 4.24 Evolución de la pérdida de entrenamiento y validación para el modelo LSTM S 64 optimizado modelo.

Las figuras **Figura 4.23** y **Figura 4.24** muestran el rendimiento del modelo LSTM S 64 en su versión más afinada. En la primera figura se observa cómo se supera la barrera psicológica de las 200 unidades de MAE, ofreciendo una predicción mucho más ajustada respecto a los modelos utilizados en fases previas. La segunda figura ilustra la evolución de la pérdida de entrenamiento y validación, donde a partir de las 250 épocas se detecta un ligero aumento en la pérdida de validación, lo que podría indicar un inicio de sobreajuste.

No obstante, ante la incapacidad del modelo para mejorar significativamente su rendimiento bajo el esquema de datos inicial (ventas agregadas semanalmente por sección censal), se planteó una nueva estrategia: agrupar los datos por comunidad autónoma en lugar de por sección censal. Esta modificación territorial permite abordar otra mejora clave: trabajar con datos diarios en lugar de semanales, lo que incrementa notablemente el número de observaciones disponibles y, por tanto, el volumen de datos con los que entrenar y evaluar el modelo. Este cambio estructural requirió la ampliación del servicio web descrito en el capítulo de desarrollo, añadiendo un nuevo endpoint capaz de devolver los datos diarios por comunidad. Con esta nueva configuración, se procedió a realizar una serie de pruebas iniciales utilizando dos modelos LSTM con distinta complejidad arquitectónica. Sin embargo, los resultados arrojaron un error absoluto medio (MAE) similar al obtenido previamente, manteniéndose en un rango de 1400–1500, a pesar de la mayor granularidad temporal.

Se procedió a analizar las posibles causas del limitado rendimiento del modelo. En primer lugar, se calculó la correlación entre las ventas y el resto de las variables disponibles. Esta exploración revela un patrón: la mayoría de las variables presentaban una correlación muy baja con la variable objetivo, algunas alrededor de 0.05. Esto sugiere que el modelo podría estar buscando patrones donde realmente no existen, forzando relaciones débiles o irrelevantes.

En la gráfica de correlación se observa que la mayoría de las variables no supera el umbral de ± 0.1 . Sin embargo, valores cercanos a cero no necesariamente indican irrelevancia para el modelo, pues técnicas como las redes LSTM pueden identificar relaciones no lineales y patrones temporales complejos que no se reflejan en esta métrica.

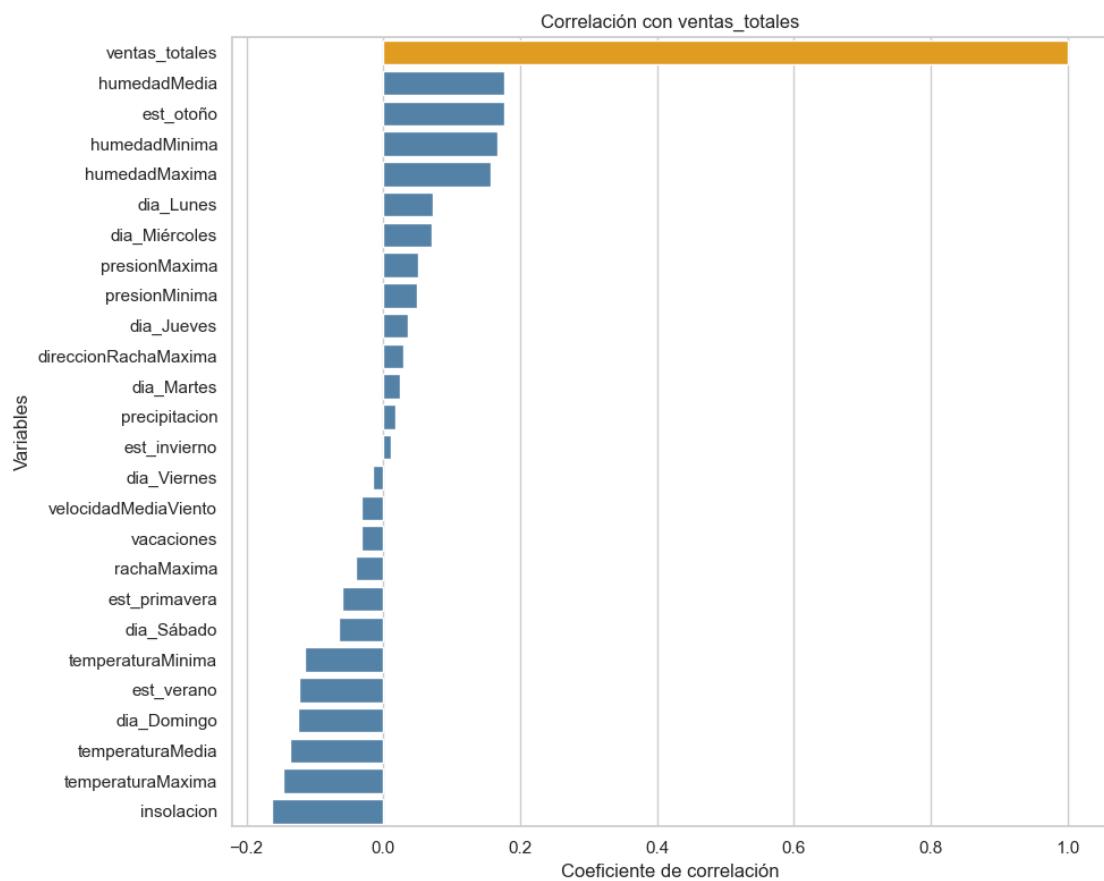


Figura 4.25 Gráfica de correlación

En consecuencia, se tomaron varias medidas orientadas a depurar las entradas al modelo:

- Eliminación de columnas estáticas: se eliminaron todas aquellas variables cuyos valores no variaban con el tiempo, como las proporcionadas por el INE para el año 2022 (ingreso medio, edad media, población, etc.). Estas variables, replicadas artificialmente a lo largo del tiempo debido a la ausencia de datos actualizados, aportaban nula variabilidad y podían estar interfiriendo negativamente en la capacidad de aprendizaje temporal del modelo.

- Ampliación del conjunto de datos: se añadieron nuevas variables que permiten capturar patrones temporales y estacionales [72] [73], fundamentales en un contexto de predicción de ventas:
 - El día de la semana (0 = lunes, 6 = domingo), con el fin de representar la estacionalidad semanal habitual en los patrones de consumo.
 - La estación del año (primavera, verano, otoño, invierno), codificada como variable categórica, que permite identificar posibles fluctuaciones de ventas vinculadas a factores climáticos o de hábitos de consumo propios de cada estación.
 - Un indicador booleano de festivo, específico para cada comunidad autónoma, que señala si un determinado día corresponde a un festivo oficial. Esta variable quizás pueda ser clave para identificar picos de ventas en días singulares.
- Evaluación adicional con medias móviles: para una mejor comprensión de la adaptación del modelo a los datos, se incorporó el uso de medias móviles como herramienta de análisis temporal. Basándonos en el estudio *"Moving averages in time series analysis"* [74], se emplea la media móvil exponencial (EMA) —una variante que asigna mayor peso a los datos más recientes, siendo más sensible a cambios recientes en la serie temporal— a diferencia de la media móvil simple que pondera de forma uniforme todos los valores. Esta técnica permite evaluar la capacidad del modelo para seguir las tendencias subyacentes de las ventas reales, mediante la comparación gráfica entre las medias móviles exponenciales de las series predichas y reales.

A continuación, se lleva a cabo una prueba inicial incorporando todas las variables adicionales introducidas recientemente. La configuración de hiperparámetros para este modelo será la siguiente:

Hiperparámetro	Valor
Ventana	14
Dropout	0.2
Batch size	32
Épocas	100
Neuronas	256

Tabla 4.7 Configuración de hiperparámetros

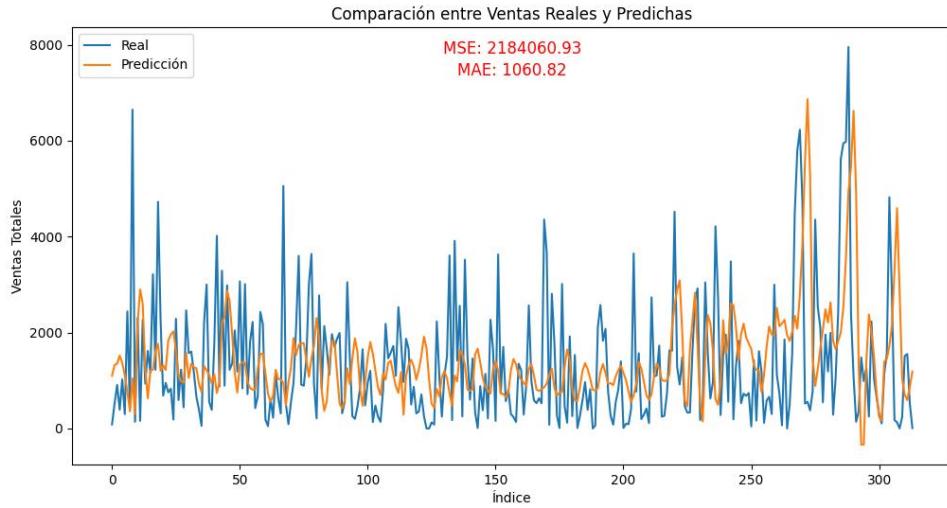


Figura 4.26 Gráfica comparación de la predicción vs real del modelo con la configuración de la Tabla 4.7 Configuración de hiperparámetros

En primer lugar, se observa una mejora significativa en el desempeño del modelo respecto a las pruebas iniciales, lo que confirma que la incorporación de las nuevas variables tiene un efecto positivo y contribuye a una mejor captura de los patrones de comportamiento de las ventas. En segundo lugar, a pesar de dicha mejora, el modelo no alcanza aún una configuración óptima, ya que presenta indicios de sobreajuste. Esto se evidencia en la discrepancia creciente entre la pérdida de entrenamiento y la de validación, tal como se muestra en la gráfica de diferencias (véase el **Anexo B: Figuras evaluación**)

Siguiendo la metodología aplicada con el modelo LSTM de 64 neuronas, se realizarán entrenamientos iterativos para identificar la mejor combinación de hiperparámetros. Los resultados de cada entrenamiento se guardarán en una tabla y se analizarán mediante mapas de calor y cálculo de correlaciones entre las variables y las métricas – mse, mae, tendencia de las medias móviles y diferencia entre entrenamiento y validación-.

Una vez completados más de 700 entrenamientos en un periodo de más de 15 horas, se dispone de una base sólida para comenzar a descartar combinaciones de hiperparámetros que ofrecen resultados inferiores (mayor error). Para ello, se aplicó la estrategia descrita anteriormente, basada en análisis visual y estadístico.

En una primera fase, se generaron mapas de calor (heatmaps) para cada combinación de hiperparámetros. Concretamente, se fijó una de las variables en el eje X y se cruzó con otra variable distinta en el eje Y, representando mediante el color el valor obtenido en una métrica específica (por ejemplo, MAE). Este procedimiento se repitió de forma sistemática para todas las combinaciones posibles entre las cinco variables analizadas (ventana, batch

size, dropout, épocas y número de neuronas). Así, por cada variable fija se generaron cuatro mapas de calor, permitiendo identificar regiones del espacio de búsqueda donde ciertas combinaciones conducen de forma consistente a mejores o peores resultados.

A continuación, se resumen los principales hallazgos extraídos de los mapas de calor, que se pueden consultar²⁰ en el **Anexo B: Figuras evaluación**

- **Batch fijo:**

- Valores de batch iguales o inferiores a 128 muestran un aumento significativo del MAE, superando los 1200 puntos de media, sin importar el valor del dropout.
- Al incrementar el tamaño del batch, el MAE mejora notablemente, destacando especialmente el rango batch 183-365 con dropout entre 0.3 y 0.4, donde el MAE se estabiliza alrededor de 950.
- En la interacción epochs-batch, lotes pequeños combinados con un alto número de épocas generan errores elevados (MAE entre 1200 y 1300). Sin embargo, aumentando ambos parámetros se obtienen mejores resultados, con MAEs cercanos a 960.
- Un caso especial se observa con pocas épocas (20-30) y batch entre 64 y 128, donde el MAE baja a valores similares (~960), aunque probablemente por subajuste.
- La combinación neuronas-batch indica que reducir ambos empeora el resultado: por ejemplo, 128 neuronas y batch 30 producen un MAE medio de 1250, mientras que aumentar neuronas a 512 con batch 30 reduce el MAE a 980.
- La relación ventana-batch muestra que un batch pequeño (ej. 7) da lugar a MAEs altos (~1100), y un batch grande (183) mejora el error a unos 962, sin importar el tamaño de la ventana.

- **Dropout fijo:**

- La variable dropout no presenta una influencia significativa en el desempeño.
- En mapas épocas-dropout, neuronas-dropout y ventana-dropout no se detectan combinaciones que impacten consistentemente el MAE.
- Esto sugiere que, dentro del rango evaluado, dropout no es un factor crítico para la optimización.

- **Épocas fijas:**

- Existe una relación clara entre épocas y neuronas: aumentar épocas suele empeorar resultados, independientemente de la complejidad del modelo.
- En general, menos épocas se asocian con mejor rendimiento.
- De forma similar, en ventana-épocas, un mayor número de épocas está asociado con un aumento del error.

- **Conclusión general:**

²⁰ Se aportan los mapas de calor más característicos.

- El tamaño del batch es el hiperparámetro más determinante para mejorar el rendimiento; es preferible batch > 128 para reducir el MAE.
- Aumentar la complejidad del modelo mediante más neuronas reduce el error, salvo una excepción con el modelo de 64 neuronas y batch 183, que también funciona bien, posiblemente por capturar patrones semestrales.
- Aumentar las épocas no mejora el desempeño y tiende a empeorarlo, indicando posibles problemas de sobreajuste.
- La ventana temporal y el dropout no muestran un impacto significativo sobre la precisión del modelo, por lo que su ajuste es menos crítico.

Por otro lado, se realizaron análisis de correlación entre las variables de entrada y diferentes métricas de evaluación. Estas métricas incluyen el MSE, el MAE, el MAE de las medias móviles y la diferencia entre la pérdida de entrenamiento y validación. El objetivo del análisis es medir cómo influye cada hiperparámetro en el rendimiento del modelo. Por ejemplo, si el número de neuronas presenta una correlación positiva con el MAE, se puede interpretar que una mayor complejidad del modelo tiende a aumentar el error medio absoluto.

A partir de todas las gráficas obtenidas, se ha calculado una media de correlaciones. El resultado se presenta en la matriz de correlación promedio entre hiperparámetros y métricas de error. Para quien desee profundizar, en el **Anexo B: Figuras evaluación**, se incluyen las matrices individuales correspondientes a cada métrica: MSE, MAE, MAE con medias móviles y diferencia de pérdida.

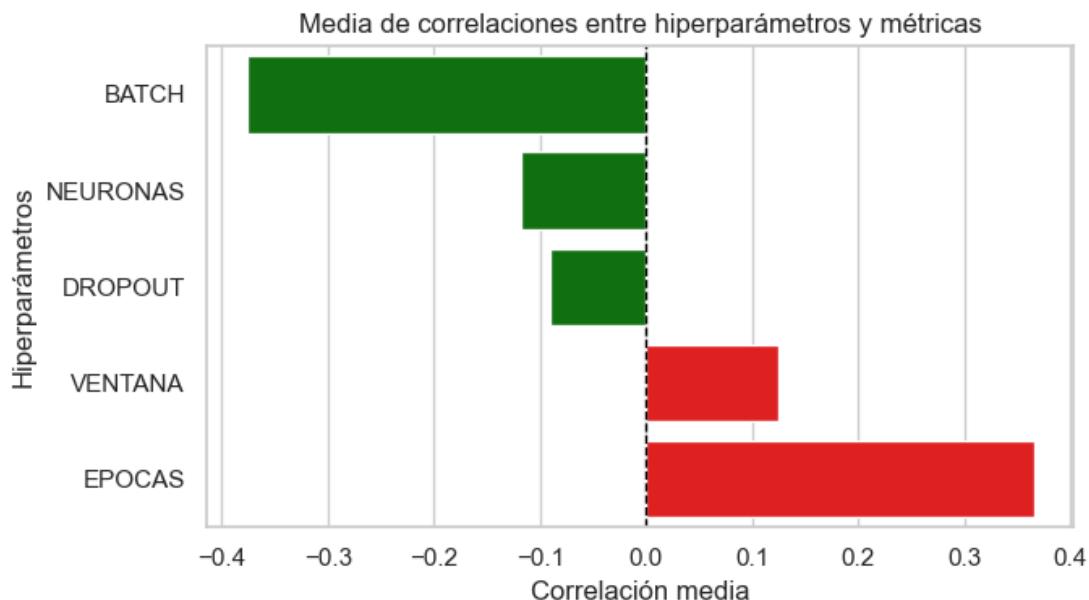


Figura 4.27 Matriz de correlación promedio entre los hiperparámetros y las métricas de error.

En la Figura 4.27, el tamaño del batch presenta una correlación media negativa de aproximadamente -0.40, lo que indica que un mayor tamaño de batch tiende a asociarse con mejores resultados (errores más bajos). En contraste, el número de épocas muestra una correlación media positiva de aproximadamente +0.40, lo que sugiere que un mayor número de épocas tiende a asociarse con un aumento del error, posiblemente debido a sobreajuste.

Otros hiperparámetros como el número de neuronas y el valor de dropout presentan correlaciones medias ligeramente negativas en torno a -0.10, indicando un efecto limitado o poco consistente sobre el rendimiento promedio de todos los modelos. Por otro lado, el tamaño de la ventana temporal presenta una correlación media positiva moderada (+0.11), lo que podría implicar que ventanas mayores no necesariamente mejoran el aprendizaje del modelo en este contexto.

Tras el análisis tanto de los mapas de calor como de las matrices de correlación, se concluye que el tamaño de batch es un factor clave para optimizar el rendimiento del modelo, prefiriéndose valores elevados, superiores a 128. Con el objetivo de tener más estabilidad en el entrenamiento y reducir el mae. Por otro lado, un aumento en el número de épocas tiende a incrementar el error, lo que sugiere que un entrenamiento más corto o controlado ayudaría a evitar el sobreajuste. Pero es importante matizar que este comportamiento podría deberse también a situaciones de *underfitting*, especialmente en modelos menos complejos. En varios casos se ha observado que, al no entrenarse lo suficiente, las predicciones del modelo tienden a aplandarse, asemejándose a una media aritmética. Aunque esto puede reducir el MAE de forma “artificial”, no implica

necesariamente que el modelo haya aprendido patrones útiles, por lo que un MAE más bajo no siempre indica un mejor desempeño real. En cuanto al número de neuronas y el dropout, su influencia es moderada y no presenta un impacto consistente, por lo que pueden ajustarse con flexibilidad según el balance deseado entre precisión y tiempo de entrenamiento. Cabe destacar que modelos con 1024 neuronas pueden superar los 20 minutos de entrenamiento por ejecución²¹, mientras que configuraciones más ligeras, como las de 64 neuronas, reducen este tiempo a apenas un minuto. Finalmente, el tamaño de la ventana temporal muestra un efecto limitado sobre el desempeño, indicando que no es un parámetro crítico para el ajuste del modelo en este contexto.

A partir de toda la información extraída de este análisis, se va a seleccionar una configuración óptima que de equilibrio entre precisión y eficiencia computacional. El modelo final propuesto utiliza una ventana temporal de 30 días, 250 épocas, un tamaño de *batch* de 183, un valor de *dropout* de 0.3 y una arquitectura de 64 neuronas. Esta configuración se ha elegido por ofrecer un rendimiento alto con un tiempo de entrenamiento significativamente reducido.

Para finalizar este capítulo y con el objetivo de evaluar la aportación de las variables externas al modelo, éste será comparado con otro modelo que mantiene la misma configuración de hiperparámetros y estructura, pero que únicamente utiliza datos históricos de ventas como variables de entrada. Esta comparación permitirá cuantificar el valor añadido de incorporar información adicional en la predicción de las ventas. Los resultados se muestran a continuación en la figura, donde se comparan las ventas reales con las predicciones generadas por ambos modelos: uno que incorpora variables externas (verde) y otro que se basa únicamente en el histórico de ventas (rojo). El área sombreada representa la diferencia entre ambas predicciones, y se indican los valores del MAE obtenidos por cada uno.

²¹ Todos los entrenamientos se han realizado en un ordenador con características muy superiores a la media. En concreto, se ha utilizado una tarjeta gráfica NVIDIA RTX 3080, que está entre el 5 % de las más potentes del mercado. Los tiempos de entrenamiento es probable que sean bastante inferiores a los que se obtendrían en equipos más comunes. La información proviene de la página Benchmarks UL: <https://benchmarks.ul.com/compare/best-gpus>

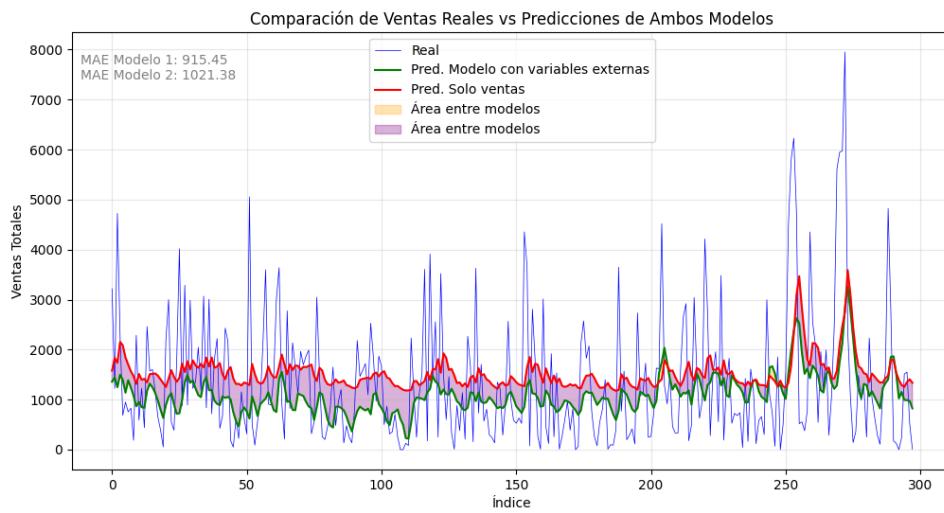


Figura 4.28 Comparación final entre las ventas reales y las predicciones generadas por los dos modelos

Aunque los resultados ponen de manifiesto el valor añadido que aportan las variables externas, está lejos de ser un modelo perfecto. Las ventas pueden estar influenciadas por muchísimos factores que no siempre se reflejan en los datos que hemos usado. Sería necesario hacer un estudio más amplio sobre qué variables realmente afectan a la demanda: por ejemplo, tener en cuenta noticias relevantes, conocer cuándo abren colegios o institutos (algo que puede disparar la venta de instrumentos musicales), los períodos de vacaciones escolares, o incluso estudios que relacionen la compra de instrumentos con ciertos grupos sociales o momentos del año. Todo esto podría ayudar a entender mejor el contexto y mejorar las predicciones, porque en muchos casos las ventas no dependen solo del tiempo o de factores internos, sino de una combinación compleja de circunstancias externas.

5 Resultados y conclusiones

Se ha conseguido establecer una conexión clara entre la introducción de variables meteorológicas, estacionales y temporales y la mejora en la predicción de ventas. Se ha desarrollado una arquitectura capaz de predecir las ventas con un error medio del 11,4 %. Teniendo en cuenta lo siguiente, estos resultados pueden considerarse aceptables:

1. La naturaleza de los datos de venta: presentan muchos picos, oscilaciones y carencia de regularidad.
2. La inexperiencia a la hora de desarrollar y entrenar modelos.
3. Las variables escogidas quizás no sean las mejores: es muy probable que estén lejos de explicar el 100 % del comportamiento de las ventas. Puede decirse que existen factores "ocultos" que no se han tenido en cuenta y que con un estudio mucho más amplio se descubriría.

Se ha comprobado que introducir demasiadas variables o aquellas cuya correlación con las ventas es casi nula puede empeorar significativamente el rendimiento del modelo. Del mismo modo, aumentar la complejidad de la arquitectura (por ejemplo, con más capas LSTM o Dense) no siempre da lugar a mejores resultados.

También se ha demostrado la gran utilidad de entrenar modelos con distintas combinaciones de hiperparámetros. Gracias a representaciones gráficas como los mapas de calor o las gráficas de correlación, se han podido identificar relaciones entre el ajuste de esos parámetros y el rendimiento final.

Sin embargo, no se ha podido establecer ninguna conexión entre variables sociodemográficas y predicción de ventas. Como se explica en el apartado de **Evaluación**, se ha tenido que prescindir de ellas debido a la falta de variabilidad anual: el INE no ofrece datos actualizados para 2023 o 2024.

En la parte de obtención y almacenamiento de datos, el resultado también ha sido positivo. Se ha estudiado qué variables se querían obtener, se ha identificado su origen, su formato, su granularidad y su calidad, y se han almacenado con éxito. Todo ello, superando barreras importantes como la de asociar coordenadas geográficas con códigos postales. Cabe destacar que existe un servicio de Correos que ofrece esta funcionalidad, pero al ser de pago, no se alinea con nuestra premisa de trabajar únicamente con datos abiertos.

Además, al crear una estructura de datos basada en la relación entre estaciones meteorológicas y municipios, se abre la puerta a futuras ampliaciones del proyecto.

En cuanto al servicio, los resultados también han sido satisfactorios. Se ha construido una herramienta que permite acceder a los datos sin necesidad de conocer la lógica interna de funcionamiento. El usuario puede obtener la misma información partiendo de distintos parámetros (comunidad, código postal, coordenadas) y siempre con una respuesta consistente.

A continuación, se presentan diversas opciones para ampliar y mejorar este Trabajo de Fin de Grado:

- **Aumentar el volumen de datos almacenados y ampliar el servicio para mostrarlos.**

El servicio podría evolucionar de una herramienta diseñada para alimentar un modelo LSTM a una plataforma mucho más versátil, útil para la toma de decisiones estratégicas. Para ello, podrían añadirse nuevas fuentes de datos abiertas como:

- **DataComex:** Datos mensuales de importación y exportación por comunidad y tipo de mercancía. Por ejemplo, si se detecta que un país vecino exporta mucho más cierto material que España, podría identificarse una oportunidad de mercado.
- **OpenStreetMap:** Permite conocer el número de comercios por zona (por ejemplo, farmacias, centros deportivos, zonas residenciales). Esto sería muy útil para empresas que buscan ubicar un nuevo negocio, como gimnasios, en zonas con demanda pero sin competencia directa. Esta información se podría combinar con renta media, población, etc.
- **Turespaña:** Proporciona datos de turismo como número de visitantes, rentabilidad o gasto medio. Si un negocio se encuentra en una zona muy turística, podría ajustar su estrategia según los períodos de baja afluencia.
- **DGT:** Permite conocer el nivel de ocupación de parkings y afluencia en zonas urbanas. Esta información, cruzada con históricos, puede ayudar a anticipar flujo de personas y ventas.
- **Crear una interfaz web (front) y desplegar el servicio online.**
Actualmente, todos los datos se encuentran en sus propias webs. Sin embargo, ninguna centraliza la información. Por ello, una posible mejora sería ofrecer una interfaz desde la que consultar todos los datos agregados de forma sencilla y rápida.
- **Entrenar el modelo LSTM en un entorno externo como AWS o Google Cloud.**

Uno de los principales problemas ha sido la capacidad de cómputo local. Entrenar el modelo ha llevado más de 15 horas. Usar un entorno externo, alquilando capacidad de cómputo, permitiría reducir este tiempo a minutos, facilitando la experimentación con más configuraciones.

A nivel personal, he de confesar que ha sido una experiencia en algunos momentos frustrante por la dificultad de encajar ciertas piezas en este puzzle. Me refiero, por ejemplo, al momento en que tuve que reorganizar todo el servicio o cuando entrenar el modelo durante horas no arrojaba mejoras claras.

Pese a pausar el proyecto durante tres meses, he conseguido retomarlo, terminarlo con éxito y desarrollar algo de lo que me siento orgulloso y de lo que puedo hablar en una futura entrevista de trabajo.

A nivel profesional, los conocimientos que he adquirido sin duda serán útiles en mi carrera como informático. Pero más allá de eso, me quedo con la capacidad de adaptarme a los cambios que han ido surgiendo durante el desarrollo, así como con el aprendizaje derivado de cada error.

Además, este proyecto me ha proporcionado una base sólida para el máster en Ciencia de Datos e Inteligencia Artificial que comenzaré el año que viene.

En definitiva, este trabajo lo podría calificar como algo muy positivo para mí. Me ha permitido aplicar conocimientos en diferentes áreas, descubrir y adquirir nuevos, superar retos reales en el desarrollo y aplicar una mentalidad crítica a la hora de evaluar mi trabajo.

6 Análisis de Impacto

El proyecto ha demostrado que se pueden construir aplicaciones basadas en **datos abiertos**. Aunque estos datos están disponibles para toda la ciudadanía, la mayoría de las personas los desconoce o no sabe cómo utilizarlos. Poner esta información su, y en particular promover su uso entre **comercios locales**, puede ser una vía real para mejorar su capacidad de toma de decisiones. Por ejemplo, el negocio estudiado en este Trabajo podría anticiparse mejor a **caídas de ventas** gracias a las predicciones del modelo desarrollado. Esto permitiría **evitar pérdidas por exceso de stock** o reducir costes operativos, ya que el modelo que integra variables sociodemográficas y meteorológicas ofrece resultados mucho más precisos que un modelo puramente histórico (*véase la Figura de la página 93*).

Más allá de la predicción como tal, el impacto se puede concretar en tres áreas clave:

1. **Gestión de Inventario:** Ajustar los pedidos de stock semanales en función de las previsiones permite reducir costes innecesarios. Esto es especialmente útil en sectores sensibles como la alimentación o la moda, donde el exceso de producto puede derivar en pérdidas por caducidad o falta de espacio de almacenamiento.
2. **Planificación de Personal:** Conociendo de antemano semanas con menor volumen de ventas, es posible optimizar los turnos del personal. Esto conlleva un ahorro en costes laborales y mejora la organización interna.
3. **Marketing:** Si se anticipa una caída en las ventas, por ejemplo, debido a condiciones meteorológicas desfavorables, se pueden lanzar campañas de marketing, descuentos o promociones. Esta estrategia ayuda a mantener la afluencia de clientes y reducir el impacto negativo de las semanas más flojas.

Además, el sistema desarrollado es **escalable**, lo que permite su reutilización o ampliación en otros contextos, como por ejemplo incorporar nuevas fuentes de datos, aplicar distintos algoritmos de predicción o extender el servicio a otros países. Lo que en definitiva haría aumentar las posibilidades de impacto, como se ha explicado en el capítulo **Resultados y conclusiones**.

7 Bibliografía

- [1] Open Knowledge Foundation, “Open Knowledge Foundation – España”. [En línea]. Disponible en: <https://okfn.org/es/>
- [2] Data.gov, “Open Government Data”. [En línea]. Disponible en: <https://data.gov/open-gov/>
- [3] A. Jiménez Carrillo, “La IA generativa por sectores”, KPMG Tendencias, febrero 2024. [En línea]. Disponible en: <https://www.tendencias.kpmg.es/2024/02/ia-generativa-sectores/>
- [4] KPMG España, *CEO Outlook 2023*. [En línea]. Disponible en: <https://assets.kpmg.com/content/dam/kpmgsites/es/pdf/2023/10/ceo-outlook-2023.pdf.coredownload.inline.pdf>
- [5] The Nobel Prize, “The Nobel Prize in Physics 2024 – Press Release”. [En línea]. Disponible en: <https://www.nobelprize.org/prizes/physics/2024/press-release/>
- [6] The Nobel Prize, *Popular Information – The Nobel Prize in Physics 2024*. [En línea]. Disponible en: <https://www.nobelprize.org/uploads/2024/11/popular-physicsprize2024-3.pdf>
- [7] The Nobel Prize, *Scientific Background – The Nobel Prize in Physics 2024*. [En línea]. Disponible en: <https://www.nobelprize.org/uploads/2024/11/advanced-physicsprize2024-3.pdf>
- [8] Gobierno de España, *IV Plan de Gobierno Abierto. Actualización 2023*. [En línea]. Disponible en: https://transparencia.gob.es/transparencia/dam/jcr:2a0d9dfb-0ee6-4329-a254-c83b09835f06/170123%20IV%20PLAN%20DE%20GOBIERNO%20ABIERTO_ACTUALIZACION.pdf
- [9] Global Data Barometer, *Análisis Regional de la Unión Europea*, pág. 88. [En línea]. Disponible en: <https://globaldatabarometer.org/wp-content/uploads/2022/05/GDB-Report-Spanish.pdf>
- [10] Agencia Estatal de Meteorología (AEMET), “Documentación del servicio de Open Data (Swagger UI)”. [En línea]. Disponible en: <https://opendata.aemet.es/dist/index.html?>
- [11] Instituto Nacional de Estadística (INE), “Datos sociodemográficos – INEbase”. [En línea]. Disponible en: <https://ine.es/dynt3/inebase/index.htm?padre=7132>
- [12] Instituto Nacional de Estadística (INE), “Cartografía censal (shapefile) – Censos 2011”. [En línea]. Disponible en: https://www.ine.es/censos2011_datos/cen11_datos_resultados_seccen.htm
- [13] Gobierno de España, “Catálogo de datos abiertos – datos.gob.es”. [En línea]. Disponible en: <https://datos.gob.es/es/catalogo>

- [14] S. Samoili, M. López Cobo, E. Gómez, G. De Prato, et al., *AI Watch – Defining Artificial Intelligence: Towards an Operational Definition and Taxonomy of Artificial Intelligence*, European Commission: Joint Research Centre, Publications Office of the European Union, 2020. [En línea]. Disponible en: <https://data.europa.eu/doi/10.2760/382730>
- [15] Departamento de Documentación del Congreso de los Diputados, *Nota documental sobre inteligencia artificial*. [En línea]. Disponible en: https://www.congreso.es/docu/docum/ddocum/notasdocumentales/nd1/inteligencia_artificial.pdf
- [16] Microsoft Azure, “¿Qué es la inteligencia artificial?”, *Cloud Computing Dictionary*, [En línea]. Disponible en: <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-artificial-intelligence#autom%C3%B3viles-sin-conductor>
- [17] Real Academia Española (RAE), “inteligencia”, *Diccionario de la lengua española (23.^a ed.)*, [En línea]. Disponible en: <https://dle.rae.es/inteligencia>
- [18] World Economic Forum, *The Future of Jobs Report 2023*. [En línea]. Disponible en: https://www3.weforum.org/docs/WEF_Future_of_Jobs_2023.pdf
- [19] High-Level Expert Group on Artificial Intelligence (AI HLEG), *A Definition of Artificial Intelligence: Main Capabilities and Scientific Disciplines*, European Commission, 18 diciembre 2018. [En línea]. Disponible en: <https://digital-strategy.ec.europa.eu/en/library/definition-artificial-intelligence-main-capabilities-and-scientific-disciplines>
- [20] IBM, “¿Qué es el aprendizaje automático?”, *IBM – Think*, 22 de septiembre de 2021. [En línea]. Disponible en: <https://www.ibm.com/topics/machine-learning>
- [21] I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” *SN Computer Science*, vol. 2, art. 160, 2021. [En línea]. Disponible en: <https://doi.org/10.1007/s42979-021-00592-x>
- [22] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons.B*, vol. 4, pp. 51–62, 2017. [En línea]. Disponible en: <https://doi.org/10.20544/HORIZONS.B.04.1.17.P05>
- [23] B. Mehlig, “Machine learning with neural networks,” *arXiv preprint arXiv:1901.05639*, 2021. [En línea]. Disponible en: <https://doi.org/10.48550/arXiv.1901.05639>
- [24] J. A. Mauricio, *Introducción al análisis de series temporales*, Universidad Complutense de Madrid, 2007. [En línea]. Disponible en: <https://www.ucm.es/data/cont/docs/518-2013-11-11-JAM-IAST-Libro.pdf>
- [25] W. Feng, N. Guan, Y. Li y X. Zhang, “Audio visual speech recognition with multimodal recurrent neural networks – Scientific Figure,” ResearchGate, 2017. [En línea]. Disponible en: https://www.researchgate.net/figure/The-standard-RNN-and-unfolded-RNN-fig1_318332317

- [26] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*, MIT Press, 2016. [En línea]. Disponible en: <https://www.deeplearningbook.org/>
- [27] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed y H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, e00938, 23 de noviembre de 2018. doi: 10.1016/j.heliyon.2018.e00938
- [28] I. Blanco, S. J. García, Á. Remesal y Fundación de las Cajas de Ahorros (CECA), "Aprendizaje profundo para series temporales en finanzas: aplicación al factor momentum," *Análisis financiero y big data*, 22 de mayo de 2023. [En línea]. Disponible en: <https://ssrn.com/abstract=4668800>
- [29] S. Hochreiter y J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735
- [30] G. Van Houdt, C. Mosquera y G. Nápoles, "A review on the Long Short-Term Memory model," *Artificial Intelligence Review*, vol. 53, 2020. doi: 10.1007/s10462-020-09838-1
- [31] G. Zhang, B. E. Patuwo y M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998. doi: 10.1016/S0169-2070(97)00044-7
- [32] T. G. Dietterich, "Overfitting and undercomputing in machine learning," *ACM Computing Surveys (CSUR)*, vol. 27, no. 3, pp. 214–217, 1995. doi: 10.1145/210399.210400
- [33] S. Geman, E. Bienenstock y R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992. doi: 10.1162/neco.1992.4.1.1
- [34] M. Perry, "The Exponentially Weighted Moving Average," *Encyclopedia of Operations Research and Management Science*, 2010. doi: 10.1002/9780470400531.eorms0314
- [35] Y. Chen, X. Zhang, Y. Huo y S. Wang, "Deep Learning-Based Estimation of Myocardial Material Parameters from Cardiac MRI," *Bioengineering*, vol. 12, art. 433, 2025. doi: 10.3390/bioengineering12040433
- [36] Y. Pu, D. Apel y C. Wei, "Applying machine learning approaches to evaluating rockburst liability: A comparison of generative and discriminative models," *Pure and Applied Geophysics*, vol. 176, 2019. doi: 10.1007/s00024-019-02197-1
- [37] Open Knowledge Foundation, "Definición de Open Data versión 2.1," [En línea]. Disponible en: <https://opendefinition.org/od/2.1/es/>
- [38] Naciones Unidas, "Datos abiertos de las Naciones Unidas," [En línea]. Disponible en: <https://data.un.org/Host.aspx?Content=About>
- [39] R. Terzić y M. Majstorovic, "Open data concept, its application and experiences," *Vojnotehnički Glasnik*, vol. 67, pp. 347–364, 2019. doi: 10.5937/vojtehg67-19935
- [40] Gobierno de España, "Portal de aplicaciones de datos abiertos," [En línea]. Disponible en: <https://datos.gob.es/es/aplicaciones>

- [41] Open Government Data, “Principios de los datos abiertos,” [En línea]. Disponible en: <https://opengovdata.org/>
- [42] Oracle, “Preguntas generales sobre Java – Centro de ayuda,” [En línea]. Disponible en: https://www.java.com/es/download/faq/index_general.html
- [43] TIOBE Software, “TIOBE Index for June 2024,” [En línea]. Disponible en: <https://www.tiobe.com/tiobe-index/>
- [44] Stack Overflow, “Developer Survey 2024,” [En línea]. Disponible en: <https://survey.stackoverflow.co/2024/>
- [45] Y. Jiang, “Research on application value of computer software development in Java programming language,” *Journal of Physics: Conference Series*, vol. 1648, art. 032152, 2020. doi: 10.1088/1742-6596/1648/3/032152
- [46] Python Software Foundation, “Sitio web oficial de Python,” [En línea]. Disponible en: <https://www.python.org/>
- [47] DB-Engines, “Ranking de sistemas de gestión de bases de datos,” [En línea]. Disponible en: <https://db-engines.com/en/ranking>
- [48] Google, “README del Repositorio Gson en GitHub,” [En línea]. Disponible en: <https://github.com/google/gson/blob/main/README.md>
- [49] Apache Software Foundation, “Apache POI – Java API for Microsoft Documents,” [En línea]. Disponible en: <https://poi.apache.org/>
- [50] Oracle, “The Java Tutorials: JDBC Basics,” [En línea]. Disponible en: <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>
- [51] E. Tenés Trillo, *Impacto de la inteligencia artificial en las empresas*, Trabajo de Fin de Grado, Universidad Politécnica de Madrid, 2023.
- [52] M. Brahmaiah y S. Talluri, “Survey on growth of business using data analytics for business intelligence in real-time world,” *Gorteria*, vol. 33, pp. 407–415, 2021.
- [53] ETSITGC y ETSII – Universidad Politécnica de Madrid, “Madrid Green Data Spaces,” 2020. [En línea]. Disponible en: <https://mgds.oeg.fi.upm.es/>
- [54] P. Llamocca Portela, *Integración y visualización de datos abiertos medioambientales*, Trabajo Fin de Máster, Universidad Complutense de Madrid, 2016. [En línea]. Disponible en: <https://hdl.handle.net/20.500.14352/25145>
- [55] A. Fernández, “Análisis de la España despoblada,” [En línea]. Disponible en: <https://anderfernandez.com/proyecto/analisis-de-la-espana-despoblada/>
- [56] A. Fernández, “BasqueMinder: herramienta para la optimización de políticas públicas,” [En línea]. Disponible en: <https://anderfernandez.com/proyecto/basqueminder-herramienta-optimizacion-politicas-publicas/>
- [57] Salesforce, “Salesforce Einstein – IA de Salesforce,” [En línea]. Disponible en: <https://www.salesforce.com/es/artificial-intelligence/>

- [58] J. Aungiers, "LSTM Neural Network for Time Series Prediction," GitHub. [En línea]. Disponible en: <https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction/tree/master>
- [59] Sanjayar, "Step by Step Guide for Sales Data Prediction [LSTM]," Kaggle, [En línea]. Disponible en: <https://www.kaggle.com/code/sanjayar/step-by-step-guide-for-sales-data-prediction-lstm>
- [60] Wikipedia, "Sección censal," *Wikipedia, La enciclopedia libre*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Secci%C3%B3n_censal
- [61] Wikipedia, "Shapefile," *Wikipedia, The Free Encyclopedia*, [En línea]. Disponible en: <https://en.wikipedia.org/wiki/Shapefile>
- [62] Instituto Nacional de Estadística (INE), "Indicadores Urbanos," *INEbase*, [En línea]. Disponible en: <https://ine.es/dynt3/inebase/index.htm?padre=12385&capsel=12384>
- [63] Instituto Nacional de Estadística (INE), "Cartografía de secciones censales del Censo 2011," [En línea]. Disponible en: https://www.ine.es/censos2011_datos/cen11_datos_resultados_seccen.htm
- [64] GeoTools, "Shapefile Plugin — GeoTools 33-SNAPSHOT User Guide," [En línea]. Disponible en: <https://docs.geotools.org/stable/userguide/library/data/shape.html>
- [65] Baeldung, "Spring Tutorial," [En línea]. Disponible en: <https://www.baeldung.com/spring-tutorial>
- [66] W. Kreuch, "Articles by wkreuch," *DEV Community*, [En línea]. Disponible en: <https://dev.to/wkreuch/>
- [67] GeeksforGeeks, "Spring Boot - Advanced Java," [En línea]. Disponible en: <https://www.geeksforgeeks.org/advance-java/spring-boot/>
- [68] Spring.io, "Building REST services – Spring Guides," [En línea]. Disponible en: <https://spring.io/guides/tutorials/rest>
- [69] Spring.io, "Spring Data JPA – Query Methods," [En línea]. Disponible en: <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html>
- [70] Spring.io, "Building a REST Service – Source Code," GitHub. [En línea]. Disponible en: <https://github.com/spring-guides/tut-rest/tree/main/nonrest/src/main/java/payroll>
- [71] Spring.io, "Exceptions," [En línea]. Disponible en: <https://docs.spring.io/spring-framework/reference/web/webflux/controller/ann-exceptions.html>
- [72] A. Zheng y A. Casari, *Feature Engineering for Machine Learning*, O'Reilly Media, 2018. ISBN: 978-1-4919-5324-2
- [73] B. V. Vishwas y A. Patel, *Hands-On Time Series Analysis with Python*, Packt Publishing, 2020. doi: 10.1007/978-1-4842-5992-4
- [74] M. Majka, *Moving Averages in Time Series Analysis*, 2024. [En línea]. Disponible en: https://www.researchgate.net/publication/384080857_Moving_Averages_in_Time_Series_Analysis

8 Anexos

8.1 Anexo A: Diagramas de Gantt con los objetivos.

8.1.1 Primer diagrama de Gantt

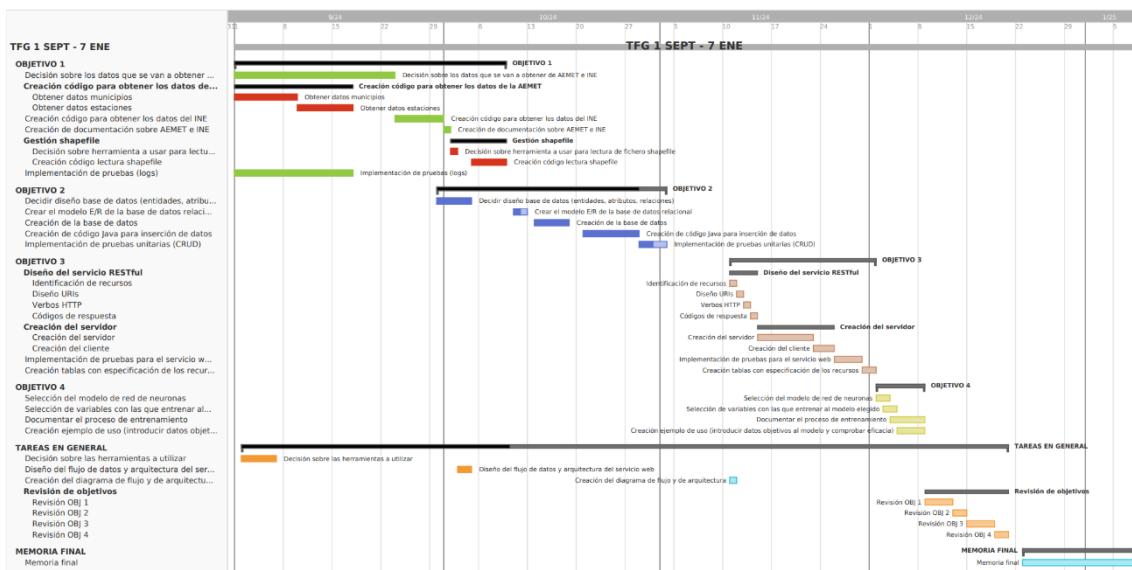


Figura Anexo 1 Diagrama de Gantt del Trabajo de Fin de Grado, dividido por objetivos.

El diagrama organiza el proyecto en cinco bloques: cuatro correspondientes a los objetivos principales del trabajo y un bloque adicional para tareas generales. El Objetivo 1 agrupa las actividades relacionadas con la obtención e inserción de datos en la base de datos. El Objetivo 2 se centra en el diseño, creación y validación de dicha base. El Objetivo 3 aborda el desarrollo del servicio REST con Spring. El Objetivo 4 abarca el desarrollo del cliente en Python, incluyendo el preprocesamiento, creación, entrenamiento y validación del modelo. El bloque adicional recoge tareas como la elaboración de diagramas de arquitectura o el propio diagrama de Gantt.

8.1.2 Segundo diagrama de Gantt



Figura Anexo 2 Segundo diagrama de Gantt del Trabajo de Fin de Grado:
planificación de revisiones

Este diagrama representa una segunda fase del proyecto, iniciada tras un periodo de pausa. Se estructura en cinco bloques correspondientes a la revisión y mejora de los cuatro objetivos principales del trabajo, seguidos por un bloque final dedicado a la redacción y revisión de la memoria.

8.2 Anexo B: Figuras evaluación

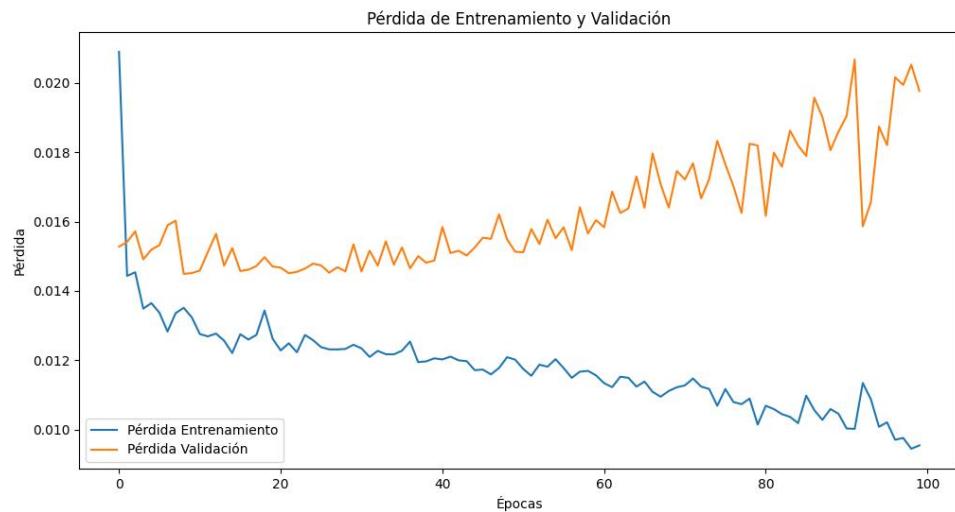


Figura Anexo 3 Evolución de la pérdida de entrenamiento y validación para este modelo.

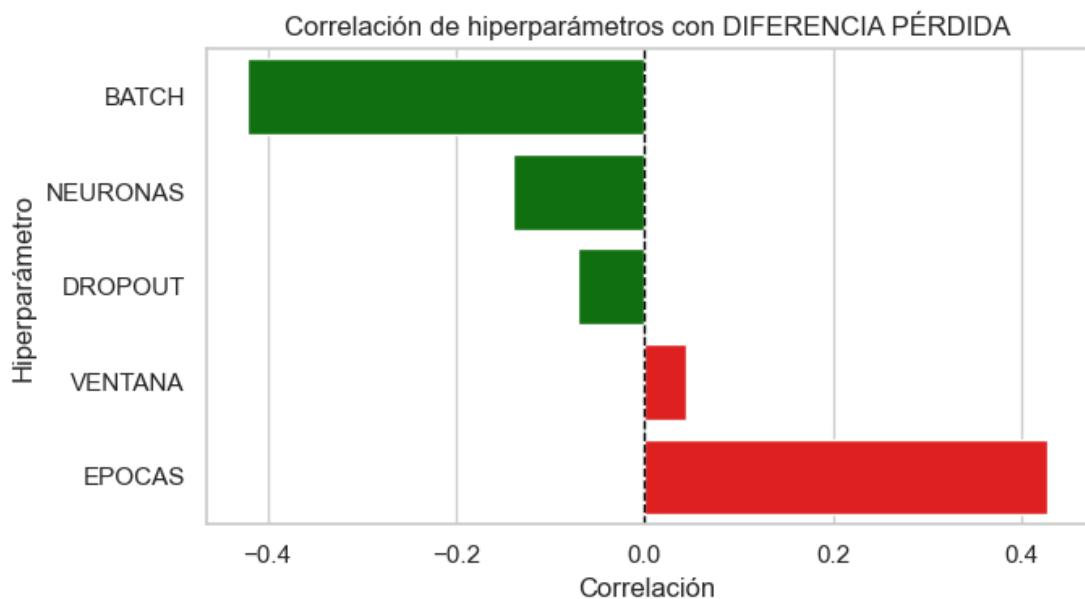


Figura Anexo 4 Gráfico de barras que muestra la correlación entre los hiperparámetros y la diferencia entre la pérdida de entrenamiento y validación. Ayuda a identificar combinaciones que podrían inducir sobreajuste.

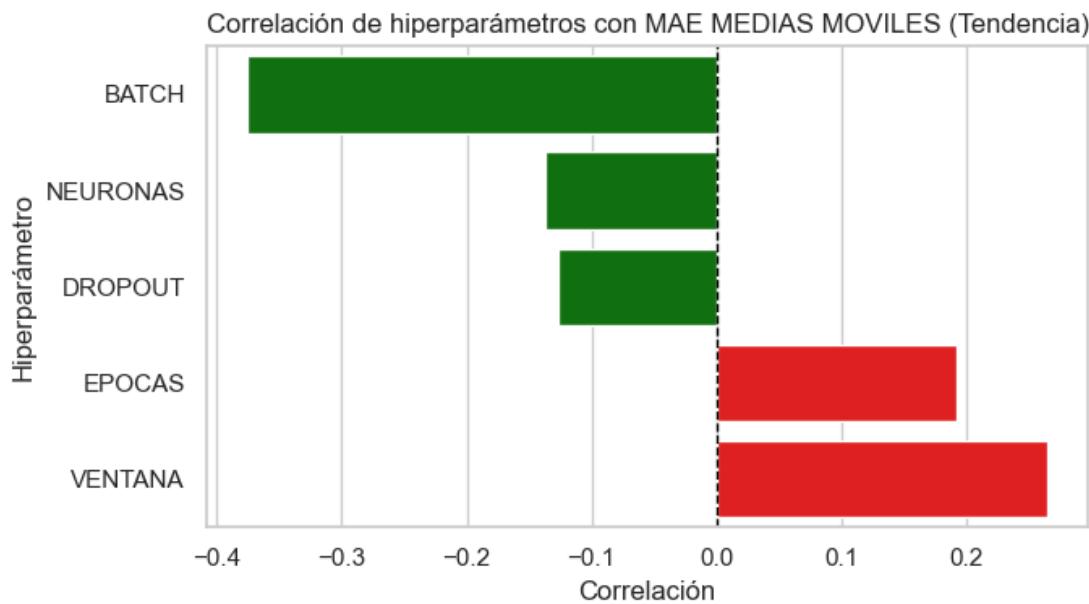


Figura Anexo 5 Gráfico de barras que representa la correlación entre los hiperparámetros y el MAE calculado sobre medias móviles.

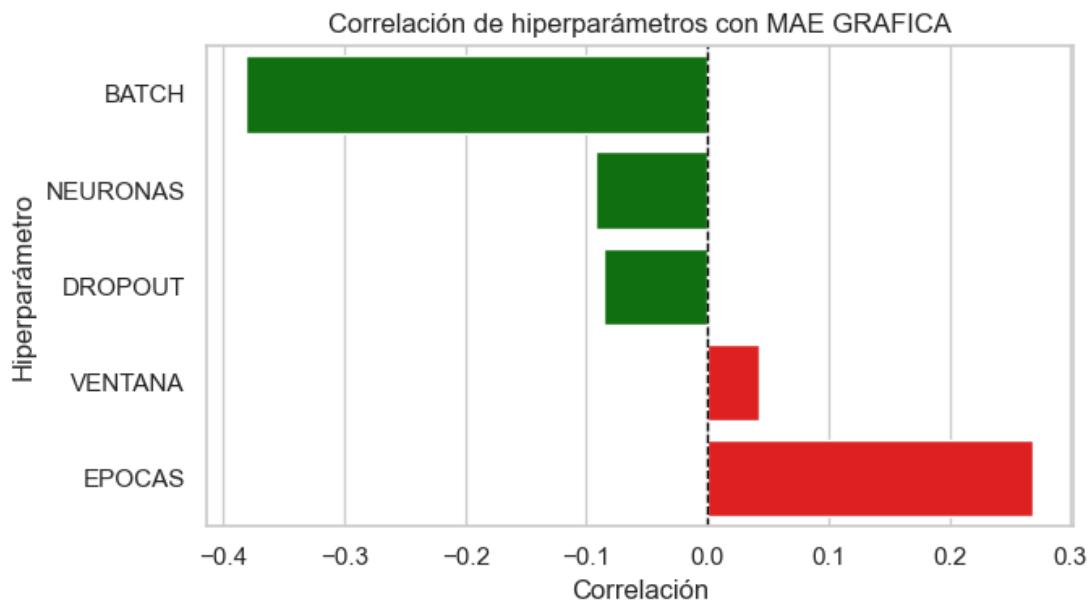


Figura Anexo 6 Gráfico de barras con la correlación entre los hiperparámetros y la métrica MAE (Error Absoluto Medio).

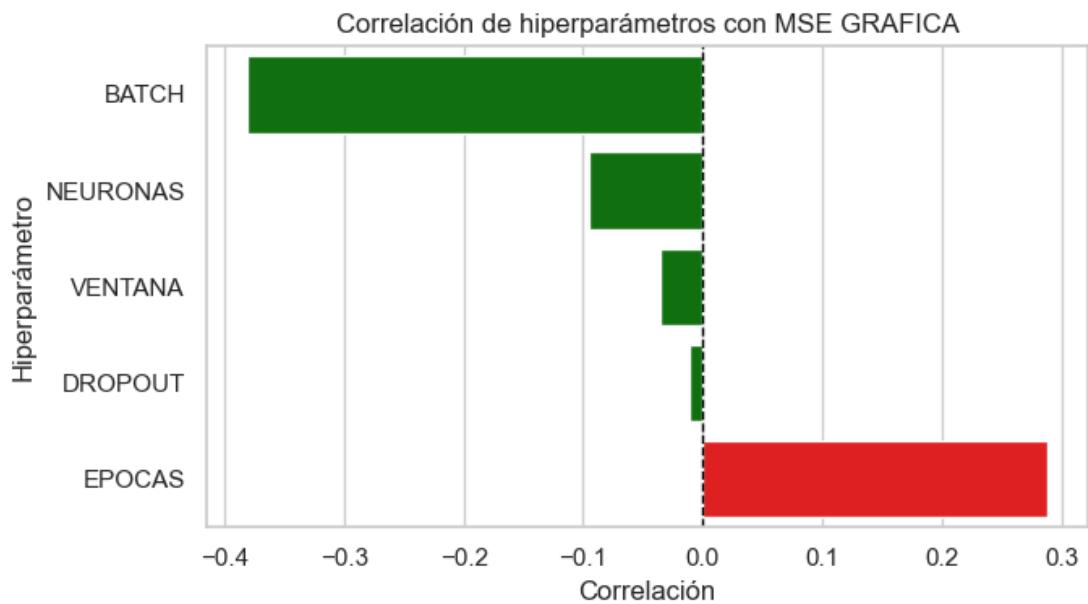


Figura Anexo 7 Gráfico de barras que muestra la correlación entre los hiperparámetros y la métrica MSE (Error Cuadrático Medio).

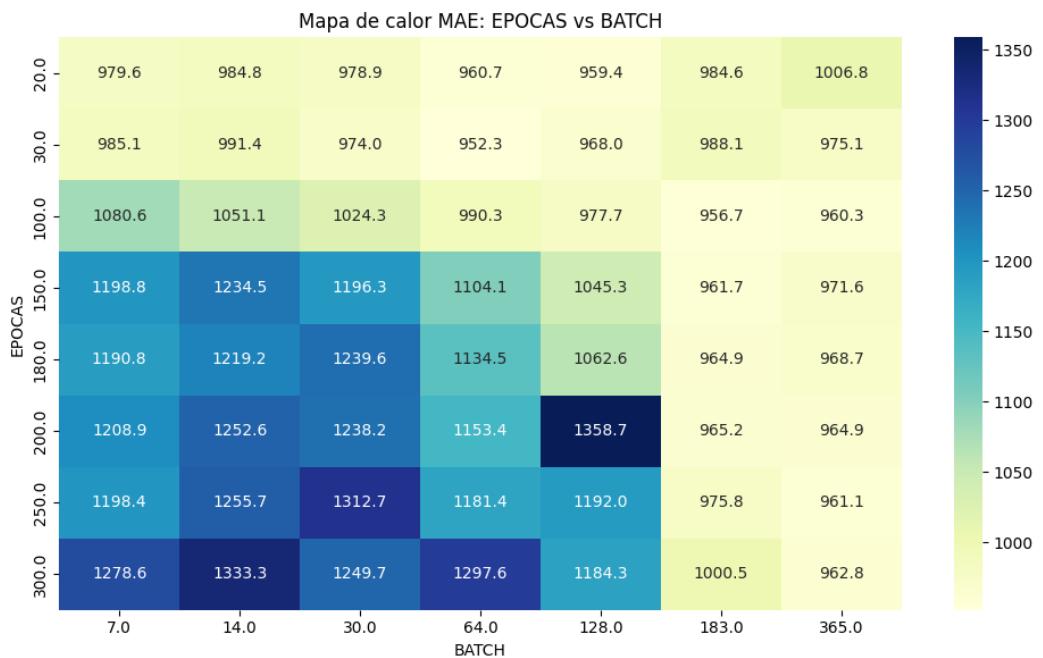


Figura Anexo 8 Mapa de calor Épocas - Batch

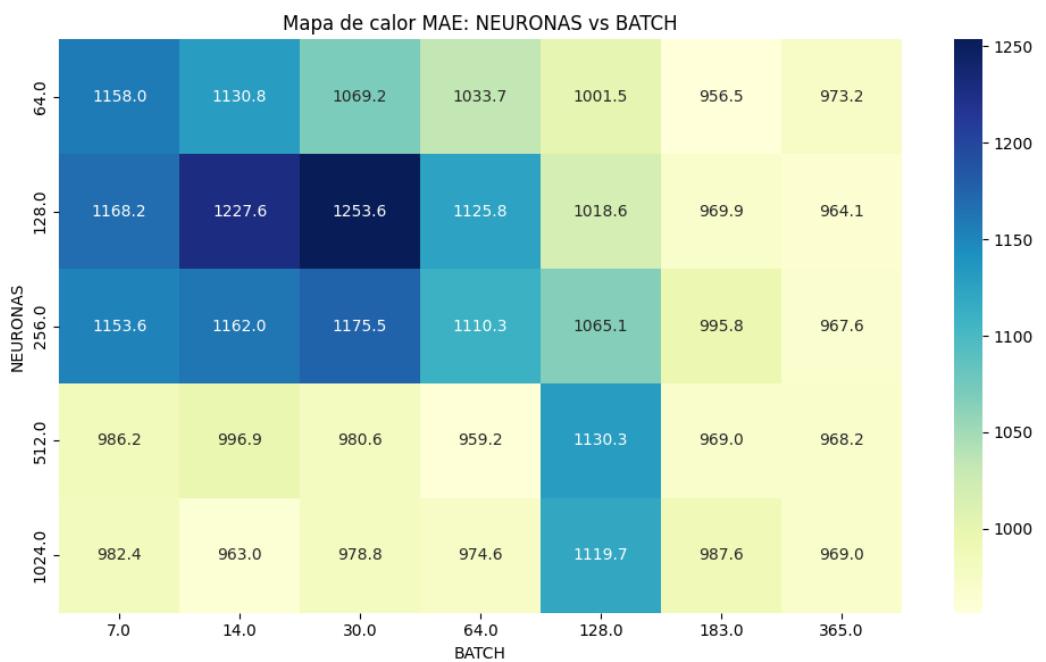


Figura Anexo 9 Mapa de calor Batch – Neuronas

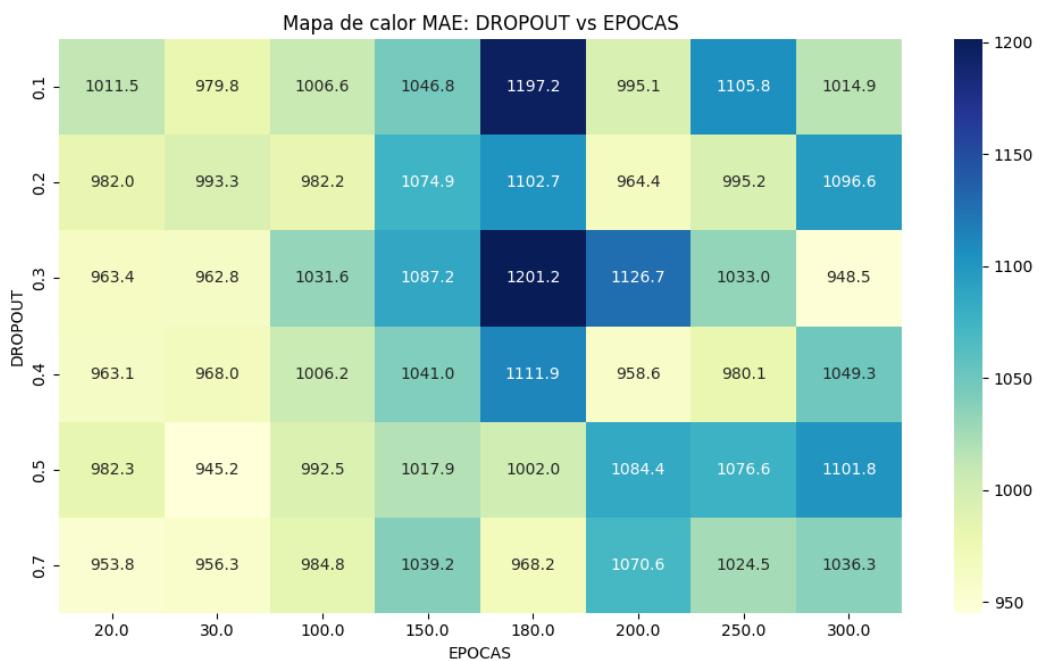


Figura Anexo 10 Mapa de calor Dropout - Épocas

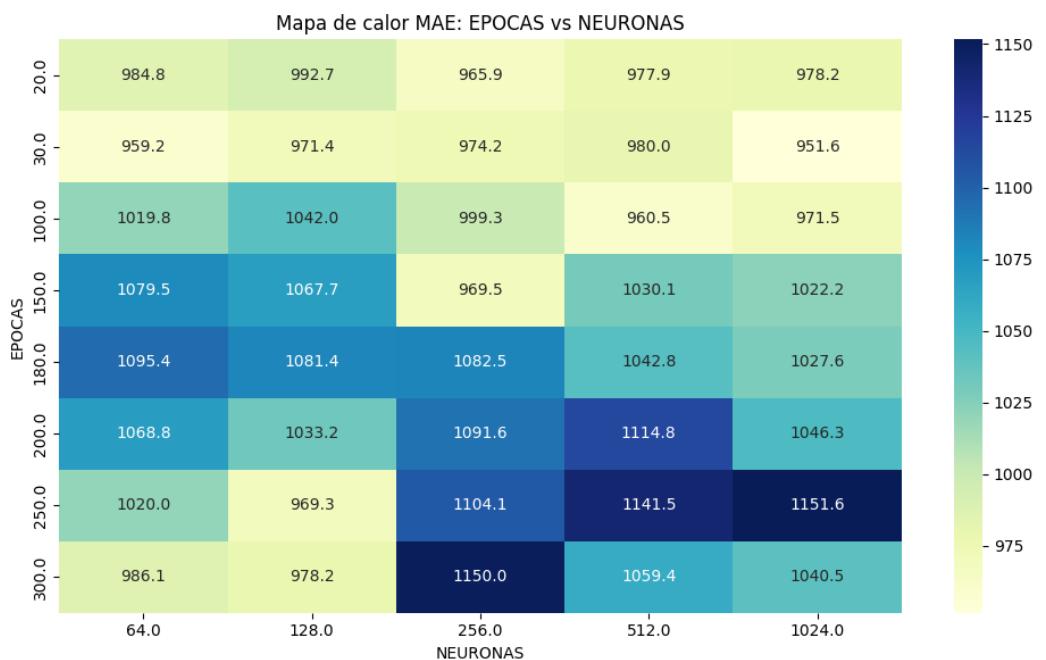


Figura Anexo 11 Mapa de calor Épocas - Neuronas

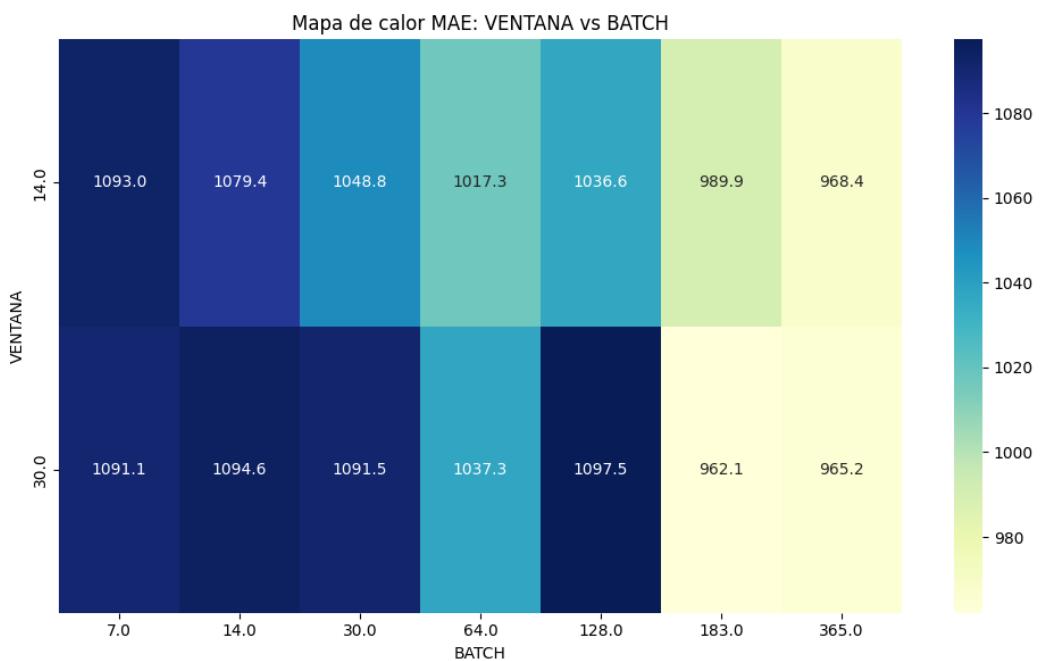


Figura Anexo 12 Mapa de calor Ventana – Batch

NEURONAS	128	256	64	512	128	256	1024
----------	-----	-----	----	-----	-----	-----	------

VENTANA	14	30	14	14	14	30	30
EPOCAS	30	20	100	20	30	100	100
BATCH	128	14	30	7	365	128	128
DROPOUT	0.2	0.7	0.5	0.3	0.2	0.5	0.1
MSE GRAFICA	1668909 90	16693 69	16698 69	167031 7	167097 7	167149 4	168528 5
MAE GRAFICA	984.79	923.71	975.64	991.57	928.65	984.55	1002.6 4
MAE MEDIAS MOVILES	465.74	510.32	428.09	488.40	473.80	481.44	484.71
DIFERENCIA PÉRDIDA	0.00303 8	0.0022 0	0.0032 2	0.0030 25	0.0014 21	0.0040 61	0.0047 47
OBSERVACIONES	X	X	X	X	X	X	X

Figura Anexo 13 Tabla con muestras de los entrenamientos del LSTM

8.3 Anexo C: Metadata de la API de la AEMET

8.3.1 Primer endpoint : Observación 12 horas

nombre_variable	descripcion	tipo_datos	requerido
idema	Indicativo climatológico de la estación meteorológica automática	string	True
lon	Longitud de la estación meteorológica (grados)	float	True
lat	Latitud de la estación meteorológica (grados)	float	True
alt	Altitud de la estación en metros	float	True
ubi	Ubicación de la estación. Nombre de la estación	string	True
fint	Fecha hora final del período de observación, se trata de datos del periodo de la hora anterior a la indicada por este campo (hora UTC)	string (AAAA-MM-DDTHH:MM:SS)	False
prec	Precipitación acumulada, medida por el pluviómetro,	float	False

	durante los 60 minutos anteriores a la hora indicada por el periodo de observación 'fint' (mm, equivalente a l/m2)		
pacutp	Precipitación acumulada, medida por el disdrómetro, durante los 60 minutos anteriores a la hora indicada por el periodo de observación 'fint'	float	False
pliqtp	Precipitación líquida acumulada durante los 60 minutos anteriores a la hora indicada por el periodo de observación 'fint'	float	False
psolt	Precipitación sólida acumulada durante los 60 minutos anteriores a la hora indicada por el periodo de observación 'fint'	float	False
vmax	Velocidad máxima del viento	float	False
vv	Velocidad media del viento	float	False
vmaxu	Velocidad máxima del viento (sensor ultrasónico)	float	False
vvu	Velocidad media del viento (sensor ultrasónico)	float	False
dv	Dirección media del viento	float	False
dvu	Dirección media del viento (sensor ultrasónico)	float	False
dmax	Dirección del viento máximo registrado	float	False
dmaxu	Dirección del viento máximo registrado por sensor ultrasónico	float	False
stdvv	Desviación estándar de velocidad del viento	float	False
stddv	Desviación estándar de la dirección del viento	float	False
stdvvu	Desviación estándar de la velocidad del viento (sensor ultrasónico)	float	False
stddvu	Desviación estándar de la dirección del viento (sensor ultrasónico)	float	False

hr	Humedad relativa instantánea del aire	float	False
inso	Duración de la insolación	float	False
pres	Presión instantánea al nivel del barómetro	float	False
pres_nmar	Presión reducida al nivel del mar	float	False
ts	Temperatura instantánea junto al suelo	float	False
tss20cm	Temperatura subsuelo a 20 cm	float	False
tss5cm	Temperatura subsuelo a 5 cm	float	False
ta	Temperatura del aire	float	False
tpr	Temperatura del punto de rocío	float	False
tamin	Temperatura mínima del aire	float	False
tamax	Temperatura máxima del aire	float	False
vis	Visibilidad	float	False
geo700	Altura a nivel barométrico de 700 hPa	float	False
geo850	Altura a nivel barométrico de 850 hPa	float	False
geo925	Altura a nivel barométrico de 925 hPa	float	False
rviento	Recorrido del viento	float	False
nieve	Espesor de la capa de nieve	float	False

Figura Anexo 14 Metadata primer endpoint AEMET

8.3.2 Segundo endpoint: Mediciones Históricas

id	Descripción	tipo_datos	requerido
fecha	fecha del dia (AAAA-MM-DD)	string	Sí
indicativo	indicativo climatológico	string	Sí
nombre	nombre (ubicación) de la estación	string	Sí
provincia	provincia de la estación	string	Sí
altitud	altitud de la estación en m sobre el nivel del mar	float	Sí
tmed	Temperatura media diaria	float	No
prec	Precipitación diaria de 07 a 07	float	No

tmin	Temperatura Mínima del día	float	No
horatmin	Hora y minuto de la temperatura mínima	string	No
tmax	Temperatura Máxima del día	float	No
horatmax	Hora y minuto de la temperatura máxima	string	No
dir	Dirección de la racha máxima	float	No
velmedia	Velocidad media del viento	float	No
racha	Racha máxima del viento	float	No
horaracha	Hora y minuto de la racha máxima	string	No
sol	Insolación	float	No
presmax	Presión máxima al nivel de referencia de la estación	float	No
horapresmax	Hora de la presión máxima (redondeada a la hora entera más próxima)	string	No
presmin	Presión mínima al nivel de referencia de la estación	float	No
horapresmin	Hora de la presión mínima (redondeada a la hora entera más próxima)	string	No
hrmedia	Humedad relativa media diaria	float	No
hrmax	Humedad relativa máxima diaria	float	No
horahrmax	Hora de la humedad relativa máxima diaria	string	No
hrmin	Humedad relativa mínima diaria	float	No
horahrmin	Hora de la humedad relativa mínima diaria	string	No

Figura Anexo 15 Metadata segundo endpoint AEMET

8.4 Anexo D: Objeto devuelvo por los endpoints del servicio

```
{
  "medicionesMediaEstacionesCercanas": [
    {
      "temperaturaMedia": 0,
      "velocidadMediaViento": 0,
      "temperaturaMaxima": 0,
      "direccionRachaMaxima": 0,
      "temperaturaMinima": 0,
      "rachaMaxima": 0,
      "humedadMedia": 0,
      "presionMaxima": 0,
    }
  ]
}
```

```

        "humedadMaxima": 0,
        "insolacion": 0,
        "humedadMinima": 0,
        "precipitacion": 0,
        "presionMinima": 0,
        "fecha": "string"
    }
],
"mediaSecciones": {
    "idSeccion": "string",
    "idDistrito": "string",
    "nombreSeccion": "string",
    "codigoPostal": "string",
    "rentaNetaMediaPersona": 0,
    "rentaNetaMediaHogar": 0,
    "rentaUnidadConsumo": 0,
    "medianaRentaConsumo": 0,
    "rentaBrutaMediaPersona": 0,
    "rentaBrutaMediaHogar": 0,
    "edadMediaPoblacion": 0,
    "porcentajeMenor18": 0,
    "porcentajeMayor65": 0,
    "tamañoMedioHogar": 0,
    "porcentajeHogaresUnipersonales": 0,
    "poblacion": 0,
    "porcentajePoblacionEspañola": 0,
    "fuenteIngresosSalario": 0,
    "fuenteIngresosPensiones": 0,
    "fuenteIngresosPDesempleado": 0,
    "fuenteIngresosOtrPrestaciones": 0,
    "fuenteIngresosOtrIngresos": 0,
    "latitud_centroide_seccion": 0,
    "longitud_centroide_seccion": 0
}
}

```

Figura Anexo 16 Formato JSON de ObjetoRespuesta