Graduado en Ingeniería Informática

Universidad a Distancia de Madrid

Departamento de Ingeniería Informática

TRABAJO DE FIN DE GRADO

Diseño y desarrollo de una aplicación web para la estimación de precios de la electricidad en España.

Autor: Juan Del Río Huertas

Directora: María Aurora Martínez Rey

MADRID, JULIO DE 2024

Declaración de originalidad:

El autor de este trabajo, Juan Del Río Huertas, con D.N.I. 36132745K, declara que el contenido de este trabajo de fin de grado es original, y en el caso de haber utilizado las ideas o contenidos de otros trabajos de otros autores están convenientemente citados, de acuerdo con las normas de citación establecidas.

El número de palabras de este trabajo, excluyendo los anexos es: 17.970.

AGRADECIMIENTOS

Quería agradecer especialmente a mi mujer, Lorena, por muchas cosas: por animarme al principio a embarcarme en esta aventura de estudiar el grado, por todo el apoyo que me ha dado todos estos años, por su paciencia, por los sacrificios que ha hecho y por esas cosas que ha hecho y que yo ni siquiera me he dado cuenta.

Quería agradecer a mis padres por haber sido siempre ejemplo de honestidad, de responsabilidad y de buscar siempre hacer las cosas bien.

Quería agradecer al resto de mi familia y amigos por su ánimo y comprensión. Especialmente agradezco a mi hermano Rafael y mi sobrino David por ayudarme a entender un poco mejor la complejidad del problema planteado.

Quería agradecer a los profesores de UDIMA, con los que he tenido la suerte de cruzarme, por ayudar a que haya disfrutado tanto estos años de carrera. Además, siempre he visto el grado como un punto de partida donde se asientan los cimientos para seguir avanzando con pie más firme. Muchas gracias por colaborar a que se hayan cumplido con creces mis expectativas.

Por último, no puedo olvidar mi agradecimiento a Aurora, que ha sido profesora de varias asignaturas y directora de este TFG, por su gran dedicación y por ayudarme a tener una visión más crítica a la hora de afrontar los retos que me han ido surgiendo.

Índice

Resumen:

Este trabajo se ha centrado en el desarrollo de un prototipo de aplicación web que ofrece a sus clientes la estimación de los precios de la electricidad del mercado mayorista diario en España.

El producto está compuesto por cuatro componentes:

- Un sistema de entrenamiento con aprendizaje automático.
- Un sistema predictor que extrae datos en tiempo real de varias páginas webs y realiza predicciones con el modelo de aprendizaje generado. Utiliza un sistema de mensajería para la comunicación entre la extracción y la predicción.
- Un sistema de almacenamiento formado por una base de datos relacional y un sistema de archivos distribuido.
- Una aplicación web como medio de comunicación entre el propietario y los clientes.

Se ha buscado que sea un sistema altamente escalable y paralelizable con el uso de tecnologías de almacenamiento y procesamiento distribuido. Además, se ha implementado usando virtualización en contenedores para facilitar su despliegue y reducir el consumo de recursos.

Palabras clave: estimación, electricidad, distribuido, paralelización, web

Abstract:

This work has focused on the development of a prototype web application that offers its clients the estimation of electricity prices in the daily wholesale market in Spain.

The product is made up of four components:

- A training system with machine learning.
- A predictor system that extracts real-time data from various web pages and makes predictions with the generated learning model. It uses a messaging system for communication between extraction and prediction.
- A storage system made up of a relational database and a distributed file system.
- A web application as a means of communication between the owner and clients.

It has been sought to be a highly scalable and parallelizable system with the use of distributed storage and processing technologies. In addition, it has been implemented using container virtualization to facilitate its deployment and reduce resource consumption.

Keywords: forecasting, electricity, distributed, parallelization, web

CAPÍTULO 1: EL PROBLEMA

1.1. Planteamiento del problema

El consumo de agua, comida y energía son la base para el desarrollo de las sociedades. En la figura 1 puede verse las interacciones del sistema de energía con los recursos naturales y la sociedad.

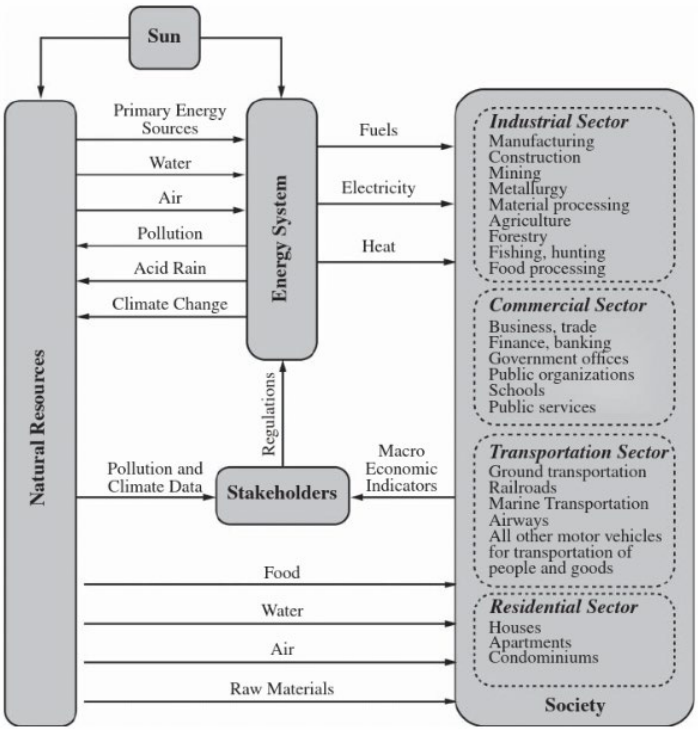


Figura 1: Relaciones con el sistema de energía (Soysal & Soysal, 2020)

El ahorro en consumo energético es un objetivo cada vez más importante para la consecución de un desarrollo sostenible (Soysal & Soysal, 2020) y la minimización de los costes que soportan los consumidores.

Dentro de las energías, siguiendo a Soysal & Soysal, la eléctrica supone una parte principal por su uso intensivo en la industria, hogares y resto de los servicios de la sociedad. Se puede diferenciar dos partes fundamentales: el suministro, que incluye su

generación y transporte, y el consumo. Desde el punto de vista del consumo, los factores fundamentales son el precio y la cantidad consumida.

El precio de la electricidad sufre constantes variaciones. En la figura 2 puede verse la evolución de los últimos doce meses en el mercado mayorista de España:

Figura2: Evolución del precio de electricidad (Operador del Mercado Ibérico de Energía, 2024)

Este precio se fija en una serie de mercados entre los que destaca el **mercado eléctrico mayorista.** En este mercado el operador media entre los generadores de electricidad y las empresas comercializadoras. Este sistema complejo funciona de la siguiente manera: cada día se presentan las ofertas de venta por parte de los generadores, donde cada uno ofrece la cantidad y el precio mínimo al que está dispuesto a vender, y las ofertas de adquisición por parte de las comercializadoras estableciendo un precio máximo al que están dispuestos a comprar. El operador del mercado genera las curvas de oferta y demanda y obtiene el precio (EUR/MWh) en el punto de corte de ambas para cada una de las 24 horas del día siguiente. Después envía el resultado al operador del sistema de transporte (REE) para que determine si es físicamente posible por posibles restricciones en las redes, dependiendo de esta limitación el precio puede variar algo.

Los consumidores finales contratarán el suministro con las comercializadoras. A menudo, estos no disponen de información actualizada de todos los factores que afectan al precio y carecen de los conocimientos especializados sobre el mercado eléctrico. Por lo que es posible que no tomen al decisión más acertada a la hora de contratar o de elegir el mejor momento para consumir la electricidad.

En este trabajo se ha tenido en cuenta una muestra de dos operadores internacionales y dos operadores nacionales que ofrecen el servicio de estimación de precios para consumo o herramientas para personal técnico, donde configurar a medida sus modelos de aprendizaje automático. Éstos tienen en común que requieren solicitar oferta para saber el coste del producto y que los servicios que ofrecen son muy generales en cuanto al ámbito geográfico o el objeto de las estimaciones. Las empresas consultadas han sido:

- QuantRisk (QuantRisk, 2024)
- ICIS (Independent Commodity Intelligence Service, 2024)
- Decide (Decide , 2024)
- Aleasoft (Aleasoft, 2024)

En base a lo anterior, se considera desarrollar un sistema de minería de datos o KDD (knowledge discovery in databases) que realice previsiones futuras a corto y medio plazo de los precios de la electricidad en el mercado mayorista de España (Lara, 2014). De esta forma, los usuarios podrían optimizar la toma de decisiones en cuanto a su gestión energética.

El producto que se ofrezca al mercado tendrá en cuenta los siguientes aspectos:

- Ampliación del rango de clientes, acercando a la pequeña y mediana empresa y a los consumidores domésticos este servicio. Se ofertarán públicamente unos precios estandarizados para eliminar la barrera inicial de tener que solicitar un estudio previo al proveedor.
- Realización de estimaciones centradas en el mercado eléctrico español y basadas en un estudio más detallado de sus circunstancias concretas con el objetivo de afinar los resultados.

1.2. Objetivo general y específicos

El objetivo general de este trabajo consiste en el diseño de una aplicación web que ofrezca el servicio de estimación de los precios de la electricidad.

Como objetivos específicos se destacan los siguientes:

- Realizar un estudio de mercado y de la situación actual.
- Estudiar los conceptos básicos relacionados con la minería de datos.
- Diseñar un sistema de minería de datos altamente escalable y confiable.
- Desarrollo de un prototipo funcional del sistema, incluyendo una aplicación web que tenga un apartado exclusivo para clientes.

1.3. Justificación

Para entender mejor la complejidad del problema expuesto en el primer punto se ahonda en este apartado en algunos conceptos relacionados.

La electricidad es una mercancía con características muy especiales (Özden-Schilling, 2021), ya que es difícil de almacenar, el transporte está muy unido a la infraestructura sin posibilidad opciones alternativas y tiene una demanda muy inelástica dado que no varía mucho ante cambios en los precios.

Los **agentes** que intervienen en el sistema eléctrico de España son:

- **Generadores**: producen electricidad tanto de fuentes gestionables como el carbón, gas natural, hidráulica e importación y de fuentes no gestionables como la energía nuclear y gran parte de las energías renovables (solar, eólica, biomasa, geotérmica,…). En la figura 3 se muestra una comparativa de la producción según las distintas fuentes.

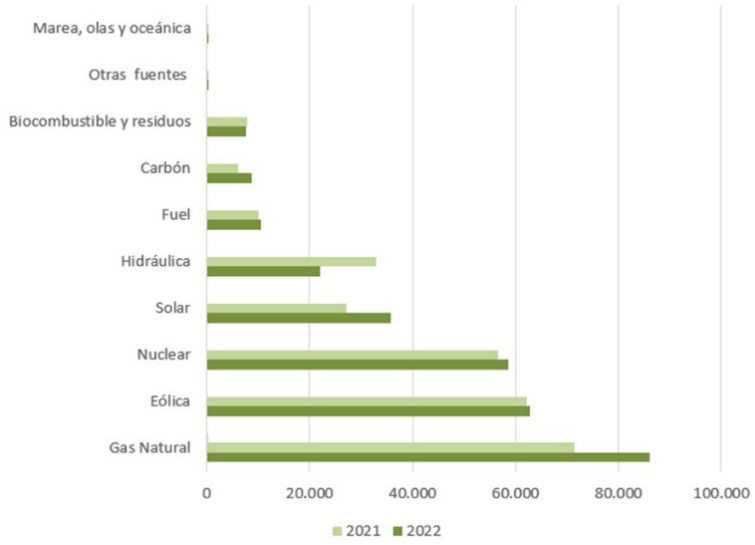


Figura 3: Producción bruta total de electricidad en España (GWh)
(Ministerio de la Transición Ecológica y el Reto Demográfico, 2023)

- **REE (Red Eléctrica de España)**: transportista único y operador (TSO) del sistema eléctrico español. Se encarga de asegurar que el suministro sea continuo y seguro.
- **Compañías distribuidoras**: llevan la electricidad desde la red de transporte en alta tensión hasta los consumidores en media o baja tensión. Hay más de 300 compañías, muchas están relacionadas con las grandes comercializadoras (Comisión Nacional de los Mercados y la Competencia, 2024).
- **Comercializadoras**: hacen de intermediarias entre los generadores y los consumidores. Hay más de 500 compañías, entre las que destacan Iberdrola, Endesa y Naturgy (Comisión Nacional de los Mercados y la Competencia, 2024).
- **Consumidores domésticos y no domésticos (empresas y exportadores):** en la figura 4 se muestra una comparativa por tipo de consumidor.

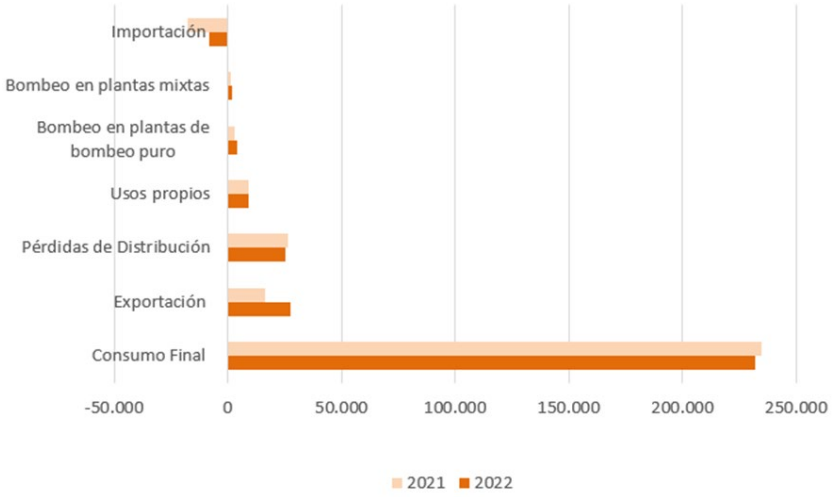


Figura 4: Demanda de la electricidad producida en España (GWh)
(Ministerio de la Transición Ecológica y el Reto Demográfico, 2023)

- **Operador del Mercado Ibérico de Energía (OMIE)**: es el operador de mercado eléctrico designado para la gestión del mercado diario e intradiario en la Península Ibérica (Operador del Mercado Ibérico de Energía, 2024).
- **Comisión Nacional de los Mercados y la Competencia (CNMC)**: es el regulador del sistema. Establece las normas para el funcionamiento, asegura que se cumplan y aplica sanciones cuando es necesario, manteniendo un entorno de competencia y equitativo (Jefatura de Estado, 2013, art.7)
- **El Estado Español**: tiene amplias competencias para regular el sistema eléctrico, incluyendo el establecimiento de los gravámenes a la generación y al consumo.
- **La Unión Europea**: regula las normas básicas del funcionamiento de los mercados europeos, del acceso a las redes y del mercado europeo de Derechos de Emisión de CO2 (Ministerio de la Transición Ecológica y el Reto Demográfico, 2024).

El mercado mayorista (spot o "al contado") está formado por un mercado diario, un mercado intradiario continuo y un mercado intradiario de subastas (Operador del Mercado Ibérico de Energía, 2024). España y Portugal comparten el mismo operador del

mercado y unas mismas reglas, de forma que resulta un mismo precio en cada hora del día (Mercado Ibérico de Electricidad, 2024).

Además del mercado eléctrico mayorista, se dispone de los mercados a plazo (de derivados o de futuros), donde las compras se realizan por anticipado a la entrega. Pueden ser organizados (OMIP y EEX) o no organizados (OTC) (Comisión Nacional de los Mercados y la Competencia, 2022). También se dispone de los contratos de compraventa a plazo entre generadores y a muy largo plazo entre un generador de energía renovable y un consumidor (Power Purchase Agreement o PPA) (Iberdrola, 2024).

Dentro del **mercado minorista**, el consumidor puede elegir entre el **mercado regulado** donde el precio varía cada hora reflejando el precio que resulta según la oferta y la demanda en el mercado eléctrico mayorista (Precio Voluntario para el Pequeño Consumidor o PVPC) y el **mercado libre** donde la comercializadora y los consumidores acuerdan los precios, que habitualmente son fijos.

Como se acaba de ver, el sector energético es complejo y cambiante. Los precios de la electricidad están sujetos a muchos condicionantes que pueden provocar fluctuaciones a lo largo del tiempo, tanto a largo plazo como a corto plazo. Esta variación en los precios afecta a los consumidores con diferente intensidad.

Para los consumidores domésticos la elección entre contratar en un mercado u otro puede suponer una gran diferencia. Para elegir la opción más ventajosa será necesarios conocimientos especializados en el tema y tiempo de estudio de las diferentes ofertas.

Para el resto de consumidores, dependiendo del sector en qué operen necesitarán mayor o menor cantidad de electricidad para su actividad. El precio de la electricidad puede ser un factor determinante en sus costes y, como consecuencia, en su competitividad.

1.4. Alcance

Este trabajo tiene un alcance limitado por factores de tiempo y de coste. El tiempo estimado para realizar el sistema completo excedería el plazo para realizar este trabajo. El despliegue del sistema requeriría de la adquisición de un hardware adecuado o la contratación de servicios en la nube para dar un servicio adecuado a los clientes. Además, la realización de un modelo de estimación competitivo necesitaría añadir datos que no se encuentran de manera abierta.

Por lo tanto, no se espera obtener un producto totalmente funcional sino un prototipo que pueda servir de base para llevar a cabo un producto comercial. Será un sistema *on-premises* en un hardware doméstico y con datos abiertos. En el subsistema de minería de datos se dejará margen para la optimización y ajuste del modelo de aprendizaje automático. La aplicación web ofrecerá unos servicios mínimos.

A pesar de las limitaciones, el prototipo estará construido sobre dos capas que posibilitarán su escalado en caso de ser necesario:

- Capa de contenedores, que podrán usar herramientas de orquestación como Kubernetes, ECS (Amazon Web Services, 2024), AKS (Microsoft, 2024) o GKE (Google, 2024).
- Capa de almacenamiento distribuido con el sistema de archivos de Hadoop (HDFS) y un clúster de nodos de Spark con Delta Lake para acelerar el procesamiento usando paralelización.

Se espera que el resultado de este trabajo facilite el desarrollo de sistemas análogos en la industria ya que el enfoque, la arquitectura y las tecnologías usadas se podrán aplicar a otros escenarios.

CAPÍTULO 2: ESTADO DE LA CUESTIÓN

2.1. Introducción

Hay dos aspectos que son más relevantes en este trabajo. Por un lado, el ámbito del problema que consiste en construir un sistema que realice estimaciones precisas de los precios de la electricidad en el mercado mayorista de España. Por otro lado, la arquitectura de software y tecnologías a emplear para el desarrollo del mismo.

A continuación, se muestra a un breve análisis de una selección de trabajos relacionados y seguidamente se expone cada aspecto.

2.2. Trabajos relacionados con la estimación de los precios de la electricidad

2.2.1. El mercado eléctrico mayorista español (Enseñat Saavedra, 2022)

Realiza un estudio detallado del funcionamiento y los factores que afectan a este mercado marginalista, donde el precio final es igual para todos los agentes premiando la eficiencia en la generación.

El conocimiento aportado ayudará a realizar un modelo predictivo que capture de forma más precisa la relación entre los datos de entrada aportados y los de precios estimados.
Además, construye varios modelos de regresión lineal múltiple que muestran resultados aceptables, aunque son demasiado simplificados. Se podrían mejorar aportando más variables de entrada, como las meteorológicas, la demanda de consumo o la situación política internacional.

2.2.2 Electricity price forecasting for MIBEL market: A machine learning approach (Messias & Rufino, 2021)

Esta tesis estudia la estimación de precios en el mismo mercado mayorista español utilizando varios algoritmos de aprendizaje automático. Aporta datos sobre el entrenamiento con los diferentes modelos y los errores conseguidos con las medidas estándar de validación. Aunque el número de variables de entrada es limitado y el rango temporal de los datos de entrada es sólo de 4 años, puede ser de gran ayuda a la hora de comparar sus resultados con los obtenidos en este trabajo.

2.2.3 Evaluation of machine learning models for predicting the system marginal price of an electricity system – italian SMP day-ahead forecasting (Dardamanis & Δαρδαμάνης, 2022)

Esta tesis de master, centrada en datos de Italia, aporta información detallada sobre el método usado desde la extracción de variables (*features*, en inglés), determinación de los datos de entrenamiento y validación, la definición y los resultados obtenidos de los algoritmos usados. Se han probado modelos basados en redes neuronales artificiales y modelos autorregresivos para series temporales tanto para estimaciones con un día de anticipación como 2 semanas.

Se espera que aporte gran valor gracias a la comparación de resultados.

2.2.4 Electricity consumption and temperature: Evidence from satellite data (Yao, 2021)

Este estudio encuentra la relación que hay en forma de U entre las dos variables aplicando métodos estadísticos. Confirma la importancia de usar la temperatura como variable de entrada en el modelo que se use en este trabajo.

2.2.5 Electricity Consumption Prediction using Energy Data, Socio-economic and Weather Indicators. A Case Study of Spain. (Grigoryan, 2021)

El trabajo de Grigoryan, centrado en la predicción del consumo eléctrico basado en técnicas estadísticas y de aprendizaje automático, confirma la necesidad de usar diferentes variables de entrada y una buena selección de las mismas. Sin embargo, no aporta mucho

valor a nivel comparativo ya que usa datos mensuales y no diarios como los que se espera utilizar en la aplicación web.

2.3. Trabajos relacionados las tecnologías a usar

2.3.1 Sistema para la extracción del conocimiento sobre los efectos adversos en las vacunas del COVID 19 (Castillo Martín, Julio 2021)

En este trabajo se desarrolla de forma detallada un proceso KDD para grandes volúmenes de datos. Aunque realiza una tarea descriptiva con un algoritmo de asociación en lugar de una tarea predictiva como sería el caso de un problema de estimación de precios, muestra que el uso del lenguaje Python con la librería Pandas es de gran utilidad para realizar el proceso de ETL.

2.3.2 Efficient Remote Memory for Parallel and Distributed Data Analytics (Thaker, 2022)

Esta tesis aporta gran valor porque el estudio presenta un framework para el procesado masivo de datos en paralelo desarrollado por la Universidad de Standford. Muestra una mejora en el rendimiento comparado con otras alternativas como Map Reduce, Spark o Dask. Sin embargo, presenta la desventaja de ser demasiado nuevo y poco probado frente a los otros que llevan siendo usados desde hace tiempo en la industria. Además, la documentación disponible es mucho menor dificultando su aprendizaje.

2.3.3. Performance evaluation of Map-reduce jar pig hive and spark with machine learning using big data (Jankatti et alt. 2020)

Este artículo aporta gran valor al estudiar el rendimiento de las diferentes tecnologías donde sobresale el uso de Spark con Hive.

2.3.4. An Enhanced Parallelisation Model for Performance Prediction of Apache Spark on a Multinode Hadoop Cluster (Ahmed et alt., 2021)

Aporta un estudio detallado en esta arquitectura enfatizando la complejidad de afinar los parámetros del cluster. Se consiguen buenos resultados, aunque los experimentos se realizan sobre tareas no predictivas y se excluyen fuentes procedentes de SQL o de *streaming*. Por lo tanto, no se podrá usar directamente para comparar con los resultados de este estudio.

2.3.5 Implementation and evaluation of a container management platform on docker: Hadoop deployment as an example. Cluster Computing (Shih et alt., 2021).

Este articulo estudia las ventajas de una arquitectura basada en un cluster de hadoop desplegada en contenedores Docker. Aporta gran valor por confirmar las ventajas que tiene esta arquitectura, como la simplificación y eficiencia en el despliegue, adaptación del tamaño y mantenimiento.

2.3.6. Employing Vertical Elasticity for Efficient Big Data Processing in Container-Based Cloud Environments (Jin-young et alt., 2021)

Este artículo realiza experimentos de procesado de grandes cantidades de datos a través de una red de contenedores Docker. En contraposición al escalado horizontal que ofrecen las máquinas virtuales (ampliando el número de máquinas cuando es necesario), los contenedores permiten un escalado vertical modificando la capacidad de almacenamiento y procesado de cada uno.

Este estudio aporta gran valor por su ayuda en la toma de decisión de la arquitectura a emplear.

2.4. Sistema de estimación de los precios de la electricidad

Este ámbito ha sido presentado en el capítulo 1. Aquí entraría en juego la búsqueda de los factores o variables que afectan a la fijación del precio. Estos van a nutrir el sistema, permitiendo que pueda generalizar de una forma más o menos precisa y que consiga resultados ajustados ante situaciones no vistas anteriormente por el sistema.

Entre estos factores tendremos los precios del gas natural y carbón, los derechos de emisión de CO2, el precio de la electricidad importada, la situación política, las regulaciones españolas y europeas, los impuestos, las subvenciones, las variaciones en la demanda y las variables meteorológicas que afectan a la producción de energías renovables como la velocidad del viento a 10 o 100 metros, precipitaciones, temperatura del aire, radicación solar en la superficie. Además, habrá que tener en cuenta el momento estudiado como el día de la semana o la estación del año que afectarán directamente a la demanda de consumo eléctrico.

2.5. Arquitectura y tecnologías a emplear para el desarrollo del sistema

2.5.1 Minería de datos

En esencia este trabajo tiene como objetivo desarrollar un producto que extraiga conocimiento valioso a partir de un gran volumen de datos. Estamos ante un problema de minería de datos (Lara, 2014).

Para resolverlo es importante seguir una metodología que estructure, organice y aplique procesos y técnicas. Esto permite aumentar la estandarización y control del proceso global, aplicar automatización y garantizar una reproductibilidad de los resultados.

En minería de datos se pueden destacar las siguientes metodologías:

- Metodología KDD (Knowledge Discovery in Databases). Propuesta por Fayyad et alt. (1996). Contiene una serie de fases con retroalimentación en los que se parte de unos datos y se obtiene conocimiento. El número de fases ha variado con el tiempo. En la figura 5 puede verse un modelo.

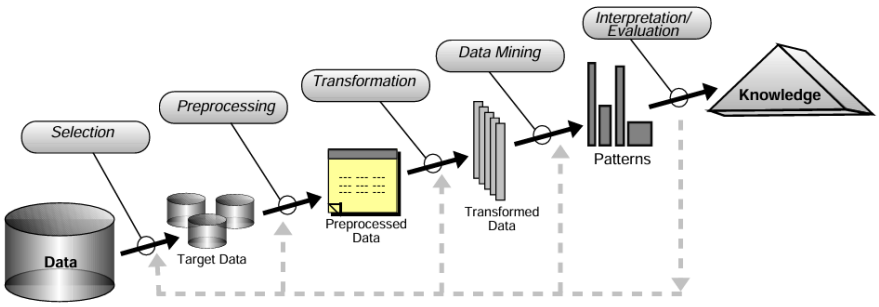


Figura 5: Fases del KDD (Fayyad et alt., 1996)

- CRISP-DM (CRoss-Industry Standard Process for Data Mining). Contiene 6 fases iterativas y con retroalimentación. Comienzan con el entendimiento del negocio, donde se busca entender el problema y los requisitos. Seguidamente se busca entender los datos y se preparan, ocupando la mayor parte del tiempo del proceso completo. Finalmente, siguiendo la figura 6, se realiza el modelado, se evalúa y se despliega.

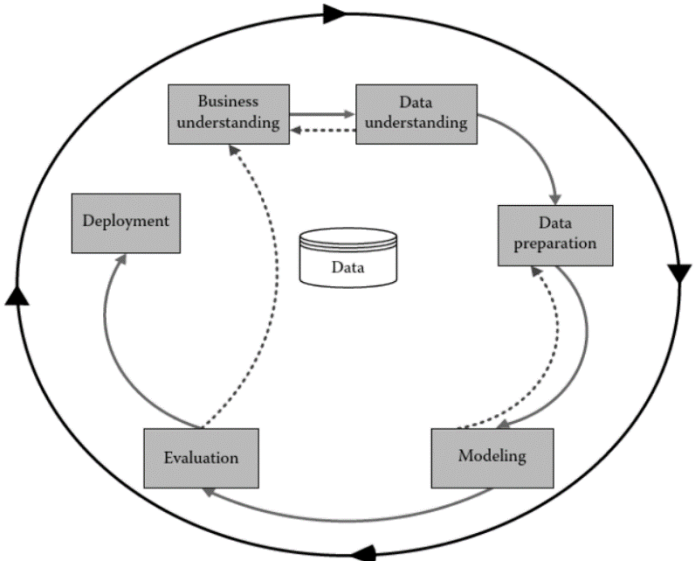


Figura 6: CRISP-DM (Osei-Bryson y Barclay, 2015)

- SEMMA. Centrada en los aspectos del desarrollo del modelo (Dean, 2014) tiene los siguientes pasos, como puede verse en la figura 7:

- **S**ample: Extraer una porción de los datos y particionarlos en los conjuntos de entrenamiento, validación y prueba.
- **E**xplore: Buscar tendencias y anomalías.
- **M**odify: Crear, seleccionar y transformar las variables que usará el modelo.
- **M**odel: Automatizar la búsqueda de predicciones sobre los datos.
- **A**ssess: Evaluar los resultados obtenidos.

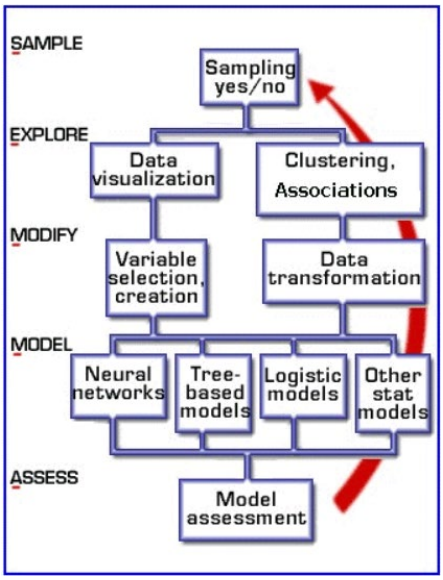


Figura 7: SEMMA (SAS Help Center, 2024)

- Proceso para minería de datos Six Sigma. Aplicación de filosofía de negocios Six Sigma como herramienta que busca la mejora continua de la calidad y eficiencia. Sigue el método DMAIC (Define, Measure, Analyze, Improve, Control). El proceso viene esquematizado en la figura 8.

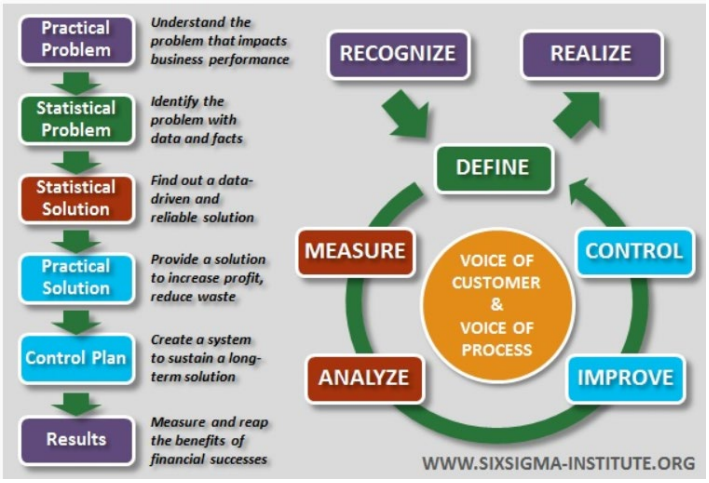


Figura 8: Proceso para minería de datos Six Sigma (Sixsigma Institute, 2024)

En este trabajo se va a seguir la metodología KDD por estar fuertemente ligada a la arquitectura del sistema y a las diferentes tecnologías a emplear. Además, se añadirá una fase previa de entendimiento del negocio como en CRISP-DM.

El sistema a desarrollar estará compuesto por las capas definidas desde el punto de vista de la minería de datos y que se muestran en la figura 8.

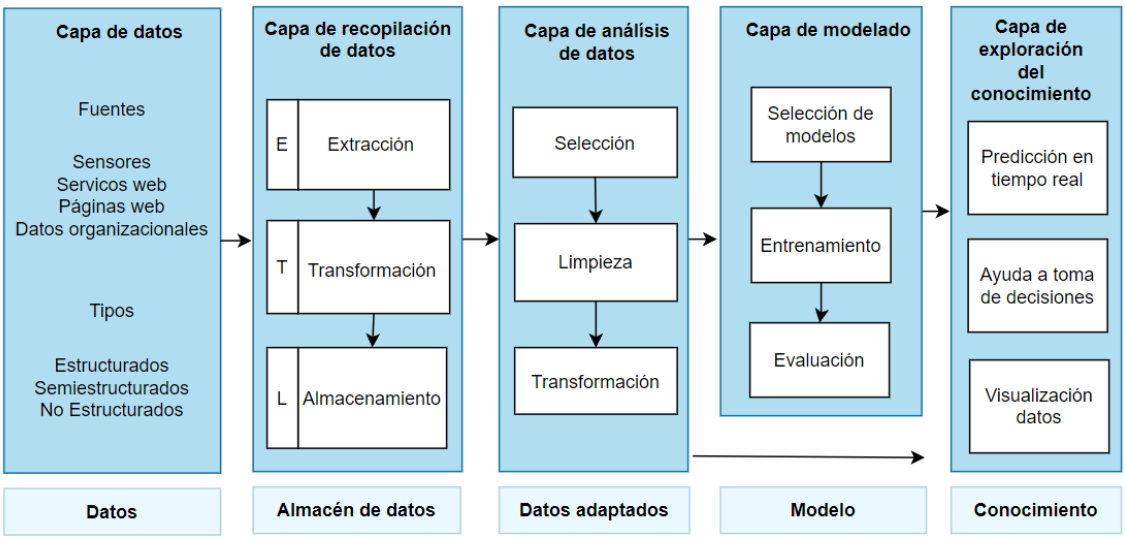


Figura 8: Capas y productos del sistema (Balusamy et alt., 2021)

2.5.2 Capa de datos

Esta capa conforma el origen de los datos. Existen distintas fuentes de datos como bases de datos relacionales, base de datos no relacionales, redes sociales, sensores, historiales médicos o transacciones económicas. En este caso se utilizarán fuentes de datos online como servicios web, contenido de páginas web y flujos de datos accedidos a través de APIs.

Los datos pueden ser clasificados en:

- Estructurados: siguen un formato rígido, pueden ser organizados en tablas y son almacenados en bases de datos relacionales, hojas de cálculo u otros archivos de texto que guardan un registro por línea como los csv.
- Semiestructurados: tienen una estructura definida pero no pueden ser almacenados en bases de datos relacionales. Entre estos tenemos los archivos con algún formato de marcado como XML y JSON.
- No estructurados: son datos complejos no organizados que no pueden ser almacenados en bases de datos relacionales. Constituye la mayor parte de la información que se genera en la actualidad como páginas web, pdfs, emails, video, audio o imágenes. Suelen venir en formato de texto o binario.

Muchos datos suelen venir acompañados de metadatos que proporcionan información adicional sobre la estructura de los datos, los procesos a los que se ha sometido o el origen de los mismos. Por ejemplo, si se descargan datos meteorológicos procedentes del programa Copérnico de la Unión Europea, éstos vienen acompañados de información muy relevante como las unidades de medida de cada variable, la resolución temporal o las coordenadas geográficas de su captura.

2.5.3 Capa de recopilación de datos

En esta capa se realiza la extracción y preprocesado de los datos para su almacenamiento inicial.

Los datos procederán de diferentes fuentes y tendrán diferentes formatos. La recopilación se puede realizar a través de consultas SQL a bases de datos, de llamadas a APIs conectadas a fuentes de datos, de extracción directa de páginas web (*web scraping*), de flujos de datos de sensores y de plataformas de intercambio de datos (como AWS DataExchange o Crunchbase) entre otros.

Una vez extraídos deberán ser transformados para que sea posible su almacenamiento en la forma requerida por el sistema.

Existen diferentes alternativas para realizar el procesado de datos:

- Hojas de cálculo como Microsoft Excel.
- Aplicaciones especializadas como Excel Power Query, Google DataPrep o IBM Watson Studio Refinery.
- Librerías específicas para algunos lenguajes de programación como Pandas para Python o Dplyr para R.
- Plataformas de procesamiento distribuido como Hadoop (MapReduce), Spark, Hive o Kafka de la fundación de software Apache (Apache Software Foundation, 2024)

En este trabajo se utilizarán las herramientas para el procesado distribuido por ofrecer mayor velocidad de proceso, escalamiento horizontal, mejora en la tolerancia a fallos y capacidad de manejar grandes volúmenes de datos.

Apache Spark permite el procesado de datos en memoria, a diferencia de Pandas o Hadoop MapReduce que trabajan en disco. Utiliza el tipo de datos RDD (Resilient Distributed Dataset) que es muy eficiente, aunque no permite su modificación una vez creado. Además, optimiza los procesos de ejecución con un planificador de grafos (DAG Scheluder). Eso le confiere rendimiento muy superior al resto de alternativas (Chambers y Zaharia, 2018).

Spark está compuesto por un proceso que ejecuta la aplicación del usuario (*Driver Program*), una serie de nodos ejecutores (*Workers*) que procesan los datos e interactúan con el sistema de almacenamiento y un gestor del cluster (*Cluster Manager*) que planifica las tareas y las distribuye entre los distintos nodos. En la figura 9 puede verse el esquema.

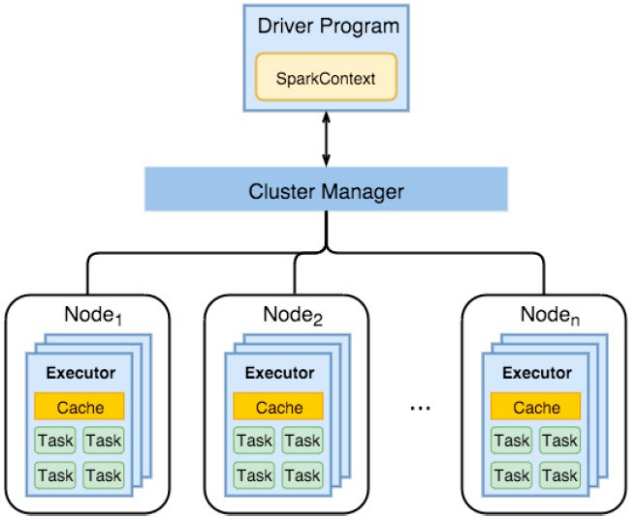


Figura 9: Componentes de Spark (Feng, 2021)

Tras la transformación de los datos se procede a su almacenamiento. Existen distintas opciones de almacenamiento:

- Ficheros en almacenamiento local o distribuido. El sistema de almacenamiento distribuido de Hadoop (HDFS) almacena los archivos a través de un *cluster* de nodos. Aunque un archivo esté físicamente repartido en varios nodos, el cliente accederá al mismo como si lo tuviera de forma local en su máquina. Este sistema

permite el almacenamiento de archivos de gran tamaño, el procesado en paralelo de los datos y la replicación de los bloques en los que se descomponen cada archivo haciéndolo robusto ante posibles pérdidas (White, 2015). Tiene una arquitectura maestro/esclavo, donde el maestro (Namenode) gestiona el sistema de ficheros y su acceso y los esclavos (Datanodes) almacenan los archivos. Los archivos son divididos en bloques, repartidos y replicados entre los distintos nodos. En la figura 10 se muestra un esquema de la arquitectura.

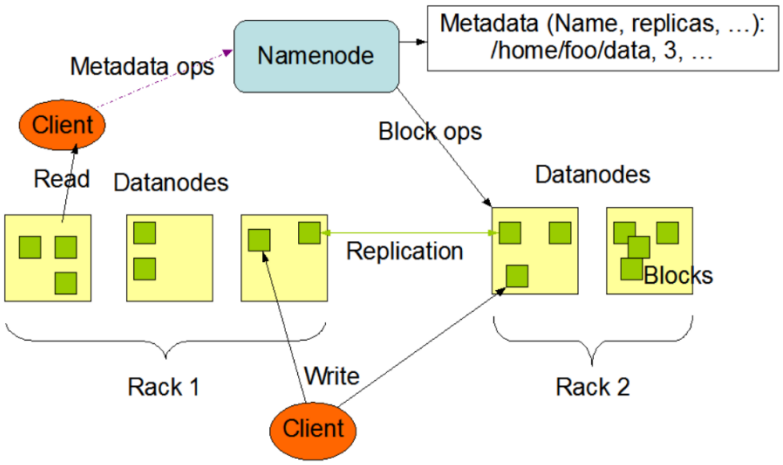


Figura 10: Arquitectura de HDFS (Apache Software Foundation, 2024)

- Bases de datos relacionales: trabajan con tablas y las relaciones entre ellas, minimizan la redundancia de datos, facilitan la copia de seguridad y recuperación ante desastres y las transacciones cumplen las propiedades ACID (atomicidad, consistencia, aislamiento y durabilidad). Sin embargo, sólo admiten datos estructurados. Algunos ejemplos comerciales y de código abierto son IBM DB2, Microsfot SQL Server, MySQL, PostgreSQL y Oracle Database.
- Bases de datos NoSql (no sólo SQL): permiten esquemas flexibles, pueden manejar todo tipo de datos, escalan eficientemente en sistemas distribuidos, pueden estar basadas en pares llave-valor, en documentos, en columnas o en grafos (Santos, 2020). Algunos ejemplos son DynamoDB, MongoDB, Cassandra y Neo4J.

- Almacén de datos (*data warehouse*): almacenan datos estructurados y semiestructurados, se adaptan bien al aumento de datos escalando horizontalmente, se suelen usar para analizar datos históricos. Algunos ejemplos de versiones en la nube son Oracle Exadata, Google BigQuery y Amazon Redshift.
- Lagos de datos (*data lakes*): almacenan todo tipo de datos en su formato original sin requerir un esquema permitiendo una carga muy rápida. Los datos se transformarán cuando sea necesario usarlos. Como ejemplos tenemos Amazon S3 Data Lake, Microsoft Azure Data Lake, Cloudera Data Lake y Snowflake.
- Data lakehouse: combina las características de un Data Lake (capacidad para almacenar datos sin estructurarlos previamente) y un Data Warehouse (optimizado para consultas SQL y análisis empresarial). Algunos ejemplos son Databricks, Apache Hive LLAP, Microsoft Azure Synapse Analytics y Amazon S3 Lakehouse. Además, se puede utilizar la tecnología Delta Lake para implementar la arquitectura data lakehouse. Tiene un formato de almacenamiento distribuido compatible con Apache Spark (Lee et alt., 2022).

En este trabajo se va a utilizar HDFS para almacenar y procesar los datos extraídos y una base de datos relacional que recogerá la información que deba ser mostrada en la aplicación web y otra información de gestión, como los usuarios del sistema.

2.5.4 Capa de análisis de datos

En esta capa se analizan los datos y se transforman para que se adapten:

- Al modelo que se vaya usar para realizar las estimaciones
- A la información que se mostrará al usuario en la ampliación web.

Es necesario realizar primero una exploración de los datos. Se visualizan los valores almacenados, el tipo de dato de cada atributo, los valores estadísticos (valores máximos,

mínimos, medias, desviaciones estándar, percentiles, etc.) y su representación gráfica si fuera preciso. De esta forma se puede analizar su estructura, contenido y sus interrelaciones.

Se realiza después una selección de datos para eliminar los que no sean relevantes, a través de un filtrado de atributos y de registros. Cuando no es necesario utilizar la totalidad de los registros se realiza un muestreo escogiendo un subconjunto de ellos.
Seguidamente, se realiza una limpieza de datos para evitar que haya registros que les falten valores o sean erróneos. Para ambos casos podemos seguir alguna de las siguientes estrategias:

- Ignorar el valor y continuar con el análisis.
- Filtrar toda la columna donde se encuentre el valor.
- Filtrar el registro donde se encuentre el valor.
- Asignar un valor correcto como puede ser la media.

Por último, se transforman los datos para adaptarse a los requerimientos del sistema. Algunas técnicas que se aplican son la cuantización, la discretización, la creación de atributos, la normalización y la reducción de dimensionalidad (Lara, 2014).

En esta capa se usarán las mismas tecnologías que en la anterior relativas al procesado y almacenamiento de los datos.

2.5.5 Capa de modelado

En esta capa se pretende crear un modelo que realice una de estas tareas:

- Descripción de los datos, principalmente buscando la creación de grupos homogéneos (*clustering*), relaciones ente los atributos (asociación) o valores atípicos.

- Predicción de valores desconocidos realizando tareas de clasificación de un ejemplo indicando a qué clase pertenece o de regresión dando un valor cuantitativo estimado.

Este trabajo se va a centrar en crear un modelo de regresión que estime los valores futuros del precio de la electricidad.

Para realizar estas tareas se emplean algunos algoritmos de aprendizaje automático (*machine learning*). Estos pueden clasificarse en:

- Supervisados: los datos de entrenamiento incluyen la solución o etiqueta.
- No supervisados: los datos de entrenamiento no están etiquetados.
- Semisupervisados: usualmente unos pocos registros están etiquetados y el resto no.

En la tabla 1 se muestran algunos ejemplos:

		Tarea					
Tipo	Algoritmo	Clustering	Asociación	Detección de atípicos	Visualización y reducción de dimensionalidad	Clasificación	Regresión
Supervisados	k vecinos más próximos (KNN)	X		X		X	X
	Regresión lineal						X
	Regresión no lineal						X
	Suport Vector Machines (SVMs)					X	X
	Redes neuronales artificiales	X				X	X
	Árboles de decisión					X	
	Random forests					X	
	Naive Bayes					X	
No supervisados	K-medias	X		X		X	X
	Análisis de clusters jerárquicos	X					
	Maximización de las espectativas	X					
	AGNES	X					
	DIANA	X					
	DBSCAN	X		X			
	STING	X					
	Análisis de componentes principales (PCA)				X		
	Kernel PCA				X		
	Locally-Linear Embedding (LLE)				X		
	t-distributed Stochastic Neighbor Embedding (t-SNE)				X		
	A priori		X				
	Eclat		X				
Semisupervisados	Deep Belief Networks (DBNs)					X	
	Restricted Boltzmann Machines (RNMs)					X	

Tabla 1: Algoritmos de aprendizaje automático relacionados con la minería de datos (elaboración propia)

2.5.6 Capa de exploración del conocimiento

En esta capa se muestran los resultados obtenidos de las capas de análisis y modelado a través de un interfaz de usuario. Permitirá analizar respecto del pasado qué ha ocurrido y realizar un diagnóstico de por qué. Respecto del futuro, aportará un escenario de qué puede ocurrir y puede sugerir qué se debería hacer (Balusamy, 2021).

En este trabajo se mostrarán tanto datos pasados como previsiones futuras a través de una página web.

El desarrollo de la aplicación web pueden realizarse desde cero usando html, css y javascript. También pueden emplearse *frameworks* que aportan una serie de ventajas como la disminución de errores, aumento de la seguridad y reducción del tiempo y coste de desarrollo. En la siguiente tabla se muestran los más utilizados en la actualidad:

Framework	Creado por	Lenguaje	FullStack	Comentarios
ASP.NET Core	Microsoft	C#	Sí	Adecuado para la creación de API RESTful
React	Facebook	JavaScript	Front-end	Uso de componentes reutilizables
Django	Lawrence Journal-World	Python	Sí	Acceso fácil a bases de datos, gran seguridad
Angular	Google	TypeScript	Sí	Uso de componentes reutilizables
Flask	Pocoo	Python	Sí	Ligero. Desarrollo rápido de aplicaciones
Ruby on Rails (Rails)	David Heinemeier	Ruby	Back-end	Enfocado en simplificar el desarrollo
Vue.js	Evan You	JavaScript	Front-end	Crea aplicaciones en una sola página (SPA)
Laravel	Taylor Otwell	PHP	Back-end	Código muy legible

Tabla 2: Frameworks para el desarrollo de aplicaciones web (elaboración propia)

2.5.7 Lenguajes de programación en minería de datos

Para la minería de datos se pueden usar diferentes lenguajes de programación como C++, Java, R o Python. Aquí se ha optado por utilizar Python ya que es ampliamente usando en minería datos por su facilidad de aprendizaje y la gran cantidad de librerías y *frameworks* disponibles de forma abierta para todo tipo de tareas. Aunque Python es más

lento que otros lenguajes (Kumar & Goswami, 2023), las librerías que se usan para el procesado de datos como ScikitLearn o TensorFlow están escritas en el lenguaje C++ con lo que puede obtenerse el mayor rendimiento.

Para extraer datos de base de datos se emplea mayoritariamente el lenguaje de consulta SQL. Es muy eficiente y tiene una sintaxis muy sencilla y estandarizada.

Por último, los lenguajes *shell* (como Shell de Linux y PowerShell de Windows) se utilizan como complemento para automatizar ciertas tareas como arrancar algunos programas. Son una capa externa para acceder a funciones de los sistemas operativos como el manejo de archivos o gestión de permisos.

2.5.8 Virtualización para el desarrollo y despliegue de aplicaciones

La virtualización consiste en crear un entorno de computación independiente y aislado de la máquina física donde se aloje. Se logra mediante el uso de un software que abstrae los recursos de hardware. Permite la gestión eficiente de los recursos, aumentar la seguridad y facilitar el compartir un software desarrollado para un entorno específico (Herrera-Izquierdo y Grob, 2017).

Hay dos enfoques:

- La virtualización completa que crea una máquina virtual completa con su propio sistema operativo, memoria y otros recursos. VirtualBox y VMware son ejemplos de esta.
- La virtualización por contenedores que encapsula una aplicación y sus dependencias dentro de un contenedor. Los contenedores comparten el sistema operativo del anfitrión y sus recursos. Son más ligeros y más eficientes en la gestión de los recursos (consumen sólo la memoria RAM necesaria y son más rápidos al arrancar, en la ejecución y apagar). Ejemplos de este enfoque son

Docker, RKT y Podman. Además, existen herramientas para la gestión (orquestación) de contenedores como Kubernetes, Amazon Elastic Container Service (ECS) y Docker Swarm.

En este trabajo se va a la utilizar la virtualización con Docker para facilitar la escalabilidad del sistema. Si es necesario añadir nodos a un cluster de Hadoop o de Spark bastaría con desplegar más contenedores de los nodos.

CAPÍTULO 3: METODOLOGÍA, PLANIFICACIÓN Y COSTES

3.1. Metodología

Para llevar a cabo el desarrollo del sistema de software se ha decidido seguir el estándar de desarrollo de software IEEE 1074/1997 (Instituto de Ingenieros Eléctricos y Electrónicos, 1997) con el objetivo de alcanzar la mayor calidad tanto del producto como del proceso de desarrollo. Se ha preferido frente al estándar ISO/IEC/IEEE 12207 (Organización Internacional de Normalización, 2017) por ser más reducido y estar limitado a actividades de software (excluye las relacionadas con la contratación, compra o fabricación de hardware). Además, este estándar no está ligado a ningún Modelo de Ciclo de Vida de Software y puede valer tanto para sistemas completos como para los que son parte de sistemas más grandes.

Este estándar está destinado a las empresas que se dedican a la gestión y desarrollo de proyectos de software. Será función de la figura del arquitecto de procesos el preparar el proceso del ciclo de vida del software.

En esta versión del estándar se incluyen 65 actividades que pueden dividirse en los siguientes grupos: Gestión de Proyectos, Pre-Desarrollo, Desarrollo, Post-Desarrollo y Actividades Integrales.

Una actividad es un conjunto definido de trabajo que describe la transformación requerida de las entradas de información en salidas. No se completa hasta que todas las entradas se han procesado y todas las salidas se han generado (Instituto de Ingenieros Eléctricos y Electrónicos, 1997). En la tabla 3 se muestran las actividades que componen este estándar.

Grupos de atividades	Actividades
A.1 Grupos de Actividades de Gestión de Proyectos	
A.1.1 Actividades de Iniciación del Proyecto	A.1.1.1 Crear SLCP (Plan de Ciclo de Vida del Software)
	A.1.1.2 Realizar Estimaciones
	A.1.1.3 Asignar Recursos del Proyecto
	A.1.1.4 Definir Métricas
A.1.2 Actividades de Planificación del Proyecto	A.1.2.1 Planificar Evaluaciones
	A.1.2.2 Planificar la Gestión de la Configuración
	A.1.2.3 Planificar la Transición del Sistema
	A.1.2.4 Planificar la Instalación
	A.1.2.5 Planificar la Documentación
	A.1.2.6 Planificar la Capacitación
	A.1.2.7 Planificar la Gestión del Proyecto
	A.1.2.8 Planificar la Integración
A.1.3 Actividades de Monitorización y Control del Proyecto	A.1.3.1 Gestionar Riesgos
	A.1.3.2 Gestionar el Proyecto
	A.1.3.3 Identificar las Necesidades de Mejora del SLCP
	A.1.3.4 Conservar Registros
	A.1.3.5 Recopilar y Analizar Datos Métricos
A.2 Grupos de Actividades Pre-Desarrollo	
A.2.1 Actividades de Exploración de Conceptos	A.2.1.1 Identificar Ideas o Necesidades
	A.2.1.2 Formular Enfoques Potenciales
	A.2.1.3 Realizar Estudios de Viabilidad
	A.2.1.4 Refinar y Finalizar la Idea o Necesidad
A.2.2 Actividades de Asignación del Sistema	A.2.2.1 Analizar Funciones
	A.2.2.2 Desarrollar la Arquitectura del Sistema
	A.2.2.3 Descomponer los Requisitos del Sistema
A.2.3 Actividades de Importación de Software	A.2.3.1 Identificar Requisitos de Software Importado
	A.2.3.2 Evaluar Fuentes de Importación de Software
	A.2.3.3 Definir el Método de Importación de Software
	A.2.3.4 Importar Software

Grupos de atividades	Actividades
A.3 Grupos de Actividades de Desarrollo	
A.3.1 Actividades de Requisitos	A.3.1.1 Definir y Desarrollar Requisitos de Software
	A.3.1.2 Definir Requisitos de Interfaz
	A.3.1.3 Priorizar e Integrar Requisitos de Software
A.3.2 Actividades de Diseño	A.3.2.1 Realizar el Diseño Arquitectónico
	A.3.2.2 Diseñar la Base de Datos (Si es aplicable)
	A.3.2.3 Diseñar las Interfaces
	A.3.3.4 Realizar el Diseño Detallado
A.3.3 Actividades de Implementación	A.3.3.1 Crear Código Ejecutable
	A.3.3.2 Crear Documentación Operativa
	A.3.3.3 Realizar la Integración
A.4 Grupos de Actividades Post-Desarrollo	
A.4.1 Actividades de Instalación	A.4.1.1 Distribuir Software
	A.4.1.2 Instalar el Software
	A.4.1.3 Aceptar el Software en el Entorno Operativo
A.4.2 Actividades de Operación y Soporte	A.4.2.1 Operar el Sistema
	A.4.2.2 Proporcionar Asistencia Técnica y Consultoría
	A.4.2.3 Mantener el Registro de Solicitudes de Soporte
A.4.3 Actividades de Mantenimiento	A.4.3.1 Identificar las Necesidades de Mejora del SW
	A.4.3.2 Implementar el Método de Reporte de Problemas
	A.4.3.3 Reaplicar el SLCP (Software Lifecycle Process)
A.4.4 Actividades de Retiro	A.4.4.1 Notificar al Usuario
	A.4.4.2 Realizar Operaciones Paralelas
	A.4.4.3 Retirar el Sistema
A.5 Grupos de Actividades Integrales	
A.5.1 Actividades de Evaluación	A.5.1.1 Realizar Revisiones
	A.5.1.2 Crear una Matriz de Trazabilidad
	A.5.1.3 Realizar Auditorías
	A.5.1.4 Desarrollar Procedimientos de Pruebas
	A.5.1.5 Crear Datos de Prueba
	A.5.1.6 Ejecutar Pruebas
	A.5.1.7 Informar los Resultados de la Evaluación
A.5.2 Actividades de Gestión de la Configuración de Software	A.5.2.1 Desarrollar la Identificación de la Configuración
	A.5.2.2 Realizar el Control de la Configuración
	A.5.2.3 Realizar la Contabilidad de Estado
A.5.3 Actividades de Desarrollo de Documentación	A.5.3.1 Implementar la Documentación
	A.5.3.2 Producir y Distribuir la Documentación
A.5.4 Actividades de Capacitación	A.5.4.1 Desarrollar Materiales de Capacitación
	A.5.4.2 Validar el Programa de Capacitación
	A.5.4.3 Implementar el Programa de Capacitación

Tabla 3: Actividades del estándar IEEE 1074/1997 (Instituto de Ingenieros Eléctricos y Electrónicos, 1997)

Dado el limitado alcance de este proyecto, un cierto número de actividades del estándar no serán aplicables como, por ejemplo:

- Las actividades de Importación de Software
- Las actividades Post-Desarrollo: instalación, soporte, mantenimiento y retiro.
- Las actividades de Capacitación

La primera actividad de esta metodología consiste en escoger el Modelo de Ciclo de Vida de Software. Entre las opciones disponibles se tiene el Modelo en Cascada, el Modelo en Espiral, el Modelo en V, el Desarrollo Rápido de Aplicaciones (RAD), el Proceso Unificado de Desarrollo (RUP), la Programación Extrema (XP) y Scrum, entre otros.

Se ha escogido el modelo en cascada. Este modelo es secuencial e iterativo. Es fácil de implementar y entender. Sin embargo, es muy rígido ante cambios de requisitos y se entrega el producto al final por lo que es más costoso corregir los errores. Por otro lado, hasta que no se termina una etapa no se puede empezar la siguiente. Se ha elegido porque se espera que los requisitos van a estar bastante definidos desde el principio y porque no se van a realizar sucesivas entregas del producto dado el corto tiempo disponible para su ejecución.

3.2. Planificación

Este trabajo se realizará entre el 5 de marzo de 2024 y el 24 de junio de 2024. Se espera dedicarle 100 días y realizar un esfuerzo total de 300 horas. Para calcular el esfuerzo necesario se ha dividido el tiempo total en 4 fases que finalizan con el envío de cada uno de los entregables exigidos. En la tabla 4 se muestra el resultado de la planificación global de este Trabajo Fin de Grado (TFG).

Actividades	(horas)
Fase 1	
Estudio de los requisitos del TFG	3,0
Investigación sobre el mercado eléctrico español	6,0
Análisis de los objetivos y alcance del trabajo	2,0
Investigación sobre trabajos relacionados	6,0
Investigación sobre procedimientos y tecnologías	7,0
Análisis de tiempos y costes	2,0
Actividades de iniciación del proyecto de desarrollo	1,5
Actividades de planificación del proyecto de desarrollo	0,9
Actividades de Pre-Desarrollo del proyecto	7,6
Actividades trasversales del proyecto de desarrollo	6,0
Redacción de los capítulos 1, 2 y 3	20,0
Envío Entrega 1 del TFG	0,2
Fase 2	
Inicio actividades de Desarrollo del proyecto	70,0
Actividades trasversales del proyecto de desarrollo	6,0
Redacción del capítulo 4	10,0
Envío Entrega 2 del TFG	0,2
Fase 3	
Fin de actividades de Desarrollo del proyecto	74,0
Pruebas del sistema	9,0
Actividades trasversales del proyecto de desarrollo	9,0
Redacción del resto de capítulos y revisión completa	20,0
Envío Entrega 3 del TFG	0,2
Fase 4	
Revisión del TFG	39,0
Envío Entrega Final del TFG	0,4
Subtotal actividades exclusivas del TFG	116,0
Subtotal actividades del proyecto de desarrollo	184,0
Total	**300,0**

Tabla 4: Planificación del TFG (elaboración propia)

Para la elaboración de las estimaciones anteriores se ha realizado un análisis detallado del esfuerzo necesario para el proyecto de desarrollo. El proyecto se desarrollará durante 60 días en el periodo comprendido entre el 20 de marzo de 2024 y el 31 de mayo de 2024. Para las actividades que se ejecutan a lo largo de todo el proyecto pero que requieren poca dedicación diaria se ha calculado el total de los tiempos medios esperados por día. Partiendo de las estimaciones anteriores y teniendo en cuenta las restricciones de cada actividad, se ha elaborado el diagrama de Gantt que se muestra en la figura 11.

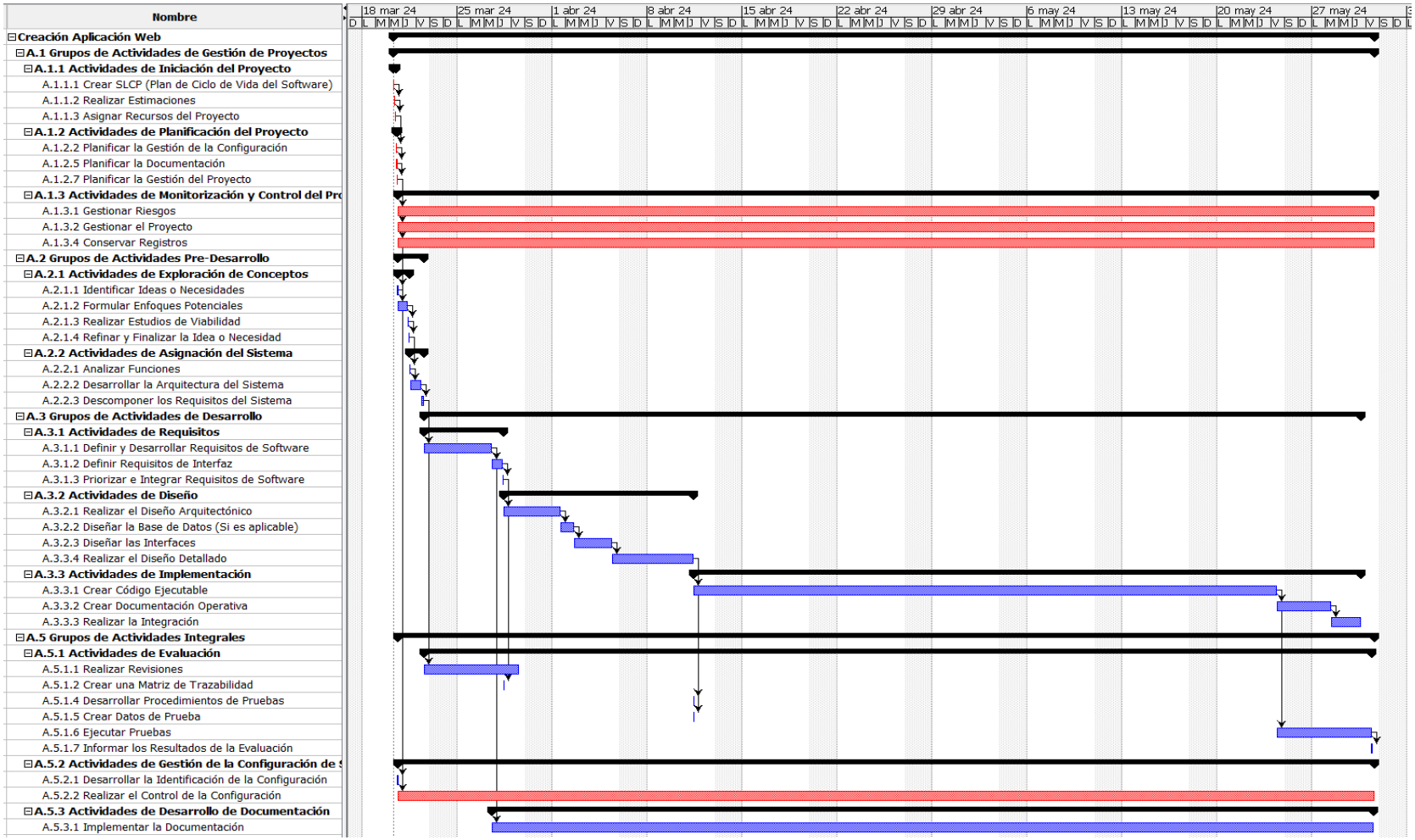


Figura 11: Diagrama de Gantt (elaboración propia)

La monitorización del avance de las actividades y el ajuste de este diagrama ayudará a gestionar adecuadamente el proyecto para que se acabe en el plazo fijado.

3.3. Costes

Para la estimación de los costes de este proyecto se van a diferenciar dos etapas: la etapa que termina con la puesta en producción del sistema (desarrollo y despliegue) y la etapa de producción del producto.

Respecto al primera etapa, únicamente se van a soportar costes por la mano de obra. No va a ser necesario invertir en equipos ni en licencias de software (se va a utilizar software gratuito para todo el proceso).

A partir de los costes medios salariales por trabajador y hora de publicados por el Instituto Nacional de Estadística en los últimos 3 años relativos a la división 62 de CNAE (programación, consultoría y otras actividades relacionadas con la informática) se han obtenido los datos de la figura 12.

Trimestre	Coste (€/hora)
2020T1	24,58
2020T2	26,37
2020T3	28,36
2020T4	28,65
2021T1	26,26
2021T2	27,22
2021T3	28,86
2021T4	29,84
2022T1	25,83
2022T2	27,55
2022T3	29,64
2022T4	29,91
2023T1	28,25
2023T2	29,00
2023T3	32,34
2023T4	33,07

Figura 12: Coste laboral total por hora del personal IT en España (www.ine.es, 2024)

El coste medio de los últimos tres años ha sido de 28,48 euros/hora. Siguiendo la técnica estadística ponderada para la estimación del coste (Rubio, 2021), se puede realizar la siguiente estimación:

$$Coste\ estimado = \frac{Mejor\ estimación + 4 \cdot Estimación\ promedio + Peor\ estimación}{6}$$
$$= \frac{24{,}58 + 4 \cdot 28{,}48 + 33{,}07}{6} = 28{,}60\ euros/hora$$

A partir de las estimaciones de esfuerzo de las actividades de esta etapa se ha obtenido la estimación del coste total de desarrollo y despliegue del sistema mostrado en la tabla 5.

Etapa	Horas	Coste (€/hora)	Coste total
Desarrollo	184	28,60	5.262,40 €
Despliegue	30	28,60	858,00 €
Total	214		**6.120,40 €**

Tabla 5: Costes de desarrollo y despliegue del producto (elaboración propia)

En cuanto a la estapa de producción, además del coste de mantenimiento se ha de contar con los costes relativos a la contratación de servicios en la nube donde se alojaría la aplicación.

La oferta de uso de contenedores en la nube es muy amplia. Se han tenido en cuenta tres proveedores de referencia en el mercado (Amazon, Microsoft y Google). Supondiendo una contratación de pago por uso y un tipo de cambio de 0.92 euros/dolar, se obtiene la comparativa de precios de la tabla 6.

Empresa	Servicio	vCPUs	Memoria (GiB)	Tarifa por hora (€)	Fuente
Amazon AWS	Amazon EC2	2	8	0,116	https://aws.amazon.com/
		8	32	0,464	
Microsoft Azure	Instancias contenedor	2	8	0,138	https://azure.microsoft.com/
		4	16	0,267	
Google Cloud	Kubernetes Engine	2	8	0,214	https://cloud.google.com/
		8	32	0,576	

Tabla 6: Comparativa de tarifas de servicios en la nube (elaboración propia)

Teniendo en cuenta dos escenarios, uno de demanda baja y otro de demanda alta de este producto, obtenemos el coste medio estimado en base a las tarifas anteriores. Y suponiendo una media de 730 horas al mes de uso y la necesidad de contratar un experto que realice el mantenimiento del sistema, se puede realizar una estimación total del coste del sistema en producción como el mostrado en la tabla 7.

		Demanda	
Tipo de coste	**Concepto**	**Baja**	**Alta**
Servicio contenedores en la nube	Coste por hora (eur/hora)	0,156	0,436
	Nº horas al mes	730	730
	Nº Contenedores	12	30
	Coste total	**1.366,57 €**	**9.537,73 €**
Personal IT	Nº horas al día	4	8
	Nº días al mes	22	22
	Coste hora (eur)	28,6	28,6
	Coste total	**2.516,80 €**	**5.033,60 €**
Coste total mensual		**3.883,37 €**	**14.571,33 €**

Tabla 7: Coste mensual del sistema en producción ante los escenarios de baja y alta demanda (elaboración propia)

Por último, habrá que tener en cuenta que ante un escenario de alta demanda sería conveniente buscar otro tipo de contrato con el proveedor de servicios en la nube, como, por ejemplo, acordando una reducción de precios en base a asegurar una permanencia o servicios mínimos contratados.

CAPÍTULO 4: DESARROLLO

4.1. Introducción

En el desarrollo de un nuevo producto de software el cliente es una pieza clave para el establecimiento y validación de los requisitos, es decir, para saber qué producto se necesita y si el producto obtenido cumple lo requerido. Dado que el presente trabajo se desarrolla en el ámbito académico, no se dispone de un cliente con el que poder interactuar. Para salvar este inconveniente se ha decidido crear un escenario ficticio donde se ha encontrado un cliente interesado en este producto. Este escenario servirá para construir el producto sobre una base más sólida sobre la que asentar el resto de fases de desarrollo. De esta forma se espera alcanzar los objetivos fijados en este trabajo.

El objeto de este trabajo es el desarrollo de un prototipo de aplicación web que estime los precios de la electricidad en España y que ofrezca sus servicios de forma estandarizada a consumidores domésticos y empresas. Se ha pensado en comenzar con un primer cliente que ayude a poner en marcha el producto y paulatinamente ir madurando el mismo para atraer nuevos clientes.

Entre los posibles clientes se ha preferido la búsqueda de uno que tenga los suficientes medios para ayudar en el proceso de ingeniería de requisitos pero que no tenga tantos como para disponer de su propio equipo de desarrollo de software. Además, debe ser una empresa donde los costes de energía y su gestión sean muy relevantes.

El cliente ficticio elegido se trata de una pequeña empresa que se dedica a ofrecer los siguientes servicios en la nube:

- IaaS, Infraestructura como servicio (IaaS): recursos de red y almacenamiento
- PaaS, Plataforma como servicio: plataforma donde ejecutar aplicaciones.
- SaaS, Software como servicio: aplicaciones con su plataforma e infraestructura

- FaaS, Función como servicio: aplicaciones basadas en eventos.

Debido a la naturaleza de estos servicios, el cliente tiene una dependencia muy grande de la energía. Como tiene la infraestructura repartida por diferentes zonas geográficas, puede repartir la carga de trabajo a conveniencia entre las diferentes zonas. La aplicación le ayudaría a gestionar adecuadamente sus costes de energía.

En los siguientes apartados se describen las actividades realizadas relativas al desarrollo siguiendo el estándar elegido IEEE 1074/1997. Salvo algunas actividades trasversales, la mayor parte de las mismas se realizarán de una manera secuencial siguiendo el modelo en cascada.

4.2. Grupos de Actividades de Gestión de Proyectos

4.2.1 Actividades de Iniciación del Proyecto

Se comienza con la creación del Plan de Ciclo de Vida del Software. En esta primera actividad se ha realizado un análisis de las actividades del estándar y se ha determinado cuales no se realizarán. Se ha estudiado los distintos modelos de ciclo de vida de software y, como se explicó en el apartado 3.1, se ha escogido el modelo en cascada. Este modelo consta de las fases de análisis de requisitos, diseño, implementación, pruebas, integración y mantenimiento. Se ha considerado que el ingeniero de software tiene los conocimientos y está capacitado para seguirlo. Se han identificado los atributos y las restricciones que se aplican al sistema final deseado y al entorno de desarrollo. Es decir, en qué va a consistir el producto, cómo se va a desarrollar y plazo del que se dispone. Esto fue explicado en los apartados anteriores.

Se continúa realizando las estimaciones. Para ello se ha identificado el trabajo que se deberá realizar y se ha descompuesto en distintas tareas. Seguidamente se ha realizado

una estimación del esfuerzo y del coste durante todo el ciclo de vida del software. Se ha elaborado el diagrama de Gantt expuesto en el apartado 3.2.

Finalmente, se han asignado los recursos al proyecto mostrados en la tabla 8.

Código	Descripción	Mínimo estimado	Máximo estimado
PRE	Presupuesto Global	0 EUR	200 EUR
MDO	Mano de obra	180 HORAS	210 HORAS
HW1	Equipos de sobremesa	1 UD.	1 UD.
HW2	Equipos portátiles	0 UD.	1 UD.
HW3	Memoria RAM adicional	0 GB.	64 GB.
HW4	Almacenamiento adicional en disco SSD	0 GB.	1 TB.
SW1	Sistemas Operativos	0 UD.	1 UD.

Tabla 8: Estimación inicial de requisitos (elaboración propia)

4.2.2 Actividades de Planificación del Proyecto

En la planificación de la configuración se ha decidido el uso de la herramienta Git (Git, 2024) y la plataforma GitHub (GitHub, 2024). El uso de estas herramientas permite llevar un control de versiones tanto del código, de la documentación, de las especificaciones y otros productos del trabajo. Permite crear distintas ramas y facilita el compartir el proyecto.

En la planificación de la documentación se han identificado el conjunto de documentos requeridos teniendo en cuenta el software a implantar, la audiencia y el propósito de los mismos (instruccional y de referencia).

Para la gestión del proyecto se ha planificado un seguimiento diario que permita identificar desviaciones de lo planificado y tomar medidas correctivas lo antes posible.

4.2.3 Actividades de Monitorización y Control del Proyecto

Estas actividades se realizarán a lo largo de todo el proyecto.

En cuanto a la gestión del riesgo se ha seguido la Metodología MAGERIT (Gobierno de España, 2024) desde un punto de vista cualitativo. El activo en riesgo es el tanto el código como la documentación. La principal amenaza identificada consiste en sufrir una pérdida de información por un fallo de hardware o de software. El valor del activo es muy alto y la degradación que sufriría sería media o alta según se produjera una pérdida parcial o total de información, como consecuencia el impacto sería alto. Sin embargo, la probabilidad de que se produzca la amenaza es baja por lo que se estima un riesgo medio. Tomando como medida de salvaguarda la realización de copias de seguridad en la nube de forma periódica se consigue obtener un riesgo residual muy bajo respecto de esta amenaza (Rubio, 2021).

Para la gestión del proyecto se realizará lo planificado. Además, se conservarán los registros obtenidos de otras actividades para formar un archivo histórico que sirva de fuente de conocimiento para futuros proyectos.

4.3. Grupos de Actividades Pre-Desarrollo

Estas servirán como base para llevar a cabo pruebas de concepto o explorar alternativas relativas a los requisitos de sistema antes de empezar con el desarrollo del software.

4.3.1. Actividades de Exploración de Conceptos

Primero se han identificado a los stakeholders, que son las personas que tienen algún interés en el sistema. Es fundamental entender las metas y necesidades que tiene cada uno. Se han encontrado los siguientes stakeholders:

Stakeholders directos:

- Gerente del cliente
- Director de producción del cliente
- Usuario que pertenece al cliente
- Usuario anónimo online
- Gerente de la empresa contratista
- Ingeniero de software del contratista
- Científico de datos del contratista

Stakeholders indirectos:

- Entidades públicas regulatorias y fiscales.

Dadas las limitaciones existentes, los roles de la empresa contratista se fusionan en el autor de este trabajo.

Se ha mantenido una reunión con el gerente y director de producción del cliente para anotar las ideas o necesidades principales en las que deberá profundizar en actividades posteriores.

Se han formulado enfoques alternativos para abordar el problema: desarrollar una aplicación de escritorio o una aplicación web, desarrollar un modelo que procese los datos en local o de forma distribuida, contratar el desarrollo a una empresa externa o realizar con recursos propios. Se han realizado estudios de viabilidad sobre cada enfoque.

Finalmente, basándose en lo anterior se ha llegado a la propuesta de creación de la aplicación web, con procesado distribuido y desarrollado con recursos propios.

4.3.2. Actividades de Asignación del Sistema

Con las actividades anteriores se pueden obtener los requisitos del sistema con los que se podrá diseñar la arquitectura. Este grupo de actividades definirán las entradas, los procesos y las salidas, junto con el hardware, el software y los interfaces. En la figura 13 se muestra la arquitectura básica del sistema:

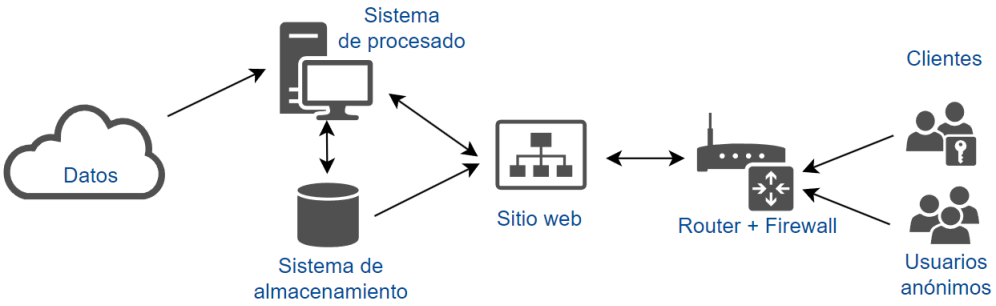


Figura 13: Arquitectura básica del sistema (elaboración propia)

Para llevar a cabo el desarrollo y despliegue se necesitarían los requisitos mínimos mostrados en la tabla 9:

	Desarrollo	Despliegue
Hardware	PC (CPU 24 núcleos, 64 GB RAM, SSD 1 TB)	12 contenedores en la nube (8 vCPUS, 8GB RAM, 500GB)
	Router WiFi (100 Mbps con conectividad estándar 802.11b/g/n)	
Software	Licencia de Windows 11 Home	Distribución Linux Ubuntu 22.04
	Resto de software gratuito: Docker, Apache Hadoop, Apache Spark, Python...	
Rendimiento	2 accesos simultáneos a la web	50 accesos simultáneos a la web
	Ancho banda: 1 Gbits	Ancho banda: 15 Gbits
	Latencia: 100 microsegundos	Latencia: 10 microsegundos
Seguridad	Nivel mínimo acceso sección clientes	Nivel máximo acceso al sistema y a la sección clientes
Legales	-	Gestión adecuada para cumplir el reglamento de Protección de Datos

Tabla 9: Requistos mínimos para el desarrollo y el despliegue (elaboración propia)

4.4. Grupos de Actividades de Desarrollo

4.4.1 Actividades de Requisitos

Estas actividades son vitales para el buen fin del proyecto. Se busca la obtención y especificación de los requisitos funcionales y no funcionales del sistema. Para ello se ha realizado una serie de sesiones de trabajo con el cliente con el objetivo de identificar dichos requisitos, refinarlos, categorizarlos y especificarlos. En la tabla 10 se refleja una lista de Actor-Objetivo, producto obtenido tras de estas reuniones.

ACTOR	OBJETIVO
Usuario anónimo online	Consultar información pública
Usuario anónimo online	Consultar precio servicios
Usuario que pertenece al cliente	Acceder a la zona privada
Usuario que pertenece al cliente	Recibir soporte técnico
Usuario que pertenece al cliente	Recibir alertas
Director de producción del cliente	Acceder a la zona privada
Gerente del contratista	Acceder a la zona privada
Gerente del contratista	Consulta estado de los clientes del servicio
Ingeniero de software del contratista	Acceder a la zona privada
Ingeniero de software del contratista	Gestionar cuentas de usuarios
Ingeniero de software del contratista	Gestionar incidencias del servicio
Científico de datos del contratista	Acceder a la zona privada
Científico de datos del contratista	Gestionar los datos del sistema de inferencia
Científico de datos del contratista	Gestionar el motor de inferencia

Tabla 10: Lista de Actor-Objetivo (elaboración propia)

Gracias a las entrevistas realizadas se han educido una serie de requisitos. Se han analizado para comprobar que son correctos, no ambiguos, completos, consistentes, clasificables, verificables y trazables. Como se han detectado inconsistencias entre algunos, ha sido necesario negociarlos con las personas interesadas. Después se han clasificado y codificado. En la tabla 11 se muestra el resultado.

REQUISITOS	OBLIGATORIOS	DESEABLES
FUNCIONALES	FO-01: Realizar predicciones en tiempo real FO-02: Interfaz de usurario en español e inglés FO-03: Sección de acceso público FO-04: Alta fiabilidad del sistema predictor	FD-01: Interfaz de usurario en otros idiomas regionales de España FD-02: API para el acceso al servicio FD-03: Menús contextuales FD-04: interfaz de usuario con temas claro y oscuro
NO FUNCIONALES	NO-01: 24h para su recuperación ante caídas del sistema NO-02: Servicio técnico en horario laboral NO-03: Formación inicial por parte del contratista NO-04: Sistema modular escalable en Docker NO-05: Programarse en Python. NO-06: Contratar servicios en la nube NO-07: Almacenamiento y procesado distribuido NO-08: Seguir la metodología en cascada NO-09: Énfasis en la seguridad desde el inicio NO-10: Bases de datos relacionales basadas en la nube para la etapa de despliegue NO-11: Conocimiento de SQL, Python y librerías Ciencia de Datos	ND-01: Compra de datos privados
EMPRESARIALES	EO-01: Tarifa plana con periodo de prueba EO-02: Plazo máximo de desarrollo 2 meses EO-03: Cumplir la ley protección de datos EO-04: Presupuesto inicial de 6.120 euros EO-05: Equipo mínimo de una persona involucrada directamente en el proyecto EO-06: Dar formación sobre BigData e Inteligencia artificial	

Tabla 11: Clasificación de los requisitos educidos (elaboración propia)

Para seguir profundizando se han elaborado los casos de uso a partir de la identificación de los actores del sistema y sus objetivos. En la tabla 12 se muestran los casos de uso definidos:

ID	OBJETIVO	ACTOR	DEFINICIÓN
CU01	Consultar información pública	Usuario anónimo online	Navegar por la página de incio del portal web.
CU02	Consultar precio servicios	Usuario anónimo online	Consultar el precio de los servicios que ofrece la empresa.
CU03	Acceder a la zona privada cliente	Usuario que pertenece al cliente	Acceder a la zona privada con el usuario y contraseñas fijadas
CU04	Consultar información exclusiva	Usuario que pertenece al cliente	Consultar la información exclusiva de clientes
CU05	Recibir soporte técnico	Usuario que pertenece al cliente	Recibir soporte técnico ante posibles incidencias
CU06	Gestionar cuentas de usuarios	Ingeniero de software del contratista	Añadir, eliminar, modificar cuentas de usuarios y los permisos que tienen en el sistema.
CU07	Gestionar incidencias del servicio	Ingeniero de software del contratista	Gestionar incidencias del servicio
CU08	Gestionar el sistema inteligente	Científico de datos del contratista	Gestionar el modelo predictor, mejorando y reentrenado el mismo.

Tabla 12: Listado de casos de uso (elaboración propia)

A continuación, se presenta el Diagrama de los Casos de Uso, que sirve de apoyo a la descripción textual de los casos de uso:

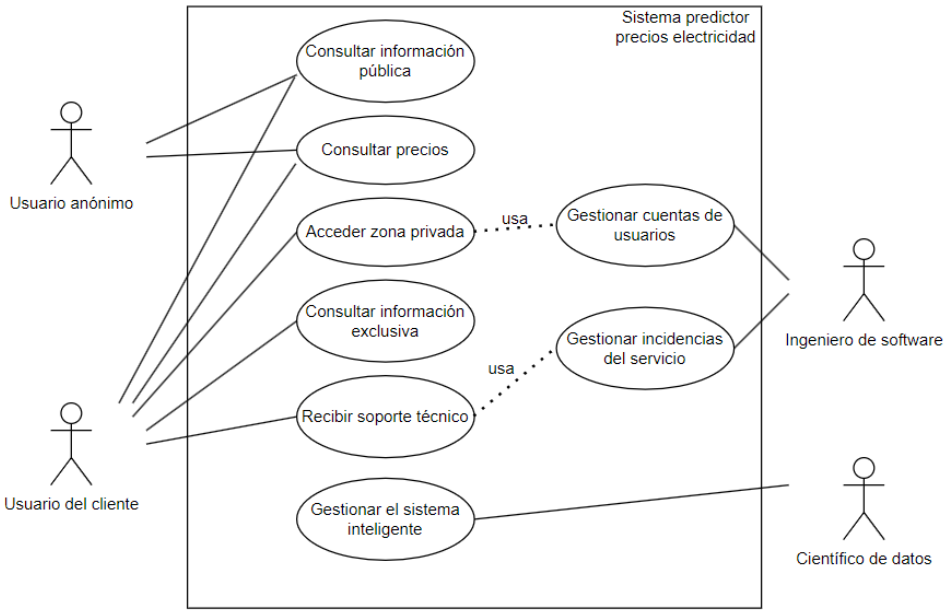


Figura 14: Diagrama de los Casos de Uso (elaboración propia)

Cada caso de uso se describe de forma textual de una forma estructurada usando una plantilla, de este modo se facilita su análisis y comprensión. En el Anexo A1 están detallados todos los casos de uso.

Adicionalmente, se han elaborado los Diagramas de Actividad y de Canal de los casos de uso. En la figura 15 se muestra un ejemplo.

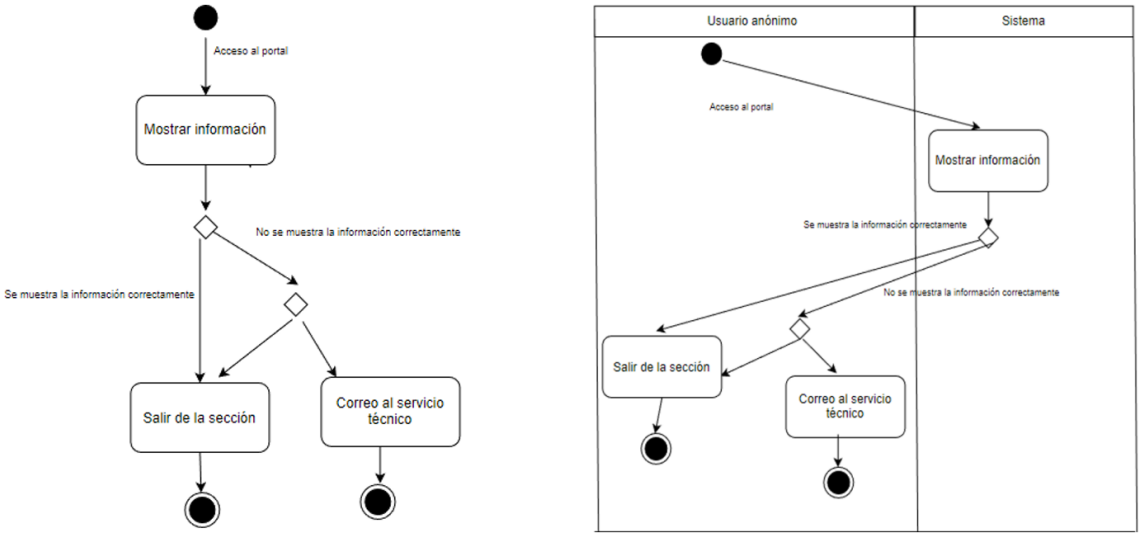


Figura 15: Diagramas de Actividad y de Canal del caso de uso CU01 (elaboración propia)

Con la información recopilada se ha elaborado el Documento de Especificación de Requisitos (ERS), que se adjunta en el Anexo A2. Este documento permite verificar la validez, consistencia, completitud, realismo y verificabilidad de los requisitos. Tras su elaboración, el cliente realiza su revisión (verifica los requisitos reflejados en el mismo). Este documento tendrá un carácter precontractual.

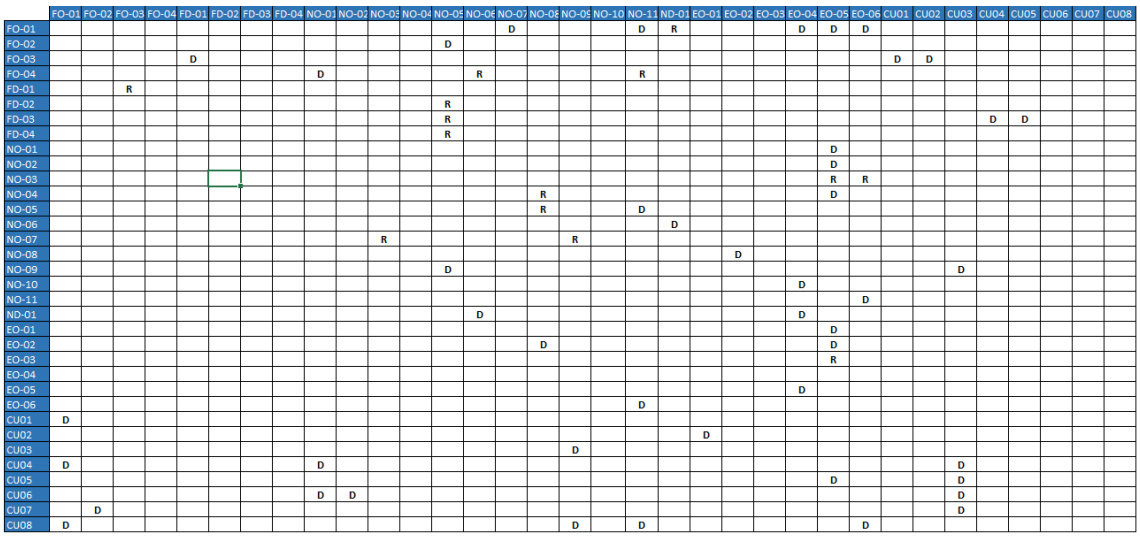
El documento tiene los siguientes apartados:

1- Introducción: establece el propósito del documento, el ámbito del sistema, definiciones, acrónimos y abreviaturas, las referencias y una visión general del documento.
2- Descripción general: describe el contexto, facilitando la posterior comprensión de los requisitos del sistema. Contiene la perspectiva y las funciones del producto, características de los usuarios, restricciones, dependencias y requisitos futuros.
3- Requisitos Específicos: se relacionan los requisitos con el detalle suficiente que permitirá al ingeniero de software y al científico de datos diseñar el sistema de forma adecuada. Contiene la especificación de las interfaces externas, las

funciones, los requisitos de rendimiento, las restricciones de diseño, los atributos del sistema y otros requisitos.

4- Apéndice: se ofrece un análisis de costes del proyecto.

Por último, se ha elaborado la matriz de trazabilidad, expuesta en la figura 16, que muestra las dependencias entre requerimientos. Una “D” en una celda muestra que el elemento de la fila depende del elemento de la columna. Una “R” muestra que existe alguna otra relación débil entre el elemento de la fila y el de la columna. Esta matriz permitirá identificar las fuentes y relaciones entre los requisitos ayudando a establecer prioridades y realizar seguimiento de los mismos.



	FO-01	FO-02	FO-03	FO-04	FD-01	FD-02	FD-03	FD-04	NO-01	NO-02	NO-03	NO-04	NO-05	NO-06	NO-07	NO-08	NO-09	NO-10	NO-11	ND-01	EO-01	EO-02	EO-03	EO-04	EO-05	EO-06	CU01	CU02	CU03	CU04	CU05	CU06	CU07	CU08
FO-01															D				D	R				D	D	D									
FO-02													D																					
FO-03					D																						D	D						
FO-04									D					R					R																
FD-01			R																															
FD-02													R																						
FD-03													R																		D	D			
FD-04													R																						
NO-01																									D									
NO-02																									D									
NO-03																									R	R								
NO-04																R									D									
NO-05																R			D															
NO-06																				D														
NO-07											R					R																			
NO-08																						D												
NO-09													D																D					
NO-10																								D										
NO-11																										D								
ND-01														D										D										
EO-01																									D									
EO-02																D									D									
EO-03																									R									
EO-04																																		
EO-05																								D										
EO-06																			D															
CU01	D																																	
CU02																					D													
CU03																													D					
CU04	D								D																				D					
CU05																									D				D					
CU06									D	D																			D					
CU07		D																											D					
CU08	D																		D							D								

Figura 16: Matriz de trazabilidad entre requerimientos (elaboración propia)

4.4.2 Actividades de Diseño

Primero, se ha realizado el diseño de alto nivel (HLDA - High Level Design Architecture) tomando como base la especificación de los requisitos software y la arquitectura del sistema realizada en el epígrafe 4.3.2 (Actividades de Asignación del Sistema). El sistema seguirá una estructura en tubería que permitirá la paralelización de las tareas y la gestión individual de cada módulo. En la figura 15 se muestra el diseño.

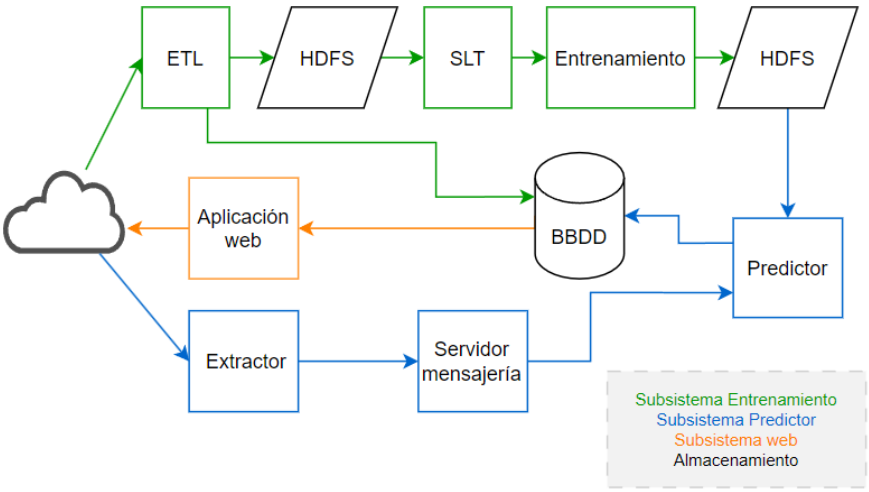


Figura 15: Diseño arquitectónico del sistema (elaboración propia)

El prototipo estará compuesto por cuatro subsistemas, que a su vez contendrán módulos independientes:

- Subsistema de entrenamiento:
    - Módulo ETL (extracción, transformación y carga): encargado de la extracción de los datos de ficheros descargados de Internet. Realizará su transformación escogiendo los atributos que se necesitan, agregando los registros para pasar de datos horarios a datos diarios. Finalmente, los cargará en los dos sistemas de almacenamiento.
    - Módulo SLT (selección, limpieza y transformación): encargado de la selección de los datos escogiendo el plazo temporal deseado. Limpiará los registros con datos faltantes o nulos. Los transformará añadiendo nuevos atributos partiendo de los existentes, como el atributo mes o día de la semana. Estos datos nutrirán el modelo de aprendizaje automático.
    - Módulo de entrenamiento: generará el modelo de aprendizaje y lo evaluará.
- Subsistema de Predictor: se ha optado por seguir el patrón de diseño “Observer” para una comunicación entre módulos asíncrona en lugar de utilizar una API.
    - Extractores: actuarán como publicadores, recopilarán automáticamente los datos de internet y los enviarán al servidor de mensajería.

- Servidor de mensajería: gestiona la recepción y envío de mensajes entre los publicadores y suscriptores.
- Predictor: actúa como suscriptor y utiliza el modelo de aprendizaje para realizar las predicciones.

- Subsistema web:
  - Aplicación web: muestra la información almacenada en la base de datos.
- Subsistemas de almacenamiento:
  - Base de datos: recopila la información a mostrar en la aplicación web.
  - Sistema de archivos HDFS: para el almacenamiento distribuido de los datos usados en el subsistema de entrenamiento.

Seguidamente, se ha realizado un diseño detallado (LLD - "Low Level Design"). Los módulos se implementarán en contenedores Docker y serán integrados en una red virtual como se muestra en la figura 16.

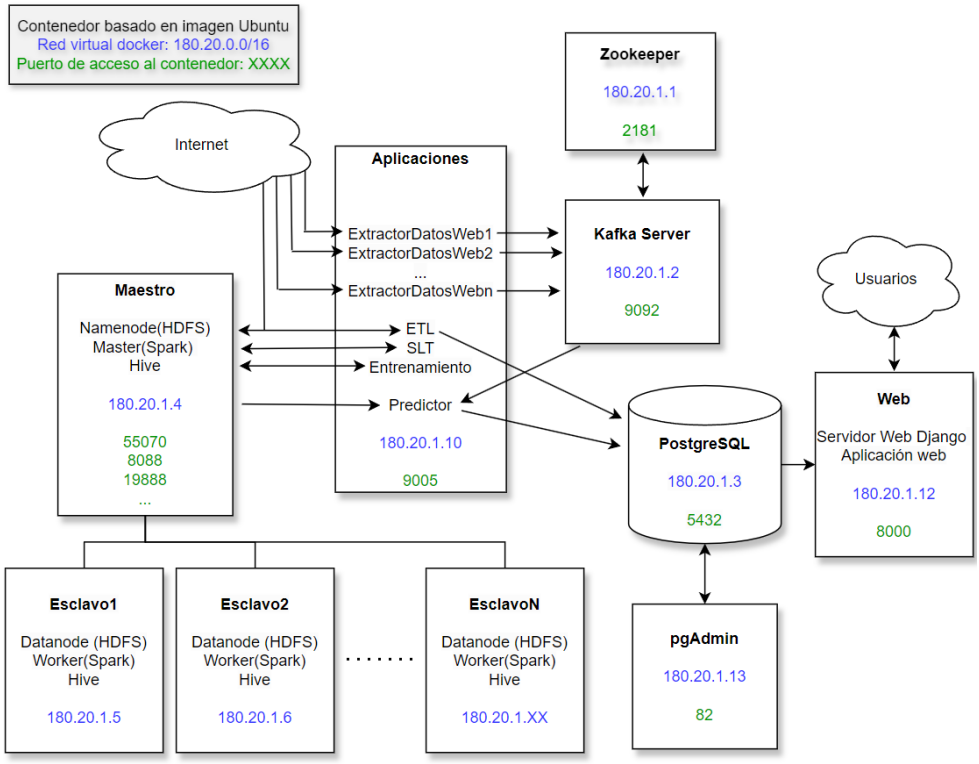


Figura 16: Diseño detallado de la red de contenedores Docker (elaboración propia)

El sistema integrará dos **clusters**. Uno de almacenamiento Hadoop con un nodo maestro y varios esclavos. Otro de procesamiento Spark con un nodo gestor y varios workers. Un contenedor tendrá el maestro de Hadoop y el gestor de Spark. Y habrá varios contenedores que tendrán cada uno un esclavo de Hadoop y un worker de Spark. De esta forma se aprovechará la capacidad de almacenamiento y proceso de cada contenedor.

En otro contendor se ejecutarán todas las aplicaciones relacionadas con la minería de datos. El **sistema de mensajería** se encontrará en un contenedor para el servidor Kafka y en otro para Zookeeper, orquestador de servidores Kafka. El sistema de almacenamiento en base de datos estará compuesto por un contenedor para la base de datos PostgreSQL y otro para el administrador web de la misma (pgAdmin).

Por último, un contenedor contendrá la **aplicación web** y servidor para ejecutarla basados en Django. La aplicación web sigue el patrón MVT o Model-View-Template (modelo-vista-plantilla): el modelo hace referencia a todo lo relacionado con el procesado de la información, la plantilla a cómo se mostrará la información y la vista es enlace entre los anteriores, decide qué plantilla mostrará cada información.

En el Diagrama de Flujo de Datos (DFD) de las figuras 17 y 18 se representa el **flujo de la información** y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida (Ortigosa y Vázquez, 2009).

Figura 17: Diagrama de Flujo de Datos: Nivel de contexto (elaboración propia)

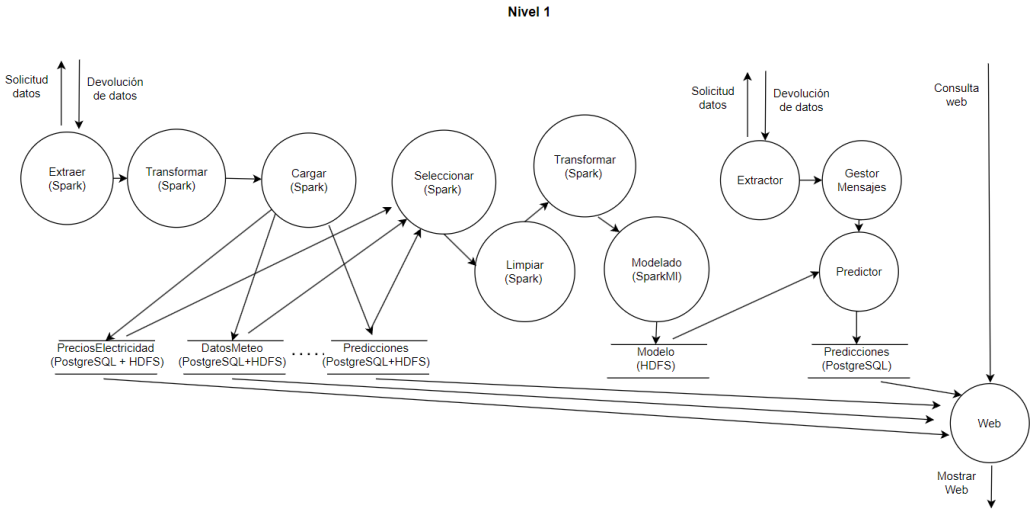


Figura 18: Diagrama de Flujo de Datos: Nivel 1 (elaboración propia)

Se ha diseñado la **base de datos** separando la parte relativa a la aplicación web, como la gestión de los usuarios y sus permisos, y la parte relacionada con el aprendizaje automático. En la figura 19 se muestra el Diagrama de Entidad Relación:

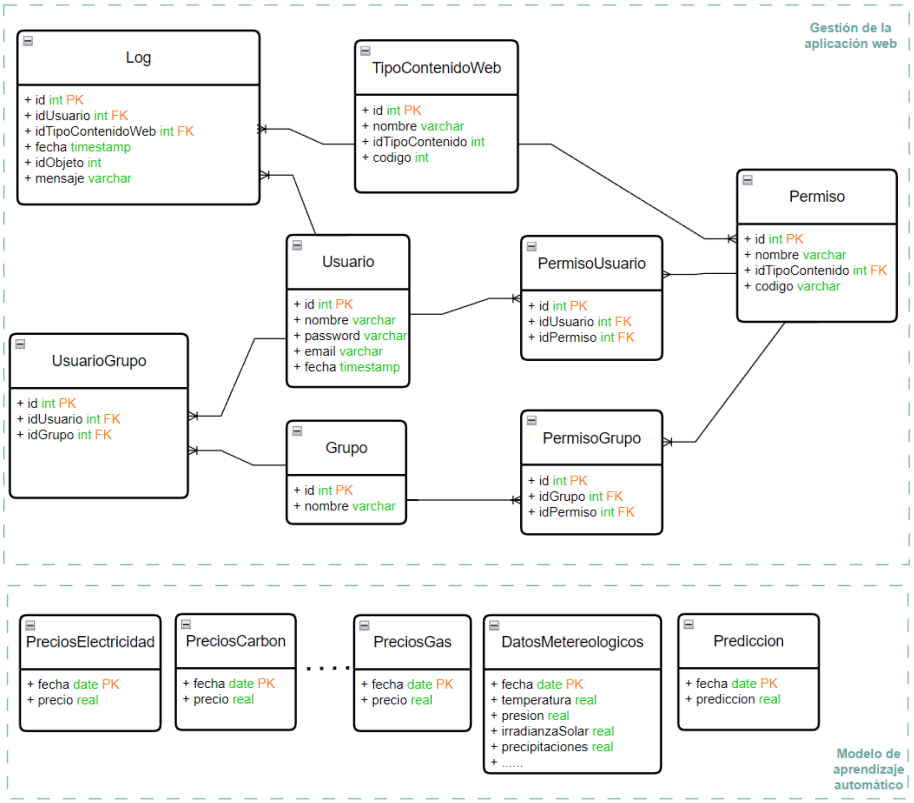


Figura 19: Diagrama Entidad Relación de la base de datos (elaboración propia)

En cuanto al diseño del interfaz gráfico de usuario se han seguido las heurísticas de Nielsen para mejorar la usabilidad (Ferré, 2015). Pueden destacarse los siguientes aspectos:

• Control del usuario y libertad de acción.

• Consistencia y estándares.

• Favorecer reconocimiento sobre recuerdo.

• Visibilidad del estado del sistema.

• Conexión entre el sistema y el mundo real.

• Diseño estético y minimalista.

En la figura 20 se muestra el mapa de navegación de la aplicación web, elaborado para detectar errores de diseño y entender la organización general de la interfaz de usuario.

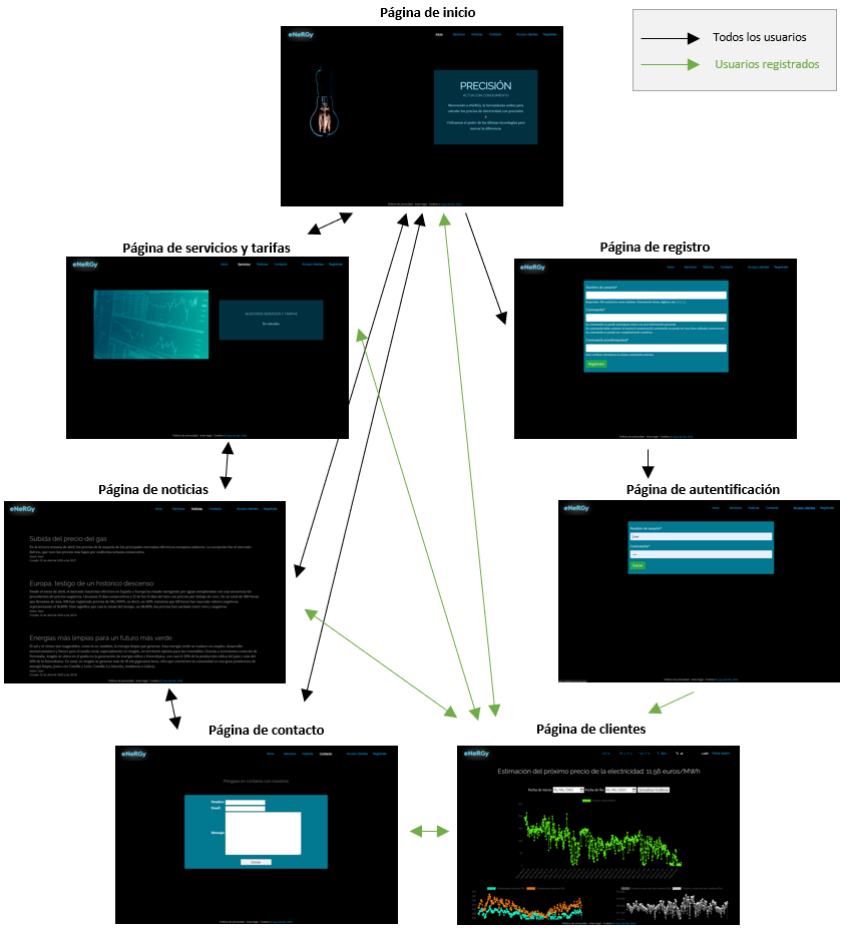


Figura 20: Mapa de navegación (elaboración propia)

4.4.3 Actividades de Implementación

Primero se ha construido la estructura que soporta todo el sistema, sobre la que correrán las aplicaciones. Es decir, se han levantado los distintos contenedores de Docker. Para ello se ha utilizado Docker Compose que permite la automatización de la construcción de redes de contenedores. Se ha creado un archivo YAML de configuración donde se detallan los diferentes servicios que se quieren incluir. En la figura 21 se muestra un fragmento del contenido donde especifica el contenedor web incluyendo la carpeta donde se encuentra el archivo Dockerfile, su nombre, el puerto de acceso, la carpeta que comparte con el sistema anfitrión y la configuración de la red.

```
167   web:
168     build: ./web
169     container_name: web
170     ports:
171       - "8000:8000"
172     depends_on:
173       - postgresql
174     networks:
175       network:
176         ipv4_address: 180.20.1.12
177     stdin_open: true
178     tty: true
179     volumes:
180       - ./web/proyectoweb:/proyectoweb
181
182 networks:
183   network:
184     driver: bridge
185     ipam:
186       config:
187         - subnet: 180.20.0.0/16
```

Figura 21: Fragmento del fichero “docker-compose.yml” (elaboración propia)

Se han creado diferentes carpetas con sus respectivos ficheros Dockerfile. En estos ficheros se detallan las instrucciones para crear cada contenedor: imagen base del contenedor (se ha utilizado imágenes de Ubuntu y Postgres), archivos a copiar dentro, librerías a instalar y configuración de las variables de entorno según cada caso. A modo de ejemplo, la figura 22 expone el contenido del archivo Dockerfile del servidor Kafka.

```
# Utilizar Ubuntu como base
FROM ubuntu:latest

# estbalecer a root como usuario
USER root

# abrir puerto
EXPOSE 9092

# Instalar dependencias
RUN apt-get update
RUN apt install -y wget openjdk-8-jdk

# coger recursos
RUN wget https://downloads.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz
RUN tar -zxvf kafka_2.13-3.7.0.tgz
RUN rm kafka_2.13-3.7.0.tgz
RUN mv /kafka_2.13-3.7.0 /kafka

RUN mkdir /kafka-logs
ADD server.properties /kafka/config/

#  Ejecutar servidor kafka
CMD /kafka/bin/kafka-server-start.sh /kafka/config/server.properties
```

Figura 22: Contenido del archivo Dockerfile del servidor Kafka (elaboración propia)

Para la creación de los nodos de Hadoop y Spark la configuración es más compleja. Se debe crear un usuario que gestionará Hadoop, HDFS y YARN (gestor de recursos de Hadoop). Se crea una clave SSH que se copiará a todos los nodos del *cluster* para la comunicación segura entre ellos. Tras copiar las librerías se establecen las variables de entorno de cada uno. Se copian los archivos con la configuración especifica de este prototipo. Se instala Delta Lake. Se abren los puertos necesarios y finalmente se ejecuta el script de inicio en modo seguro con ssh.

Los archivos de configuración son:

- Ficheros xml ("core-site.xml", "hdfs-site.xml", "mapred-site.xml" y "yarn-site.xml") que especifican el factor de replicación, la dirección de red del maestro y la disposición máxima de memoria, entre otros aspectos. En la figura 23 se muestra un fragmento de la configuración de HDFS.

```
<property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>maestro:50090</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
<property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
```

Figura 23: Fragmento del archivo "hdfs-site.xml"

- Ficheros “masters” y “workers” donde se especifica las direcciones de red de cada uno.

En un terminal desde el director raíz se ejecuta la instrucción “docker compose up”. Ésta crea todas las imágenes de los contenedores y los arranca. En este prototipo los *clusters* de Hadoop y Spark tendrán un nodo maestro y cuatro nodos esclavos. El terminal muestra, entre otros, los mensajes del proceso y la aplicación de Docker informa sobre el estado del sistema como se muestra en la figura 24.

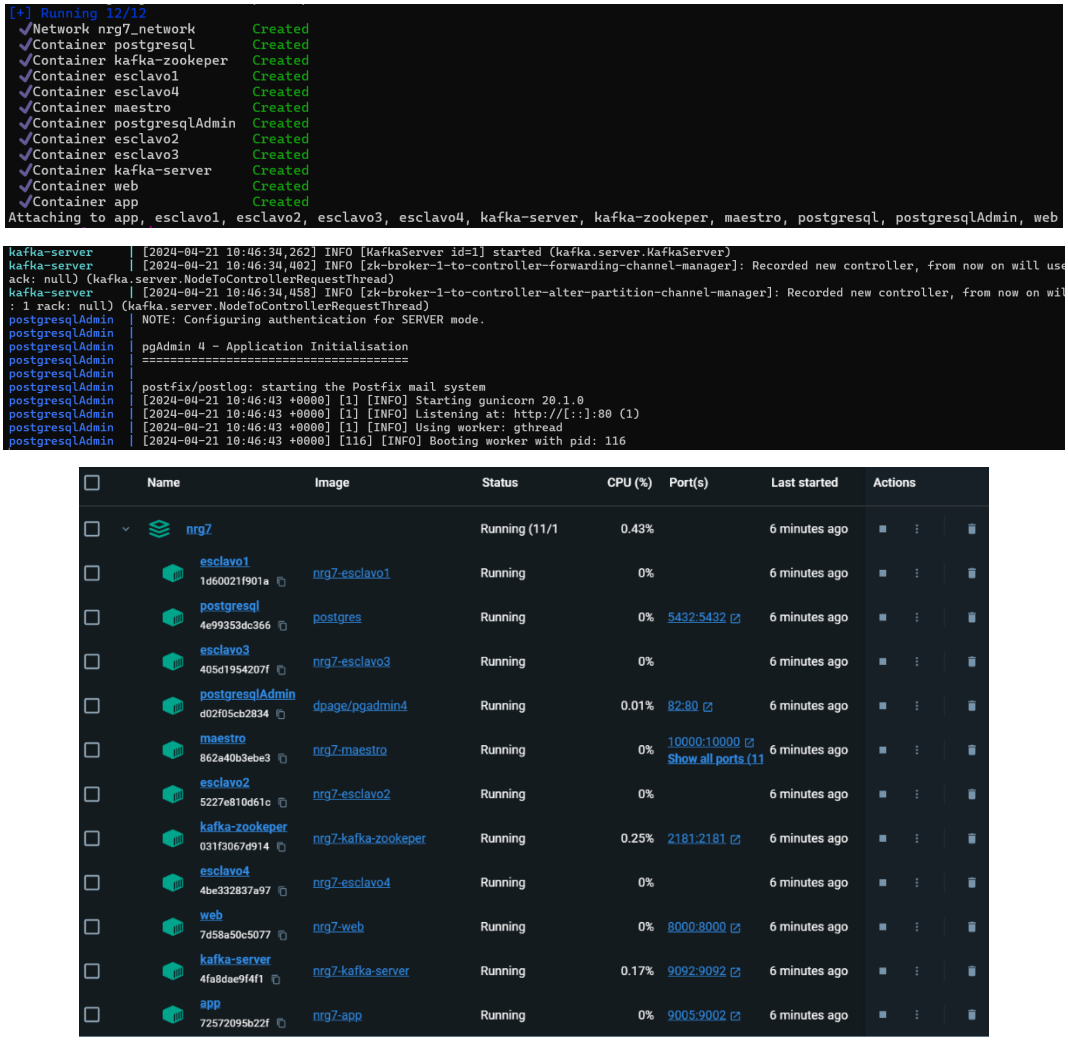


Figura 24: Creación y estado de los contenedores Docker (elaboración propia)

Seguidamente, se ha creado un script (“start.sh”) para iniciar los servicios de Hadoop, Spark, Hive y Delta. El contenido se indica en la figura 25.

```
#!/bin/bash
  echo "----- Formateando hdfs -------"
  docker exec -u hduser -it maestro ./home/hduser/hadoop/bin/hdfs namenode -format
  sleep 5
  echo "----- Arrancando hdfs -------"
  docker exec -u hduser -it maestro start-dfs.sh
  sleep 5
  echo "----- Arrancando yarn -------"
  docker exec -u hduser -d maestro start-yarn.sh
  sleep 5
  echo "----- Arrancando MapReduce-JobHistory Server-------"
  docker exec -u hduser -d maestro mr-jobhistory-daemon.sh start historyserver
  echo "----- Arrancando Spark -------"
  docker exec -u hduser -d maestro start-master.sh
  docker exec -u hduser -d esclavo1 start-slave.sh maestro:7077
  docker exec -u hduser -d esclavo2 start-slave.sh maestro:7077
  docker exec -u hduser -d esclavo3 start-slave.sh maestro:7077
  docker exec -u hduser -d esclavo4 start-slave.sh maestro:7077
  sleep 5
  echo "----- Arrancando  History Server -------"
  docker exec -u hduser maestro start-history-server.sh
  echo "----- Preparando hdfs para hive -----"
  docker exec -u hduser -it maestro hdfs dfs -mkdir -p /tmp
  docker exec -u hduser -it maestro hdfs dfs -mkdir -p /user/hive/warehouse
  docker exec -u hduser -it maestro hdfs dfs -chmod g+w /tmp
  docker exec -u hduser -it maestro hdfs dfs -chmod g+w /user/hive/warehouse
  sleep 5
  echo "----- Arrancando Hive Metastore ----- "
  docker exec -u hduser -d maestro hive --service metastore
  docker exec -u hduser -d maestro hive --service hiveserver2
  echo "Hadoop info          : http://localhost:8088/cluster"
  echo "HDFS                 : http://localhost:55070/dfshealth.html"
  echo "JobHistory Server    : http://localhost:19888"
  echo "Spark maestro        : http://localhost:8080"
  echo "Spark History Server : http://localhost:18080"
```

Figura 25: Archivo de inicio servicios “start.sh” (elaboración propia)

Tras su ejecución en otro terminal con la instrucción “bash ./start.sh” se obtiene confirmación de que el proceso ha terminado correctamente, como se muestra en la figura 26.

```
----- Arrancando hdfs -------
Starting namenodes on [maestro]
maestro: Warning: Permanently added 'maestro' (ED25519) to the list of known hosts.
Starting datanodes
180.20.1.6: Warning: Permanently added '180.20.1.6' (ED25519) to the list of known hosts.
180.20.1.7: Warning: Permanently added '180.20.1.7' (ED25519) to the list of known hosts.
180.20.1.8: Warning: Permanently added '180.20.1.8' (ED25519) to the list of known hosts.
180.20.1.5: Warning: Permanently added '180.20.1.5' (ED25519) to the list of known hosts.
Starting secondary namenodes [maestro]
2024-04-21 10:59:24,815 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
----- Arrancando yarn -------
----- Arrancando MapReduce-JobHistory Server-------
----- Arrancando Spark -------
----- Arrancando  History Server -------
starting org.apache.spark.deploy.history.HistoryServer, logging to /home/hduser/spark/logs/spark--org.apache.spark.deplo
y.history.HistoryServer-1-862a40b3ebe3.out
```

Figura 26: Mensajes mostrados por consola al iniciar los servicios (elaboración propia)

A través del puerto 8088 se accede a la aplicación web de Hadoop que muestra el estado de los nodos del cluster, lo que viene reflejado en la figura 27.

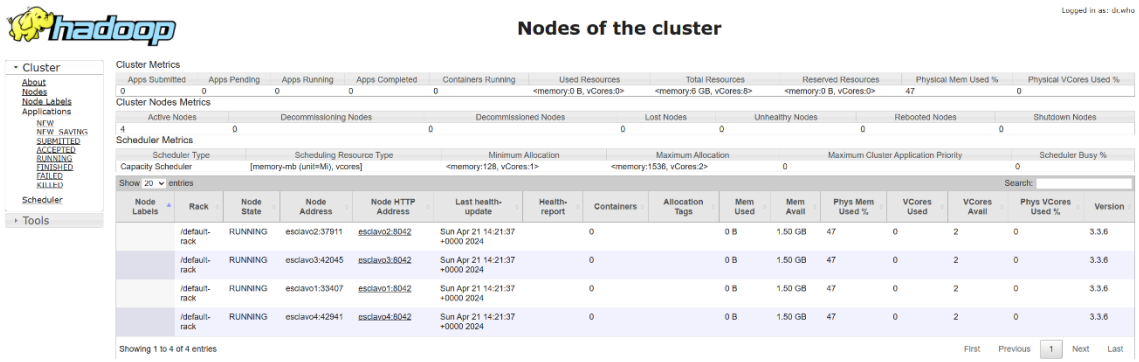


Figura 27: Información del cluster de Hadoop (elaboración propia)

En la figura 28 se muestra la aplicación web de Spark con la información relativa a sus nodos del cluster. Se accede a través del puerto 8080.

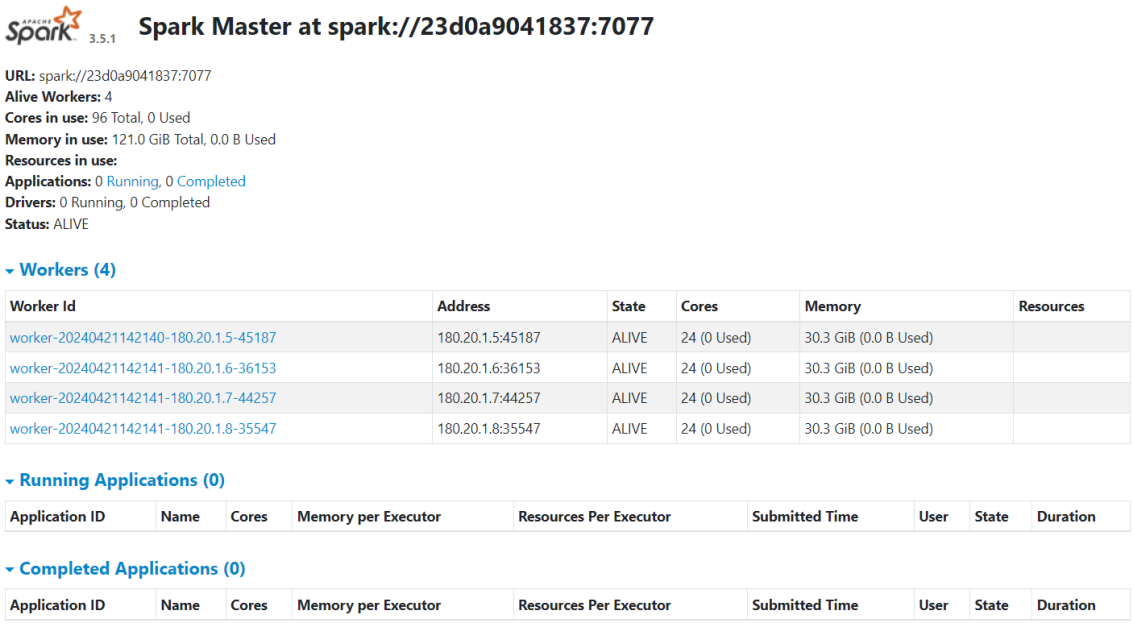


Figura 28: Información del cluster de Spark (elaboración propia)

La figura 29 muestra el gestor de la base de datos PostgreSQL, pgAdmin. Se accede a través del puerto 82.

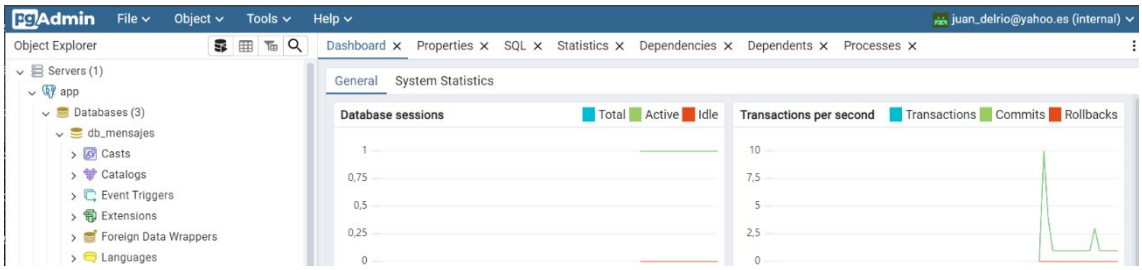


Figura 29: Gestor de PostgreSQL (elaboración propia)

Una vez que la estructura se ha levantado satisfactoriamente, se inicia la fase de codificación de los distintos módulos. Para el desarrollo se han utilizado los editores Visual Studio Code y Notepad++. Se ha codificado en el lenguaje Python tanto los módulos de minería de datos como la aplicación web. Además, se ha utilizado el lenguaje SQL para la gestión de los datos.

En el módulo ETL se inicia una sesión de Spark especificando el gestor de recursos y la configuración de Delta Lake con las siguientes líneas:

```
builder = SparkSession.builder.master("yarn").appName("app")
.config("spark.sql.extensions",
"io.delta.sql.DeltaSparkSessionExtension").config("spark.sql.catalog.s
park_catalog","org.apache.spark.sql.delta.catalog.DeltaCatalog")

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

Se cargan los datos de diferentes fuentes en Dataframes de Spark, dejando a Spark que identifique el esquema de los mismos y recupere los encabezados:

```
df_datos =  spark.read.option("inferSchema",true") .option("header",
"true").csv("file:///datos.csv")
```

Se transforman los datos usando las funciones SQL de Spark (select, filter, join, groupBy, …) y se van almacenando en distintos Dataframes. Se almacenan los datos en tablas de la base de datos creando una conexión y ejecutando instrucciones SQL.

Finalmente, se almacena con formato Delta Lake en el sistema de archivos HDFS con esta instrucción:

```
df.write.format('delta').mode("overwrite").save("hdfs:///datos" )
```

Los módulos SLT y Entrenamiento se han codificado en un solo fichero. Tras crear sesión de Spark y recuperar los datos almacenados en el módulo anterior se realizan las tareas de selección, limpieza y transformación con las funciones SQL. Después se crea un modelo de Regresión Lineal. Se establecen las columnas de atributos y etiquetas y los datos de entrenamiento y evaluación. Se entrena el modelo, se guarda para su posterior uso y se evalúa a través de la Raíz del Error Cuadrático Medio (RMSE) con el siguiente código:

```
lr_model = lr.fit(train_data)
lr_model.save("hdfs:///linear_regression_model")
predictions = lr_model.transform(test_data)
# Mostrar predicciones
predictions.select(predictions['*']).show()
# Evaluar modelo
evaluator = RegressionEvaluator(labelCol="precio",
 predictionCol="prediction", metricName="rmse")
```

En el módulo Extractor se crea un objeto productor de Kafka:

```
kafka_producer_obj = KafkaProducer(bootstrap_servers='kafka-
 server:9092',value_serializer=lambda x: dumps(x).encode('utf-8'))
```

Se extraen los datos de Internet a través de las APIs de cada página. Se establece el tema (topic, en inglés) y se envían los datos al servidor Kafka:

```
topic = "Topic_Variables"
envio = kafka_producer_obj.send(topic, mensaje)
```

En el módulo Predictor se crea un objeto consumidor de Kafka:

```
bootstrap_servers = ['kafka-server:9092']
    consumer = KafkaConsumer(
        topic,
        bootstrap_servers=bootstrap_servers,
        group_id=group_id,
        auto_offset_reset='latest',
        enable_auto_commit=True,
        max_poll_records = 1,
        value_deserializer=lambda x: x.decode('utf-8')
    )
```

Se extraen los mensajes, se procesan y se cierra el consumidor. Se carga el modelo de regresión lineal, se realiza la predicción y se almacena en base de datos el resultado.

Para crear la aplicación web se entra en el contenedor "web" con la instrucción "docker exec -it web bash". Se ejecutan las instrucciones:

```
django-admin startproject proyecto_web
python manage.py startapp web
python manage.py makemigrations
python manage.py migrate
```

Se inicia el servidor web con "python3 manage.py runserver 0.0.0.0:8000", de esta forma será accesible en el puerto 8000.

Dentro del proyecto se ha creado una aplicación por cada página web (inicio, contacto, servicios, autenticación, panel y blog). En el archivo "settings.py" se han establecido las aplicaciones incluidas, se ha configurado la conexión a la base de datos y las etiquetas de los mensajes, entre otros.

En el archivo "urls.py" se definen los accesos a cada página con esta lista:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('web.urls')),
    path('servicios/', include('servicios.urls')),
    path('blog/', include('blog.urls')),
```

```
    path('contacto/', include('contacto.urls')),
    path('panel/', include('panel.urls')),
    path('registro/', include('autenticacion.urls')),
]
```

Se crea la plantilla base que será usada en todas las páginas y el resto de plantillas con código Html. Se crean los archivos de estilos css y se añaden las imágenes.

Para la página de contacto se ha creado un formulario que se lanza con una petición http. Para la página Panel, que muestra las predicciones y los gráficos, se han definido los modelos que almacenarán la información de la base de datos. Como ejemplo se muestra la clase para la tabla de precios:

```
class tb_precios(models.Model):
    fecha = models.DateField(primary_key=True)
    precio = models.FloatField()
```

En el archivo "views.py" se implementa la extracción de la información de la base de datos. Por ejemplo:

```
datos_precios = tb_precios.objects.order_by('fecha')
fechas_precios = [dato.fecha.strftime('%Y-%m-%d') for dato in
datos_precios]
valores_precios = [dato.precio for dato in datos_precios]
```

Por último, en la plantilla de Panel se muestra la estimación realizada y los gráficos con la evolución temporal de algunas variables. Se ha usado la librería de Javascript chart.js para mostrar los gráficos, dibujándolos dentro de elementos canvas. Se ha añadido un formulario para la selección dinámica del rango de fechas a mostrar implementado en lenguaje Javascript. Para la maquetación de las páginas se ha usado la librería BootStrap de Javascript.

CAPÍTULO 5: EVALUACIÓN

5.1. Pruebas

5.1.1. Pruebas unitarias

5.1.1.1. Pruebas unitarias de caja blanca

Para la prueba de ruta base (cobertura de caminos) se ha calculado la complejidad ciclomática de McCabe de los distintos módulos. En la siguiente tabla se muestran los resultados:

	Nº nodos condición	Complejidad McGabe
01 ETL	2	3
02 Entrenamiento	1	2
03 Extractor variables	1	2
04 Extractor Precios	1	2
05 Predictor	7	8
Aplicación web	9	10

Tabla 13: Complejidad ciclomática de McCabe (elaboración propia)

Tienen una complejidad muy baja debido a la estructura en tubería de la mayor parte del código. Como consecuencia, la prueba de las estructuras de control ha sido sencilla y se ha garantizado la ejecución de todas las sentencias del programa al menos una vez.

5.1.1.2. Pruebas unitarias de caja negra

Se han realizado pruebas con posibles datos de entrada en los módulos de los subsistemas de entrenamiento y predictor tanto con fuentes correctas como erróneas. Para la aplicación web se han utilizado clases de equivalencia y valores límite para comprobar el funcionamiento ante distintas entradas de los usuarios en los formularios de acceso, registro y contacto.

5.1.2. Pruebas de integración

La estructura del sistema presenta un cierto grado de complejidad por lo que se ha probado con detenimiento el correcto funcionamiento del sistema en conjunto. Por otro lado, las aplicaciones de minería de datos y web presentan únicamente acoplamiento de datos (los módulos sólo comparten datos entre ellos). Por lo tanto, las pruebas de integración de esta parte se han centrado en comprobar la correcta lectura y escritura de datos de cada módulo.

5.1.3. Pruebas de validación

Las pruebas de validación buscan errores a nivel de los requerimientos. Se desea saber si el producto cumple con lo requerido por el usuario. Se realizarán cuando el sistema esté íntegramente desarrollado. En su diseño se han enfocado en las expectativas que tiene el usuario sobre el funcionamiento del sistema, la documentación y otros requisitos no funcionales como la facilidad de uso.

Como el producto en desarrollo es una aplicación web, las pruebas están adaptadas a este tipo de software. Además, para facilitar su compresión y seguimiento se muestran con un formato común y se han codificado.

A continuación, se muestran las distintas pruebas diseñadas:

5.1.3.1. Pruebas de Contenido

Con el objetivo de descubrir errores sintácticos, semánticos y de organización de la información se han diseñado las pruebas reflejadas en las tablas 14,15, 16 y 17.

Identificador del caso de prueba	P001
Caso de uso/requisito probado	FO-02: Interfaz de usurario en español e inglés FD-01: Interfaz de usurario en Catalán, Gallego y Euskera CU01: Consultar información pública CU02: Consultar precio servicios CU04: Consultar información exclusiva
Tipo de prueba	De contenido
Estrategia de prueba	**Análisis sintáctico**
Objetivo de la prueba	Uso de la aplicación satisfactorio recibiendo la información de una forma gramaticalmente correcta.
Descripción de la prueba	Se busca errores tipográficos y gramaticales a través de: - Uso de un corrector ortográfico - Un revisor humano Se realizará para los distintos idiomas que muestra la aplicación
Entrada	Texto completo de cada uno de los apartados que muestra la aplicación.
Resultado esperado	Nº errores encontrados a corregir < 10

Tabla 14: Caso de prueba P001 (elaboración propia)

Identificador del caso de prueba	P002
Caso de uso/requisito probado	FO-02: Interfaz de usurario en español e inglés FO-03: Sección de acceso público FD-01: Interfaz de usurario en Catalán, Gallego y Euskera EO-03: Cumplir la ley protección de datos CU01: Consultar información pública CU02: Consultar precio servicios CU04: Consultar información exclusiva
Tipo de prueba	De contenido
Estrategia de prueba	**Análisis semántico**
Objetivo de la prueba	Ofrecer información adecuada y fiable
Descripción de la prueba	El experto en Energía Eléctrica busca errores en la precisión, completitud y adecuación de la información que proporciona la aplicación. Puntos a revisar: - Precisión de la información en tablas y gráficos. - Grado de detalle adecuado en cada elemento - Lenguaje asequible al usuario no experto - Adecuada aportación de las fuentes - Consistencia de la información en toda la aplicación - Adecuación a las leyes de protección de datos y propiedad intelectual
Entrada	Texto completo, gráficos, tablas e imágenes de cada uno de los apartados que muestra la aplicación.
Resultado esperado	Nº elementos necesitados de corrección < 5

Tabla 15: Caso de prueba P002 (elaboración propia)

Identificador del caso de prueba	P003
Caso de uso/requisito probado	FO-02: Interfaz de usurario en español e inglés FO-03: Sección acceso público FD-01: Interfaz de usurario en Catalán, Gallego y Euskera CU01: Consultar información pública CU02: Consultar precio servicios CU04: Consultar información exclusiva
Tipo de prueba	De contenido
Estrategia de prueba	**Análisis de la organización de la información**
Objetivo de la prueba	Ofrecer información ordenada
Descripción de la prueba	El ingeniero de software revisa la forma en que está estructurada la información a través de un análisis visual.
Entrada	Diferentes páginas que muestra la aplicación.
Resultado esperado	Nº cambios necesitados de realizar < 5

Tabla 16: Caso de prueba P003 (elaboración propia)

Identificador del caso de prueba	P004
Caso de uso/requisito probado	CU01: Consultar información pública CU02: Consultar precio servicios CU04: Consultar información exclusiva NO-06: Contratar servicios en la nube NO-10: Bases de datos relacionales basadas en la nube
Tipo de prueba	De contenido
Estrategia de prueba	**Análisis de la base de datos**
Objetivo de la prueba	Comprobar correcto funcionamiento del proceso de extracción de datos de las bases de datos
Descripción de la prueba	El ingeniero de software revisa el proceso de extracción de información: - Acceso a la base de datos - Consulta de la información requerida - Envío al sistema central - Mostrar adecuadamente la información recibida
Entrada	Datos de acceso al usuario/aplicación, parámetros de la consulta
Resultado esperado	Nº errores en envío + nº errores de acceso < 3

Tabla 17: Caso de prueba P004 (elaboración propia)

5.1.3.2. Pruebas de interfaz de usuario

En las tablas 18 a 23 se detallan las pruebas destinadas a verificar la sintaxis y semántica del interfaz de usuario y, de esta forma, obtener una valoración de su usabilidad.

Identificador del caso de prueba	P005
Caso de uso/requisito probado	FO-03: Sección acceso público FD-03: Menús contextuales FD-04: Interfaz de usuario con temas claro y oscuro CU01: Consultar información pública CU02: Consultar precio servicios CU04: Consultar información exclusiva
Tipo de prueba	De interfaz de usuario
Estrategia de prueba	**Análisis de las características de la interfaz**
Objetivo de la prueba	Comprobar correcto diseño estético
Descripción de la prueba	El ingeniero de software revisa visual de los tipos de fuente, colores y posición de los elementos. Se busca que el diseño de las páginas tenga: - Simplicidad - Proximidad de los elementos comunes - Coherencia en la localización entre cada página - Con mezcla de colores adecuados. - Otros atributos que hagan del interfaz amigable, intuitivo y fácil de recordar
Entrada	Pantalla de cada página
Resultado esperado	Nº Defectos de diseño < 5

Tabla 18: Caso de prueba P005 (elaboración propia)

Identificador del caso de prueba	P006
Caso de uso/requisito probado	CU01 a CU08 FO-03: Sección acceso público FD-03: Menús contextuales
Tipo de prueba	De interfaz de usuario
Estrategia de prueba	**Análisis de los mecanismos de interfaz**
Objetivo de la prueba	Comprobar la correcta interacción entre los componentes de la interfaz y el usuario
Descripción de la prueba	El ingeniero de Software sigue los caminos posibles dentro de la interfaz revisando: - Vínculos: correcto funcionamiento - Formularios: etiquetas correctas, valores por defecto, envío correcto… - Ventanas pop-up: tamaño, posición, diseño… - Cookies: gestión adecuada - Modificación correcta del Panel por el usuario
Entrada	Guion prefijado que cubra los caminos probables del usuario.
Resultado esperado	Nº vínculos incorrectos + nº errores en formularios + nº otros errores < 10

Tabla 19: Caso de prueba P006 (elaboración propia)

Identificador del caso de prueba	P007
Caso de uso/requisito probado	CU01 a CU08 FO-03: Sección acceso público FD-03: Menús contextuales
Tipo de prueba	De interfaz de usuario
Estrategia de prueba	**Análisis semántico de interfaz**
Objetivo de la prueba	Comprobar si se ofrece ayuda, retroalimentación y consistencia en el diseño
Descripción de la prueba	El ingeniero de Software prueba cada elemento de la interfaz: vínculos, botones, formularios. Analizar la información proporcionada al usuario en cada elemento.
Entrada	Guion prefijado que cubra la ejecución de todos los elementos.
Resultado esperado	Nº fallos en notificaciones < 3

Tabla 20: Caso de prueba P007 (elaboración propia)

Identificador del caso de prueba	P008
Caso de uso/requisito probado	CU01 a CU08 FO-03: Sección acceso público FD-03: Menús contextuales
Tipo de prueba	De interfaz de usuario
Estrategia de prueba	**Análisis de la usabilidad**
Objetivo de la prueba	Comprobar la usabilidad de la aplicación
Descripción de la prueba	Varios usuarios prueban: - La interactividad con los elementos - La navegabilidad de la web - Legibilidad de textos y gráficos - Capacidad de personalización - Accesibilidad por personas que tiene discapacidades
Entrada	Guion prefijado que cubra la ejecución de todos los elementos.
Resultado esperado	Nº de comportamientos de la aplicación no deseados < 5

Tabla 21: Caso de prueba P008 (elaboración propia)

Identificador del caso de prueba	P009
Caso de uso/requisito probado	CU01 a CU08 FO-03: Sección acceso público FD-03: Menús contextuales
Tipo de prueba	De interfaz de usuario
Estrategia de prueba	**Análisis de la compatibilidad**
Objetivo de la prueba	Comprobar la compatibilidad de la aplicación en distintos dispositivos y sistemas operativos
Descripción de la prueba	El ingeniero de software prueba del funcionamiento y diseño mostrado de la aplicación en distintos: - Dispositivos: ordenador personal, teléfono móvil, Tablet - Sistemas Operativos: Windows, Linux, Mac-Os - Navegadores: Chrome, Edge, Firefox, etc.
Entrada	Acceso a la aplicación a través de la web.
Resultado esperado	Nº errores en dispositivos + en Sist.Operativos + navegadores < 5

Tabla 22: Caso de prueba P009 (elaboración propia)

Identificador del caso de prueba	P010
Caso de uso/requisito probado	CU01 a CU08 FO-03: Interfaz de usurario en español e inglés FD-01: Interfaz de usurario en Catalán, Gallego y Euskera FD-04: interfaz de usuario con temas claro y oscuro
Tipo de prueba	De interfaz de usuario
Estrategia de prueba	**Análisis de configuración de la interfaz de usuario**
Objetivo de la prueba	Comprobar la correcta formación de las distintas configuraciones de la interfaz que puede hacer el usuario
Descripción de la prueba	El ingeniero de software prueba las distintas configuraciones de los elementos de las páginas web.
Entrada	Elección de cambios en la configuración.
Resultado esperado	Nº resultados no deseados al cambiar la configuración < 3

Tabla 23: Caso de prueba P010 (elaboración propia)

5.1.3.3. Prueba de seguridad

En la tabla 24 se especifican las pruebas destinadas a evaluar las vulnerabilidades del sistema.

Identificador del caso de prueba	P011
Caso de uso/requisito probado	CU03: Acceder a la zona privada cliente CU07: Gestionar cuentas de usuarios NO-07: Almacenamiento y procesado distribuido NO-09: Énfasis en la seguridad desde el inicio con métodos activos y pasivos
Tipo de prueba	De seguridad
Estrategia de prueba	**Análisis de las vulnerabilidades del sistema**
Objetivo de la prueba	Comprobar las vulnerabilidades en el lado del cliente y del servidor para que el usuario esté satisfecho con el uso seguro de la aplicación
Descripción de la prueba	Se contrata Empresa Experta en Seguridad Informática para analizar: - Problemas de seguridad de los navegadores - Gestión cookies - Envío de datos por la red - Ataques de inyección SQL contra la base de datos - Ataques al servidor
Entrada	Sistema completo hardware y software
Resultado esperado	$\sum$ Nº vulnerabilidades descubiertas del tipo i x grado de efecto negativo < 0.10

Tabla 24: Caso de prueba P011 (elaboración propia)

5.1.3.4. Pruebas de rendimiento

Con el objetivo de identificar problemas de rendimiento se han diseñado las pruebas reflejadas en las tablas 25 y 26.

Identificador del caso de prueba	P012
Caso de uso/requisito probado	CU01 a CU08 FO-01: Realizar predicciones en tiempo real. NO-06: Contratar servicios en la nube
Tipo de prueba	De rendimiento
Estrategia de prueba	**Análisis de la latencia en la interacción con el usuario**
Objetivo de la prueba	Comprobar la latencia de la aplicación
Descripción de la prueba	El ingeniero de software realiza pruebas de respuesta del sistema en su interacción con el usuario en diferentes localizaciones y conexiones a Internet
Entrada 1	Envío de datos del usuario y recepción de respuesta del servidor en redes banda ancha
Resultado esperado 1	Media de tiempo de respuesta < 40 milisegundos
Entrada 2	Envío de datos del usuario y recepción de respuesta del servidor en redes de conexión limitada
Resultado esperado 2	Media de tiempo de respuesta < 500 milisegundos

Tabla 25: Caso de prueba P012 (elaboración propia)

Identificador del caso de prueba	P013
Caso de uso/requisito probado	CU01 a CU08 FO-01: Realizar predicciones en tiempo real. NO-04: Sistema modular escalable en Docker NO-06: Contratar servicios en la nube NO-07: Almacenamiento y procesado distribuido NO-10: Bases de datos relacionales basadas en la nube
Tipo de prueba	De rendimiento
Estrategia de prueba	**Análisis de la escalabilidad de la aplicación**
Objetivo de la prueba	Comprobar el rendimiento con un aumento de los usuarios
Descripción de la prueba	El ingeniero de software realiza pruebas de respuesta del sistema ante incrementos de usuarios que acceden a la web.
Entrada 1	Tiempo de latencia medio con 1000 usuarios conectados en banda ancha
Resultado esperado 1	Media de tiempo de respuesta < 100 milisegundos
Entrada 2	Tiempo de latencia medio con 10.000 usuarios conectados en banda ancha
Resultado esperado 2	Media de tiempo de respuesta < 150 milisegundos
Entrada 3	Tiempo de latencia medio con 100.000 usuarios conectados en banda ancha
Resultado esperado 3	Media de tiempo de respuesta < 300 milisegundos

Tabla 26: Caso de prueba P013 (elaboración propia)

5.1.3.5. Pruebas de sistemas inteligentes

En la tabla 27 se indican las pruebas relacionadas con la calidad del modelo predictivo.

Identificador del caso de prueba	P014
Caso de uso/requisito probado	CU08: Gestionar el sistema inteligente FO-04: Alta fiabilidad del sistema predictor NO-11: Conocimiento de SQL, Python y librerías Ciencia de Datos. EO-06: Formación sobre Big data e Inteligencia Artificial.
Tipo de prueba	De sistemas de Inteligencia Artificial
Estrategia de prueba	**Análisis de la capacidad predictiva del sistema inteligente**
Objetivo de la prueba	Comprobar que el sistema predictivo de resultados satisfactorios y útiles para el usuario
Descripción de la prueba	El científico de datos realiza pruebas de la fiabilidad del sistema comprobando periódicamente las mediciones de los errores producidos en las estimaciones en tiempo real.
Entrada	Datos en tiempo real para realizar las predicciones.
Resultado esperado	Error cuadrático medio (MSE) de las estimaciones sobre datos reales < 7%

Tabla 27: Caso de prueba P014 (elaboración propia)

5.1.4. Pruebas de sistema

Una vez realizadas las pruebas de los componentes individuales del sistema, se realizan las pruebas de recuperación, de esfuerzo, de rendimiento y despliegue del sistema completo. Estas pruebas se postponen hasta que el sistema no esté desarrollado en su totalidad.

5.2. Matriz de trazabilidad de los requisitos de la ERS con las pruebas del sistema.

Se ha elaborado la matriz de trazabilidad de los requisitos, mostrada en la tabla 28, para ayudar a verificar que todos los requisitos estén implementados adecuadamente y analizar qué pruebas cubre cada requisito.

PRUEBA \ REQUISITO	FO-01	FO-02	FO-03	FO-04	FD-01	FD-02	FD-03	FD-04	NO-01	NO-02	NO-03	NO-04	NO-05	NO-06	NO-07	NO-08	NO-09	NO-10	NO-11	ND-01	EO-01	EO-02	EO-03	EO-04	EO-05	EO-06
Unitaria caja blanca																										
De ruta base		X	X		X	X	X	X																		
De estructura de control		X	X		X	X	X	X																		
Unitaria caja negra																										
De interfaz		X	X		X	X	X	X																		
Partic.de equivalencias		X	X		X	X	X	X																		
De valores límite		X	X		X	X	X	X									X									
Integración																										
Ascendente		X	X		X	X	X	X				X	X			X		X	X							
De validación																										
P01: Análisis sintáctico		X			X																					
P02: Análisis semántico		X	X		X																		X			
P03: A.organiz. información		X	X		X																					
P04: Análisis BBDD		X	X		X									X				X								
P05: Carácteristicas interfaz			X				X	X																		
P06: Mecanismos interfaz			X				X																			
P07: Análisis semántico interfaz			X				X																			
P08: Análisis de la usabilidad			X				X																			
P09: Análisis compatibilidad			X				X																			
P10: Análisis de configuración			X		X			X																		
P11: Análisis vulnerabilidades															X		X									
P12: Análisis de la latencia	X													X												
P13: Análisis escalabilidad	X											X		X	X			X								
P14: Análisis de la IA				X															X							X
De sistema																										
De recuperación									X	X					X											
De esfuerzo				X											X											
De rendimiento	X			X								X	X		X											
De despliegue				X		X					X	X		X	X	X				X	X	X		X	X	

Tabla 28: Matriz de trazabilidad de los requisitos (elaboración propia)

5.3. Métricas

Buscando la mayor calidad del producto resultante, se han diseñado dos métricas que ayudarán a obtener información cuantificable sobre los requisitos obtenidos. De esta forma, se busca asentar el diseño del sistema sobre una base objetiva más sólida (Pressman, 2010).

5.3.1. Grado de cumplimiento

En esta métrica se ha calculado el grado de cumplimiento de la especificación en base a la proporción de requisitos que cumplen las características de corrección, especificidad, completitud, consistencia, verificabilidad, modificabilidad y trazabilidad.

Se tiene:
Rnc = número de requisitos que no cumplen la característica
Rc = número de requisitos que cumplen la característica

Rt = número total de requisitos = Funcionales Obligatorios + Func. Deseables + No Funcionales Obligatorios + No Funcionales Deseables + Empresariales = 4 + 4 + 11 + 1 + 6 = 26

Se calcula la media de los porcentajes individuales, $Q_i = \frac{Rc_i}{Rt}$.

	Rnc	Rc = Rt-Rnc	Q=Rc/Rt
Corrección	1	25	0,96
Especificidad	4	22	0,85
Completitud	6	20	0,77
Consistencia	0	26	1,00
Verificabilidad	3	23	0,88
Modificabilidad	0	26	1,00
Trazabilidad	0	26	1,00
		Media	**0,92**

Tabla 29: Grado de cumplimiento de los requisitos (elaboración propia)

El valor obtenido es bastante cercano a 1.0, lo que indica un alto grado de cumplimiento del modelo de requisitos realizado.

5.3.2. Puntos de función (FP)

Se han usado puntos de función (FP) para predecir el tamaño del sistema. Ayudarán a estimar la complejidad del diseño y el coste en codificación, integración y prueba. Los puntos de función se centran en el tamaño lógico en lugar del tamaño físico (número de líneas de código, LOC) y se basan en la funcionalidad requerida por el usuario. Se definen de la siguiente forma:

- Nº entradas externas (EE): introducidos por usuarios u otras aplicaciones
- Nº salidas externas (SE): páginas mostradas o mensajes informativos
- Nº Consultas Externa (CE): entrada en línea que da como resultado una salida inmediata.
- Nº de archivo lógicos internos (ALI): agrupación de datos dentro de la aplicación mantenidos por entradas externas
- Nº Archivos de interfaz externos (AIE): agrupación de datos fuera de la aplicación que puede usarse dentro.

En la tabla 30 se detallan las estimaciones realizadas para realizar el conteo.

	Optimista	Probable	Pesimista	Estimado	Peso	Conteo FP
Entradas externas (EE)	10	14	20	14	3	54
Salidas externas (SE)	20	27	35	27	2	54
Consultas Externa (CE)	2	2	10	2	5	10
Archivos lógicos internos (ALI)	20	25	38	25	3	75
Archivos de interfaz externos (AIE)	5	6	7	6	2	12
Conteo Total						**205**

Tabla 30: Conteo Puntos de Función (elaboración propia)

Se han calculado los factores de complejidad, siendo 0 (no importante) a 5 (esencial), como se muestra en la tabla 31.

Factor complejidad	Valor
Respaldo y recuperación	5
Comunicaciones de datos	4
Procesamiento distribuido	4
Rendimiento crítico	4
Entorno de operación existente	2
Entrada de datos en línea	1
Transacción de entrada a través de múltiples pantallas	0
Archivos maestros actualizados en línea	1
Valores de dominio de información complejos	4
Procesamiento interno complejo	5
Código diseñado para reutilización	5
Conversión/instalación en diseño	3
Múltiples instalaciones	1
Aplicación diseñada para el cambio	5
Suma Total Fi	**44**

Tabla 31: Factores de complejidad (elaboración propia)

Siguiendo a Pressman, se ha usado la siguiente fórmula para el cálculo del FP estimado:

$$FP_{estimado} = conteo\ total\ x\ [0{,}65 + \left(0{,}01\ x\ \sum F_i \right)]$$

$$FP_{estimado} = 205\ x\ [0{,}65 + (\ 0{,}01\ x\ 44)] = 223{,}45$$

Con la estimación del tamaño lógico del sistema se pueden usar diferentes métricas. A modo de ejemplo se ofrecen las siguientes:

- Productividad promedio $= \frac{FP_{estimado}}{persona\ x\ mes} = \frac{223{,}45}{1\ x\ 2} = 111{,}73\frac{FP}{pm}$
- Calidad promedio $= \frac{nº\ defectos}{FP_{estimado}} = \frac{65}{223{,}45} = 0{,}29\frac{Defectos}{FP}$
- Coste promedio $= \frac{Coste\ Total}{FP_{estimado}} = \frac{6.120{,}00}{223{,}45} = 27{,}39\ \frac{Euros}{FP}$

Estas métricas ayudarán a llevar la planificación del proyecto de una forma más sólida que la basada en estimaciones preliminares. Permitirán realizar comparaciones objetivas entre las diferentes versiones del producto. Adicionalmente, servirán de base para planificar futuros proyectos.

CAPÍTULO 6: RESULTADOS Y CONCLUSIONES

6.1. Resultados

En los apartados anteriores se ha mostrado la configuración de la capa inferior que soporta el sistema, formado por los clusters de Hadoop y Spark, el servidor Kafka, la base de datos PostgreSQL y el servidor web Django, y la implementación de las aplicaciones de minería de datos y web. A continuación, se muestra el resultado obtenido. Para ello es necesario seguir el siguiente orden:

6.1.1. Puesta en marcha de la estructura

Como se vio en el apartado "4.4.3 Actividades de Implementación", se ejecutan los contenedores y se arrancan los servicios de Hadoop, Spark, Delta y Hive. Para confirmar el éxito del proceso, además de las aplicaciones web de cada servicio, se puede utilizar el terminal en cada contenedor.

Dentro del maestro se puede confirmar los procesos de Java que están corriendo con el comando "jps". En la figura 30 se ofrece el resultado obtenido en el contenedor maestro.

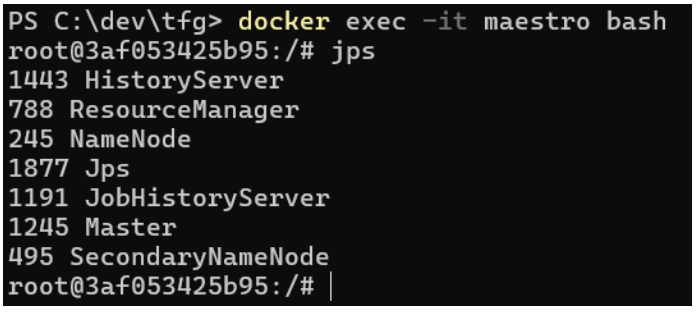


Figura 30: Procesos de Java ejecutándose en el nodo Maestro (elaboración propia)

Puede verse que están funcionando los servicios de HDFS (p.e. NameNode y SecondaryNameNode) y de Spark (p.e. HistoryServer y Master).

Con el comando “hdfs dfsadmin -report” se muestra información del sistema de archivos HDFS. En la figura 31 se muestra un extracto obtenido.

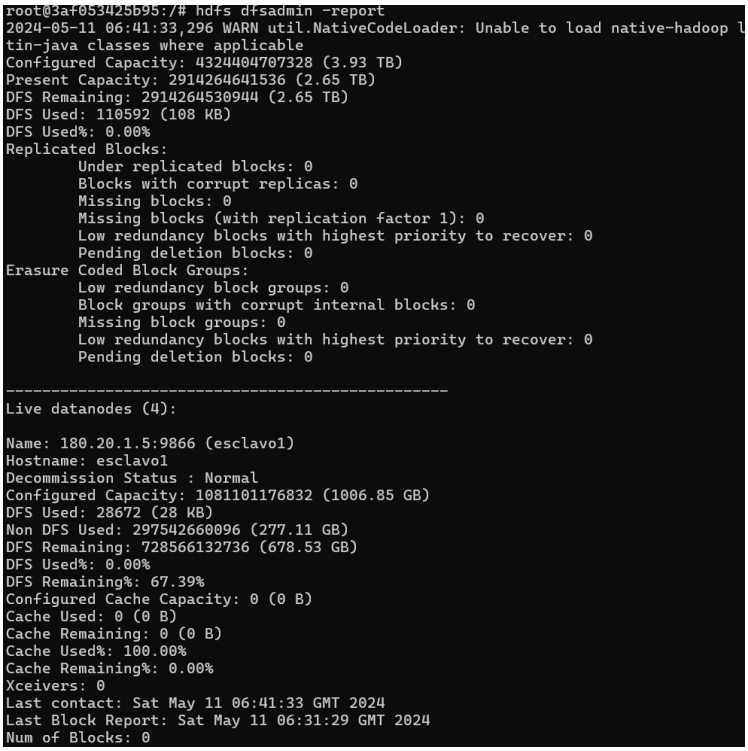


Figura 31: Extracto del informe sobre HDFS en el cluster de Hadoop (elaboración propia)

Se puede comprobar la correcta comunicación de los contenedores dentro de la red de Docker con el comando “ping”. A modo de ejemplo, se muestra en la figura 32 el comando ejecutado desde el nodo maestro.

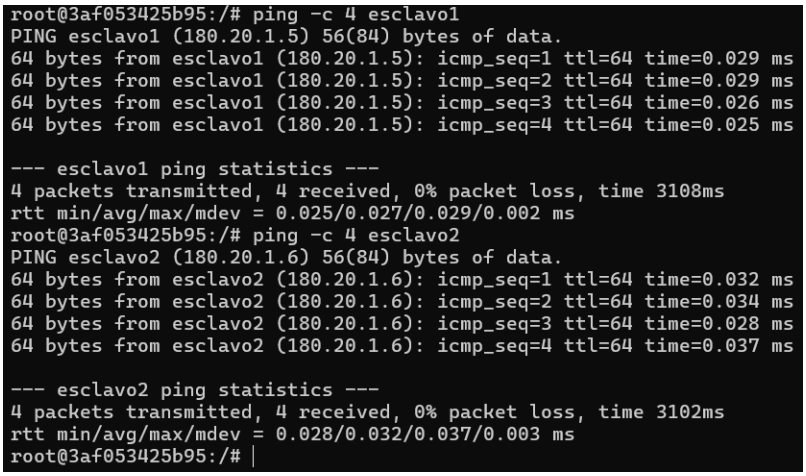


Figura 32: Comprobación de la conexión dentro de la red de Docker

Del mismo modo se puede comprobar, como se muestra en la figura 33, que el servicio del gestor de recursos YARN está activo con el comando “yarn node -list -showDetails”.

```
root@3af053425b95:/# yarn node -list -showDetails
WARNING: YARN_CONF_DIR has been replaced by HADOOP_CONF_DIR. Using value of YARN_CONF_DIR.
2024-05-11 06:48:17,897 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
2024-05-11 06:48:17,921 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to Resource
2024-05-11 06:48:18,084 INFO conf.Configuration: resource-types.xml not found
2024-05-11 06:48:18,084 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
Total Nodes:4
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
  esclavo2:46103                RUNNING     esclavo2:8042                                  0
Detailed Node Information :
        Configured Resources : <memory:1536, vCores:2>
        Allocated Resources : <memory:0, vCores:0>
        Resource Utilization by Node : PMem:11905 MB, VMem:12011 MB, VCores:0.07333334
        Resource Utilization by Containers : PMem:0 MB, VMem:0 MB, VCores:0.0
        Node-Labels :
  esclavo3:35499                RUNNING     esclavo3:8042                                  0
Detailed Node Information :
        Configured Resources : <memory:1536, vCores:2>
        Allocated Resources : <memory:0, vCores:0>
        Resource Utilization by Node : PMem:11905 MB, VMem:12011 MB, VCores:0.07666667
        Resource Utilization by Containers : PMem:0 MB, VMem:0 MB, VCores:0.0
        Node-Labels :
```

Figura 33: Extracto de la información sobre el estado actual del gestor de recursos YARN (elaboración propia)

Para confirmar que la base de datos está creada y las tablas de datos están vacías, se puede usar a la aplicación web pgAdmin. Como ejemplo, en la figura 34 se muestra el contenido de la tabla de datos.

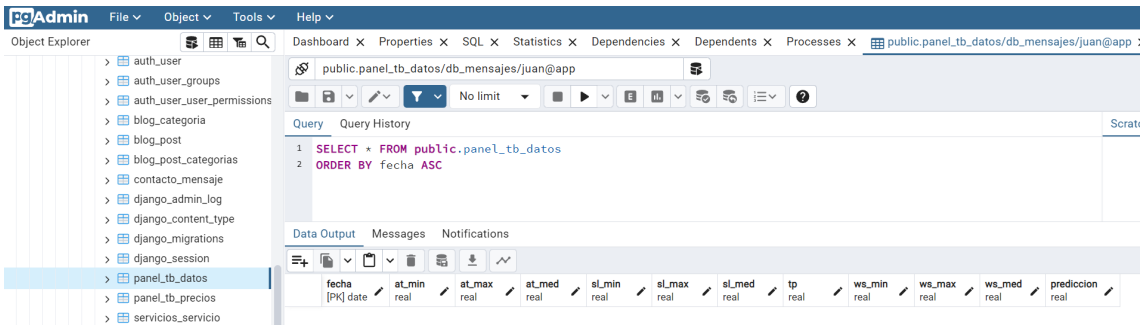


Figura 34: Tabla de datos en base de datos PostgreSQL (elaboración propia)

6.1.2. Ejecución subsistema de entrenamiento

Entrando en el contenedor de aplicaciones se ejecutan los módulos de minería de datos.

El módulo de ETL se ejecuta en el cluster de Spark con el gestor YARN y con Delta Lake habilitado con la siguiente instrucción:

home/hduser/spark/bin/spark-submit --packages io.delta:delta-spark_2.12:3.1.0 --master yarn --deploy-mode client ./codigo/01_etl.py

Este módulo se nutre con los datos de “Climate and energy indicators for Europe from 1979 to present derived from reanalysis” obtenidos del portal web de la Unión Europea Copernicus (Unión Europea, 2024) y por los precios históricos obtenidos del sistema de información de Red Eléctrica Española (Red Eléctrica de España, 2024).

La aplicación lanzada se pone en estado “Running” como puede verse en la figura 35.

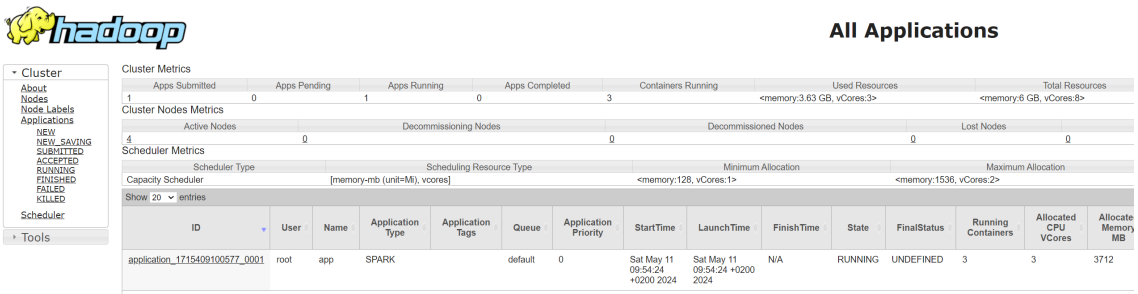


Figura 35: Estado de la aplicación en el cluster de Hadoop (elaboración propia)

Spark divide la aplicación en Jobs y transforma cada Job en un DAGs (Gráfico Acíclico Dirigido). Cada DAG puede estar compuesto por uno o varios Stages. Este proceso lo gestiona el planificador de DAGs (DAGScheluder). Finalmente, YARN gestiona los recursos que necesitará el sistema para realizar las tareas. Se muestra un extracto de ejemplo en la figura 36.

```
24/05/11 07:54:36 INFO SparkContext: Starting job: csv at NativeMethodAccessorImpl.java:0
24/05/11 07:54:36 INFO DAGScheduler: Got job 1 (csv at NativeMethodAccessorImpl.java:0) with 12 output partitions
24/05/11 07:54:36 INFO DAGScheduler: Final stage: ResultStage 1 (csv at NativeMethodAccessorImpl.java:0)
24/05/11 07:54:36 INFO DAGScheduler: Parents of final stage: List()
24/05/11 07:54:36 INFO DAGScheduler: Missing parents: List()
24/05/11 07:54:36 INFO DAGScheduler: Submitting ResultStage 1 (MapPartitionsRDD[9] at csv at NativeMethodAccessorImpl.java:0), which has no m
24/05/11 07:54:36 INFO MemoryStore: Block broadcast_3 stored as values in memory (estimated size 33.1 KiB, free 365.2 MiB)
24/05/11 07:54:36 INFO MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 15.1 KiB, free 365.1 MiB)
24/05/11 07:54:36 INFO BlockManagerInfo: Added broadcast_3_piece0 in memory on ba04d74c206e:34533 (size: 15.1 KiB, free: 366.2 MiB)
24/05/11 07:54:36 INFO SparkContext: Created broadcast 3 from broadcast at DAGScheduler.scala:1585
24/05/11 07:54:36 INFO DAGScheduler: Submitting 12 missing tasks from ResultStage 1 (MapPartitionsRDD[9] at csv at NativeMethodAccessorImpl.j
 7, 8, 9, 10, 11))
24/05/11 07:54:36 INFO YarnScheduler: Adding task set 1.0 with 12 tasks resource profile 0
24/05/11 07:54:36 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1) (esclavo2, executor 2, partition 0, PROCESS_LOCAL, 8254 bytes)
24/05/11 07:54:36 INFO TaskSetManager: Starting task 1.0 in stage 1.0 (TID 2) (esclavo4, executor 1, partition 1, PROCESS_LOCAL, 8254 bytes)
24/05/11 07:54:36 INFO BlockManagerInfo: Added broadcast_3_piece0 in memory on esclavo4:45277 (size: 15.1 KiB, free: 366.2 MiB)
24/05/11 07:54:36 INFO BlockManagerInfo: Added broadcast_3_piece0 in memory on esclavo2:36491 (size: 15.1 KiB, free: 366.3 MiB)
24/05/11 07:54:36 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on esclavo4:45277 (size: 54.4 KiB, free: 366.2 MiB)
24/05/11 07:54:37 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on esclavo2:36491 (size: 54.4 KiB, free: 366.2 MiB)
24/05/11 07:54:39 INFO TaskSetManager: Starting task 2.0 in stage 1.0 (TID 3) (esclavo4, executor 1, partition 2, PROCESS_LOCAL, 8254 bytes)
24/05/11 07:54:39 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 2) in 3643 ms on esclavo4 (executor 1) (1/12)
```

Figura 36: Mensajes informativos durante la ejecución de la aplicación ETL (elaboración propia)

En la aplicación web de Spark puede seguirse la ejecución de la aplicación como muestran las figuras 37, 38 y 39.

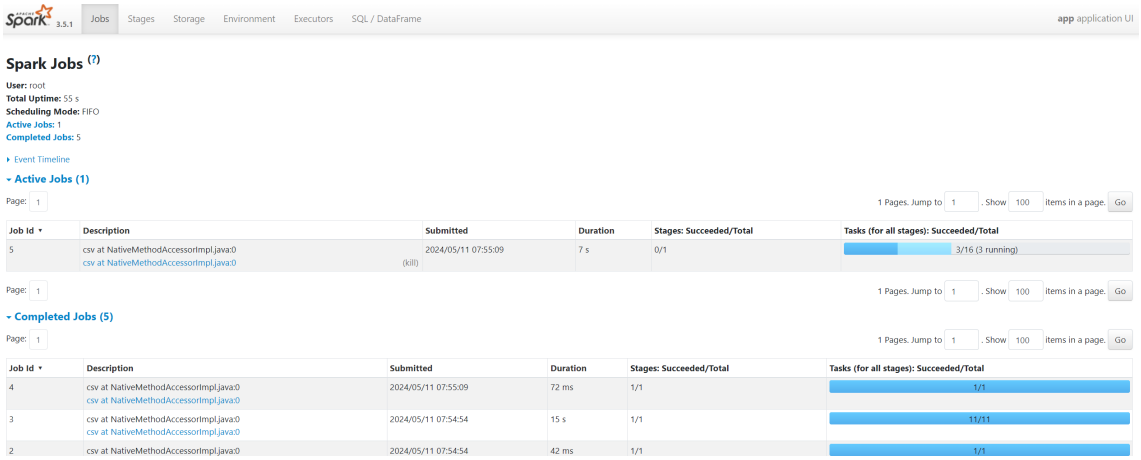


Figura 37: Información sobre los Jobs durante la ejecución de la aplicación ETL (elaboración propia)

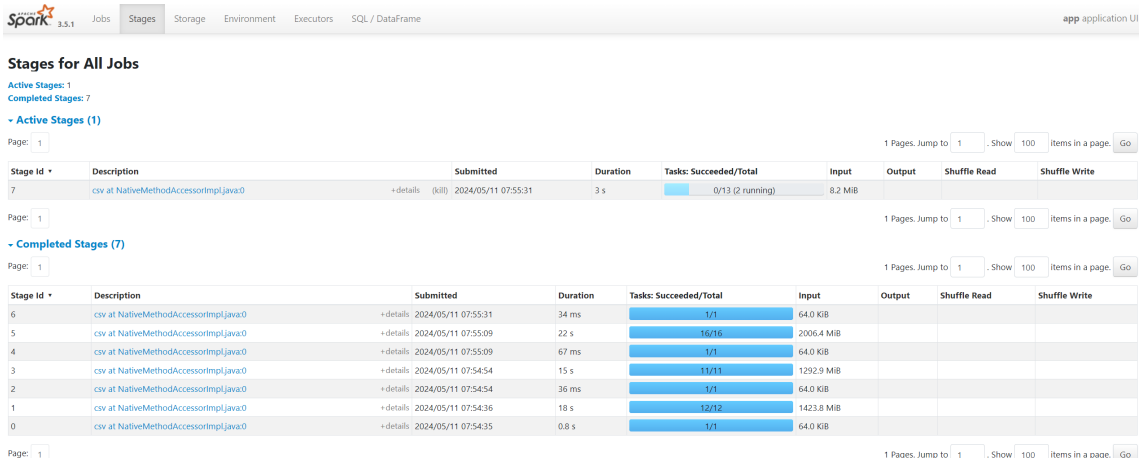


Figura 38: Información sobre los Stages durante la ejecución de la aplicación ETL (elaboración propia)

Una vez finalizada, la aplicación pasa al estado de finalizado exitosamente.

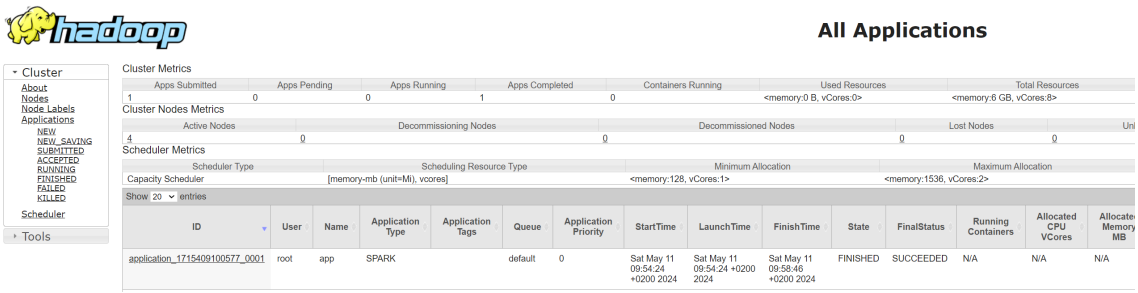


Figura 39: Estado finalizado de la aplicación en el cluster de Hadoop (elaboración propia)

En la figura 40 se muestran los mensajes informativos al parar los procesos y liberarse los recursos utilizados.

```
24/05/11 07:58:46 INFO SparkContext: Invoking stop() from shutdown hook
24/05/11 07:58:46 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/05/11 07:58:46 INFO SparkUI: Stopped Spark web UI at http://ba04d74c206e:4040
24/05/11 07:58:46 INFO YarnClientSchedulerBackend: Interrupting monitor thread
24/05/11 07:58:46 INFO YarnClientSchedulerBackend: Shutting down all executors
24/05/11 07:58:46 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
24/05/11 07:58:46 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
24/05/11 07:58:46 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/05/11 07:58:46 INFO MemoryStore: MemoryStore cleared
24/05/11 07:58:46 INFO BlockManager: BlockManager stopped
24/05/11 07:58:46 INFO BlockManagerMaster: BlockManagerMaster stopped
24/05/11 07:58:46 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/05/11 07:58:46 INFO SparkContext: Successfully stopped SparkContext
24/05/11 07:58:46 INFO ShutdownHookManager: Shutdown hook called
24/05/11 07:58:46 INFO ShutdownHookManager: Deleting directory /tmp/spark-3fbfbb91-f7a5-4677-b07b-3c3a4e518161/py
24/05/11 07:58:46 INFO ShutdownHookManager: Deleting directory /tmp/spark-3fbfbb91-f7a5-4677-b07b-3c3a4e518161
24/05/11 07:58:46 INFO ShutdownHookManager: Deleting directory /tmp/spark-72e85a20-9f4b-4fa2-a028-991dcd4ae768
```

Figura 40: Mensajes informativos a la finalización de la aplicación ETL (elaboración propia)

En este momento se ha realizado la carga de los datos en el sistema de archivos como puede verse en la figura 41.

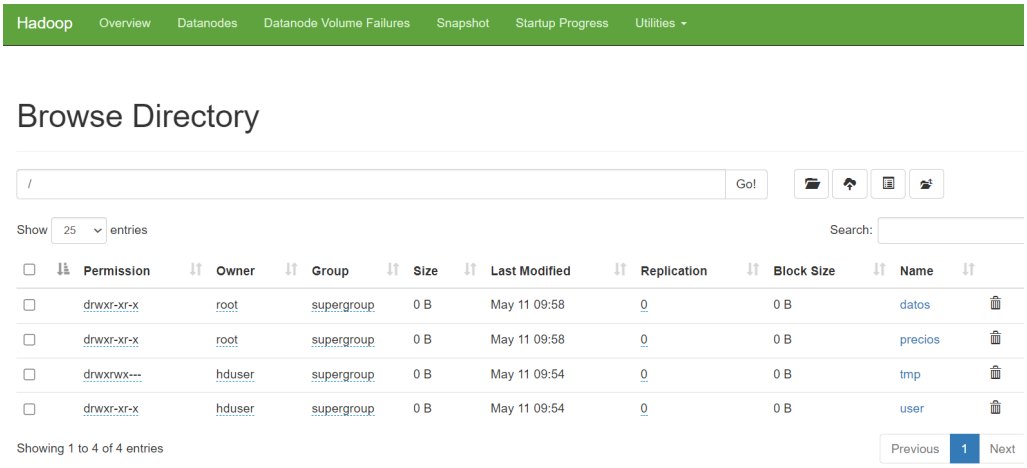


Figura 41: Archivos almacenados en HDFS (elaboración propia)

Cada bloque de información se almacena replicado tal y como se estableció en la configuración. Como ejemplo, se muestra en la figura 42 detalle de un bloque.

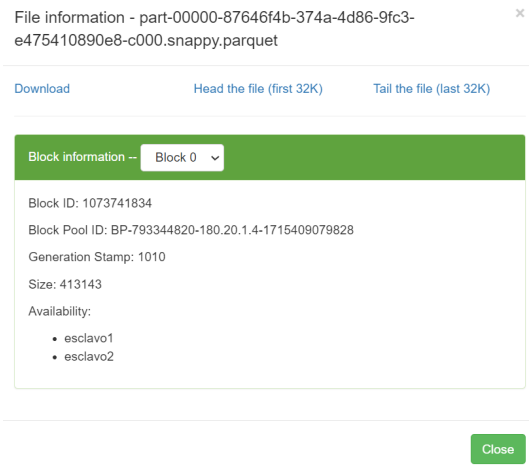


Figura 42: Nodos donde está disponible un bloque de información en HDFS (elaboración propia)

Seguidamente se ejecuta el módulo de entrenamiento del mismo modo que el anterior. El modelo se basará en los datos de las variables meteorológicas para realizar las estimaciones. Se ha usado un *pipeline* de 4 pasos, que incluyen el estandarizado de varias variables y el uso de un modelo de regresión lineal. Se usa validación cruzada con varios rangos de hiperparámetros y se escoge el modelo que ofrece mejor resultado. Se obtiene el siguiente resultado:

```
Raíz Error Cuadrático Medio(RMSE) en datos de evaluación = 8.10375 euros/MWh
```

Se guarda el modelo en HDFS para su posterior uso en el sistema predictor y se genera el gráfico de dispersión con los resultados obtenidos. Se muestra ejemplo en la figura 43.

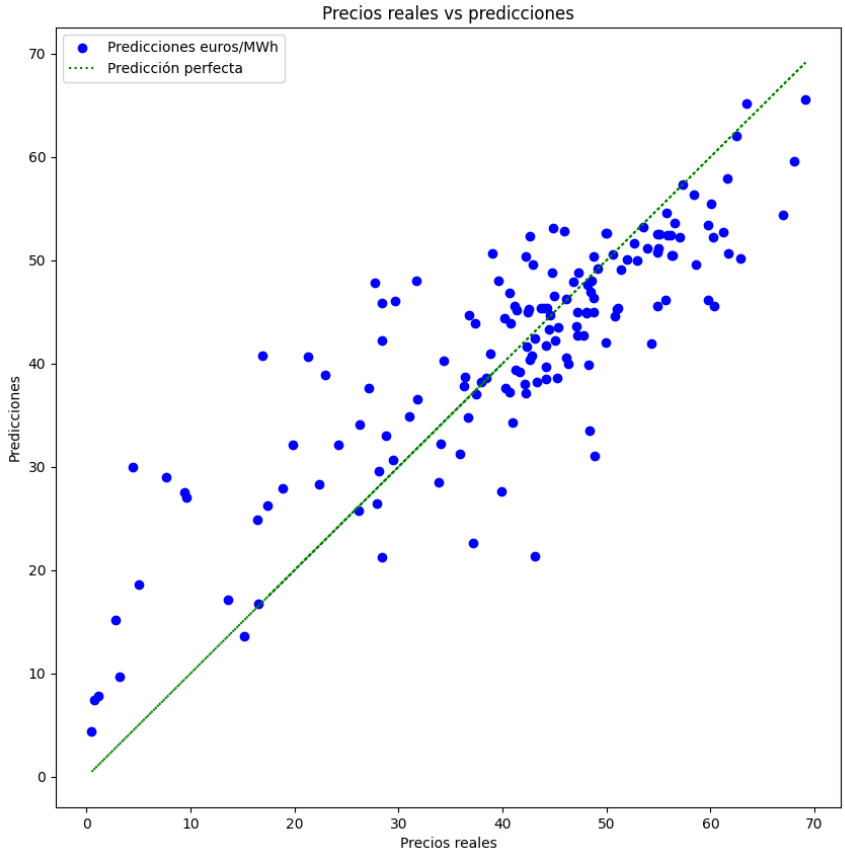


Figura 43: Gráfico de dispersión de los precios reales y las predicciones realizadas (elaboración propia)

6.1.3. Ejecución subsistema predictor

En este sistema es necesario ejecutar cada módulo de forma separada y simultánea.
Los módulos extractores obtienen los datos usando las API de cada web, previa solicitud de la API Key personal. En la web de AEMET se obtienen los datos meteorológicos actuales en dos pasos: en el primero se solicita las direcciones de los datos y metadatos y en el segundo se extraen los datos en formato json. En la web de ESIOS se obtiene el precio de la electricidad en formato json también. Después, se lanzan los mensajes al servidor Kafka con temas distintos. En la figura 44, aparecen los mensajes de dos terminales ejecutando cada uno de los módulos.

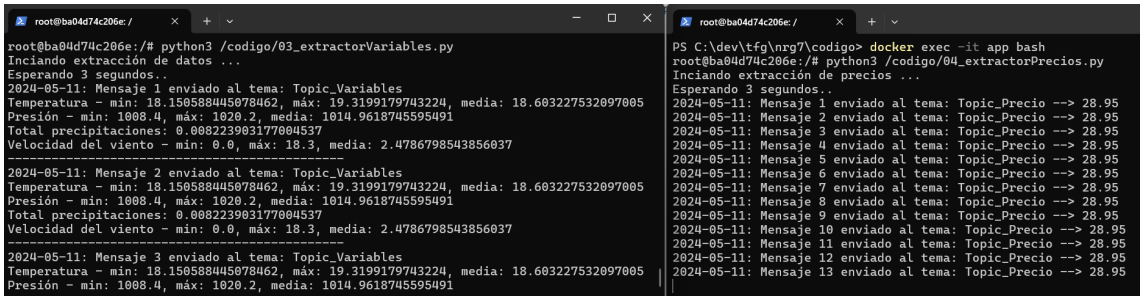


Figura 44: Mensajes durante la ejecución de los dos extractores (elaboración propia)

El módulo predictor recupera los mensajes del servidor Kafka, adapta los datos, utiliza el modelo guardado anteriormente para realizar las predicciones y guarda todos los datos en la base de datos PostgreSQL para su posterior visualización en la web. Un extracto del resultado puede verse en la figura 45.

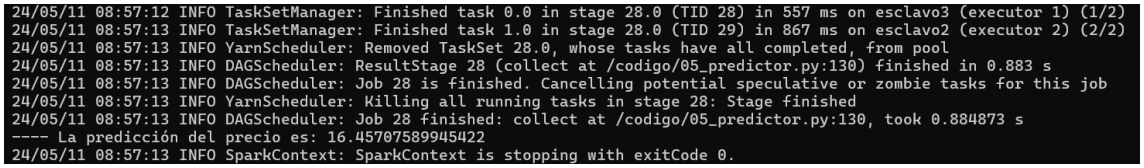


Figura 45: Extracto de los mensajes con el resultado de la predicción (elaboración propia)

Como ejemplo, en la figura 46 se muestra el contenido actualizado de la tabla de datos.

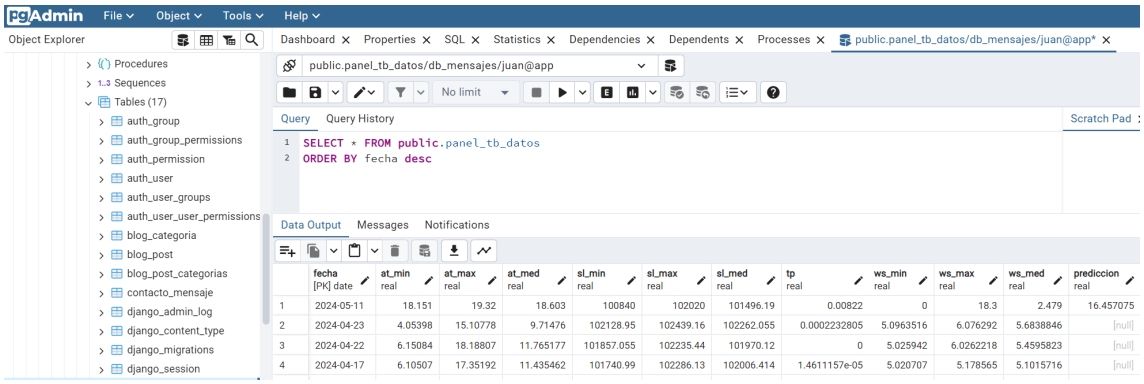


Figura 46: Tabla de datos con la información cargada (elaboración propia)

6.1.4. Ejecución subsistema web

En el contenedor web se ejecuta el servidor web como se muestra en la figura 47.

```
PS C:\dev\tfg\nrg7\codigo> docker exec -it web bash
root@0bf8e4b8b570:/# cd proyectoweb
root@0bf8e4b8b570:/proyectoweb# python3 manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 11, 2024 - 09:02:34
Django version 4.2.10, using settings 'proyectoweb.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Figura 47: Ejecución del servidor web Django (elaboración propia)

Accediendo al puerto 8000 se muestra la página de inicio de la web (figura 48).

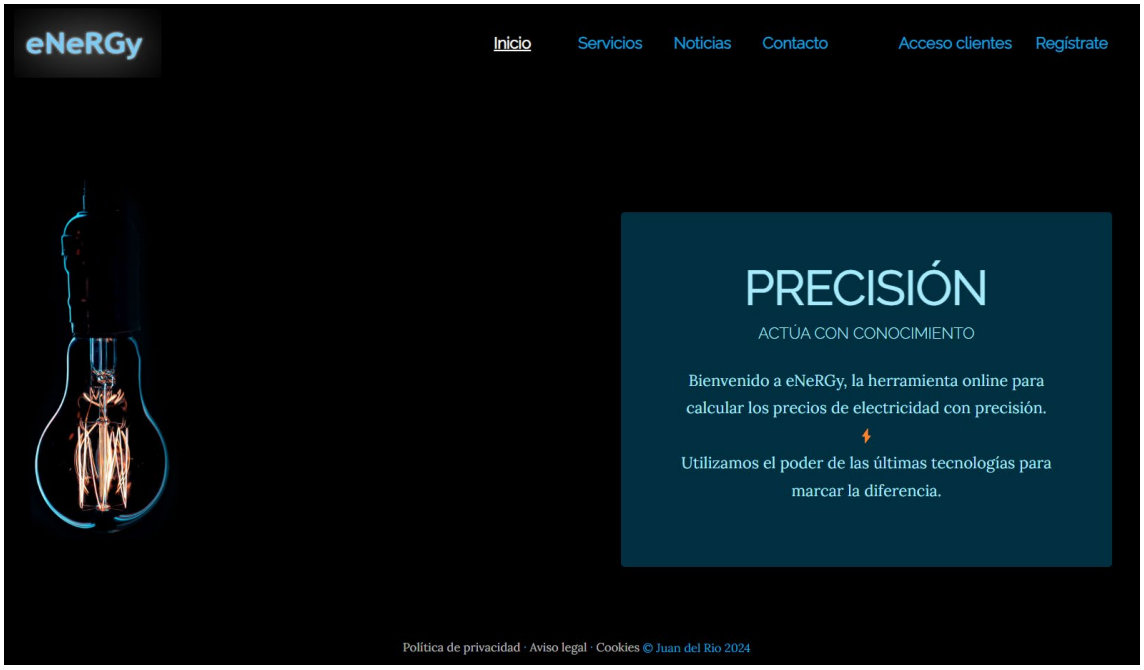


Figura 48: Página de inicio de la web (elaboración propia)

Con un usuario registrado se puede acceder a la zona de clientes, que muestra el precio estimado de la electricidad y varios gráficos con la evolución de las variables. Esta página aparece en la figura 49.

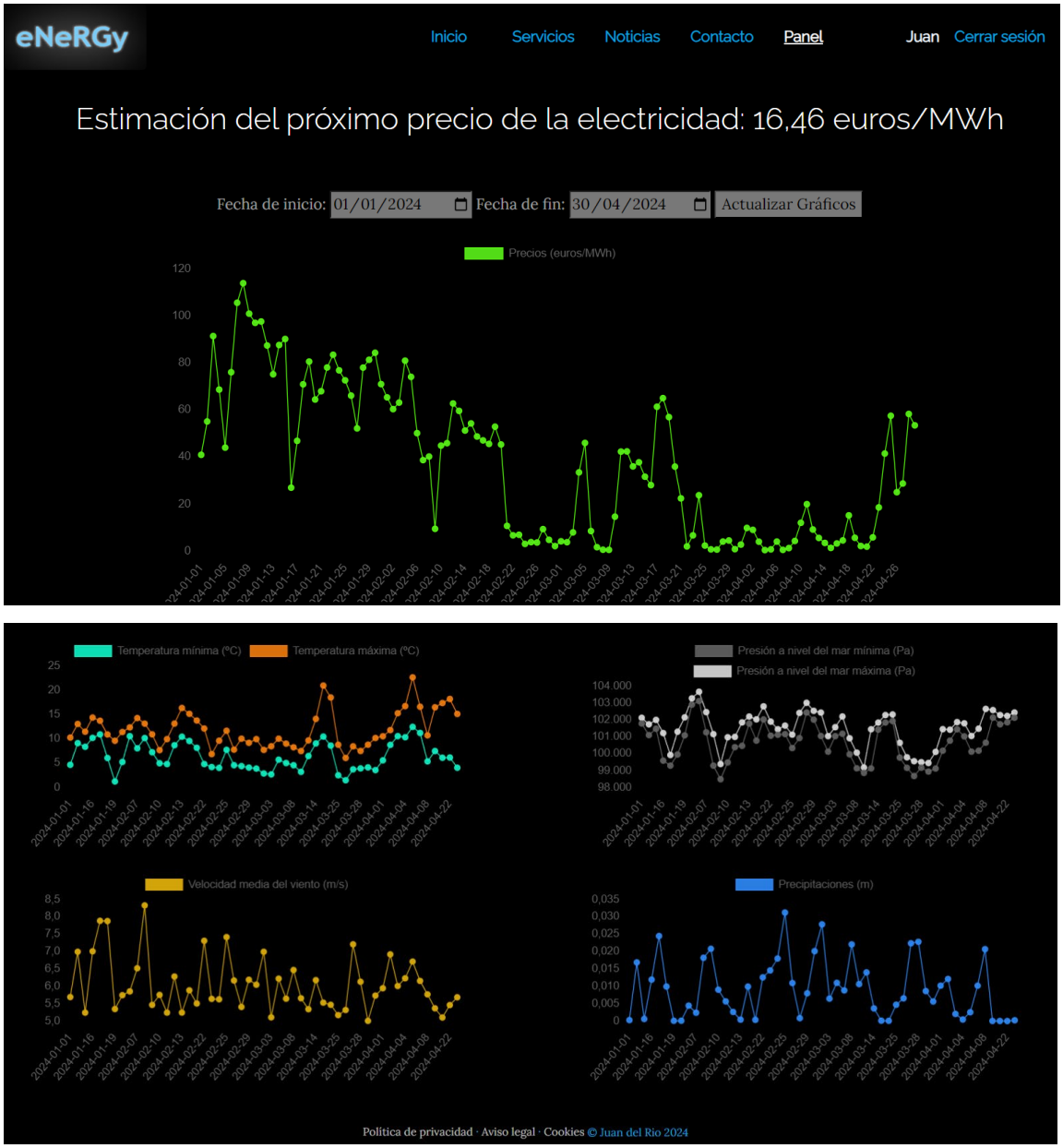


Figura 49: Página exclusiva para clientes en la web (elaboración propia)

6.2. Optimización del sistema de entrenamiento

Con el objetivo de encontrar el modelo que realice las mejores predicciones, se ha estudiado el resultado con los diferentes algoritmos disponibles de la librería de Spark en

la actualidad (Spark 3.5.1) con los hiperpárametros en la validación cruzada. La tabla 32 detalla los valores utilizados en este trabajo.

Modelo	Hiperparámetros	
Linear Regressor	maxIter	10, 50, 100
	regParam	0.0, 0.01, 0.1
	elasticNetParam	0.0, 0.5, 1.0
	solver	'auto', 'l-bfgs', 'normal'
Generalized Linear Regressor	maxIter	10, 50, 100
	regParam	0.0, 0.01, 0.1
Gradient-boosted tree regressor	maxDepth	3, 5, 7
	maxBins	32, 64, 128
	maxIter	10, 20, 30
	stepSize	0.1, 0.05
	subsamplingRate	0.8, 1.0
Random Forest Regressor	maxDepth	5, 10, 15
	numTrees	10, 20, 30
Decision Tree Regressor	maxDepth	3, 5, 7
	maxBins	32, 64, 128
	minInstancesPerNode	1, 5, 10
	minInfoGain	0.0, 0.1, 0.2

Tabla 32: Hiperpárametros para la validación cruzada (elaboración propia)

Los resultados obtenidos se muestran en la tabla 33 y figura 50.

Modelo	Raíz error cuadrático medio (RMSE) € /MWh	Tiempo de entrenamiento (segundos)
Linear Regressor	8,11004	173
Generalized Linear Regressor	8,10375	45
Gradient-boosted tree regressor	9,34170	445
Random Forest Regressor	8,60448	62
Decision Tree Regressor	10,27710	145

Tabla 33: Resultados obtenidos por cada modelo (elaboración propia)

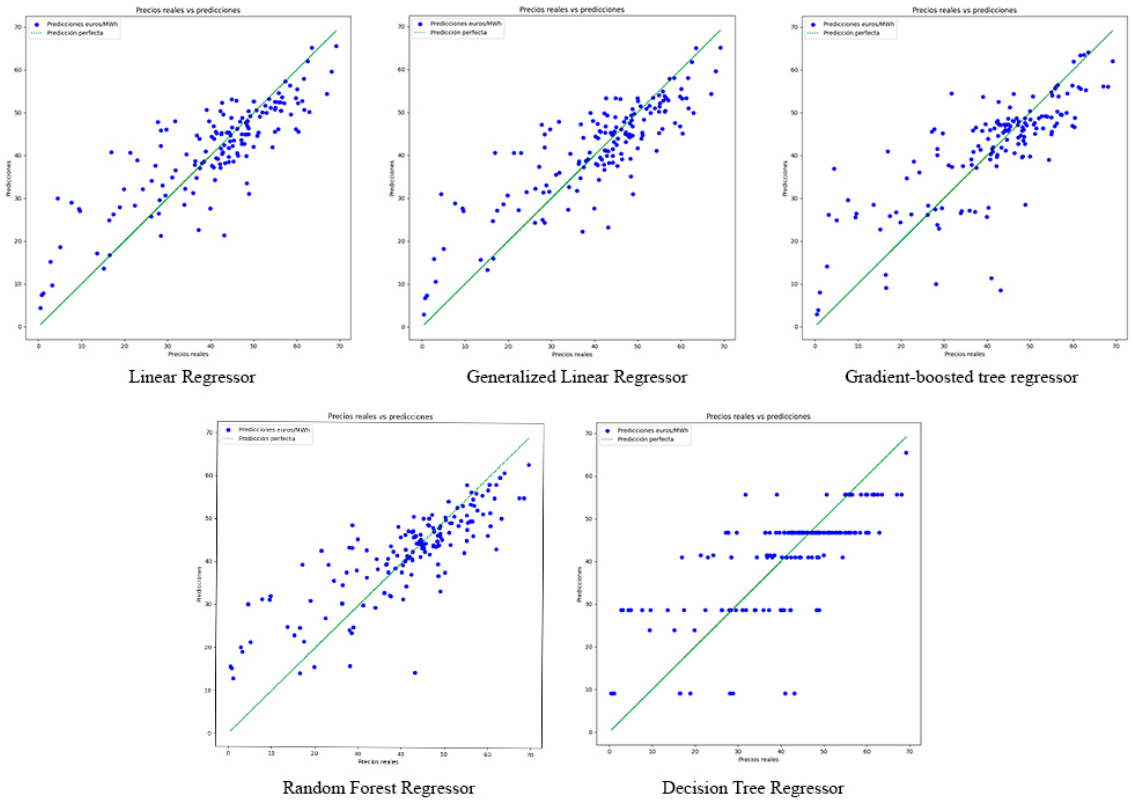


Figura 50: Gráficos de dispersión obtenidos por cada modelo de aprendizaje (elaboración propia)

Los modelos de regresión lineal y el Random Forest Regressor obtienen las estimaciones más precisas, siendo el algoritmo de regresión lineal generalizado el que obtiene mejor resultado tanto en error como en tiempo de entrenamiento.

Por otro lado, la librería de aprendizaje automático utilizada en este trabajo, Spark MLlib, aprovecha el escalado de su arquitectura para acelerar los procesos. Sin embargo, no está adaptada para utilizar la intensiva paralelización de las GPUs (Unidades de Procesamiento Gráfico) o TPUs (Unidades de Procesamiento Tensorial). La empresa Nvidia ha creado una biblioteca llamada Rapids para solventar este inconveniente (Nvidia, 2024). A pesar de no haberse usado dicha librería, se ha observado que durante la ejecución del entrenamiento el sistema ha utilizado un 27% de la capacidad de la GPU, como se muestra en la figura 51.

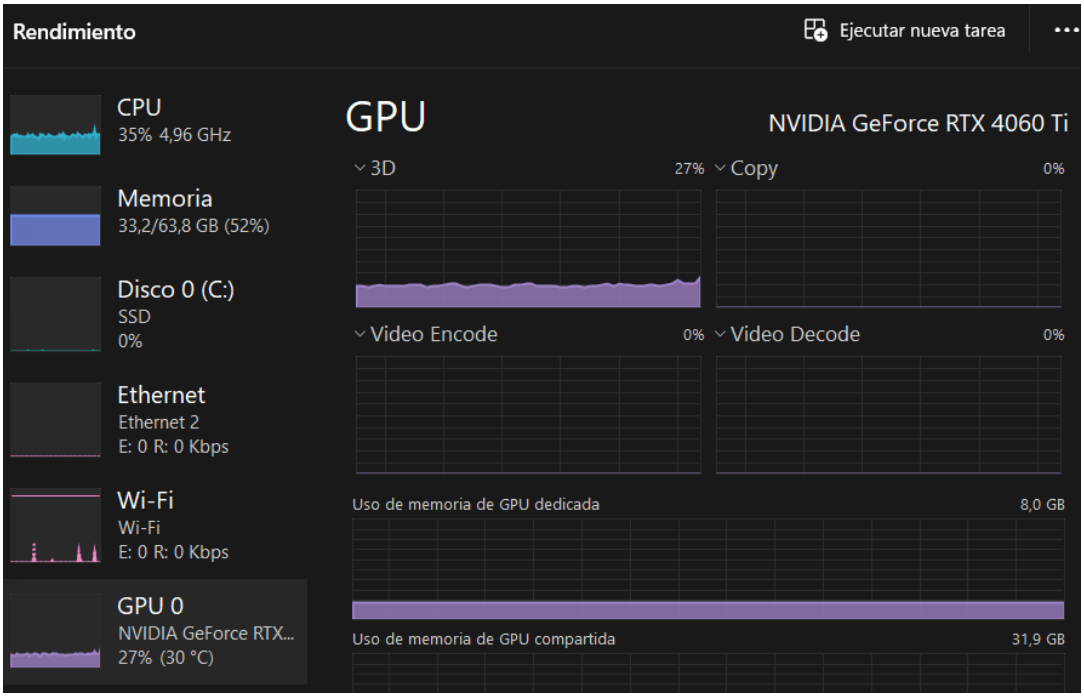


Figura 51: Estado de los recursos del sistema al ejecutar el módulo de entrenamiento (elaboración propia)

6.3. Conclusiones

En este trabajo se ha desarrollado un prototipo con el objetivo de servir de base para una futura aplicación comercial. El resultado ha sido una aplicación web funcional con apartados como un blog de noticias, una sección informativa de servicios, otra sección de contacto y un área exclusiva para clientes donde se muestra la estimación del precio de la electricidad y gráficos de la evolución de distintas variables usadas en el modelo predictivo.

El sistema construido tiene las siguientes características:

- Escalabilidad tanto para almacenamiento como procesamiento, por el uso de la arquitectura de Hadoop y Spark. De esta forma será capaz de tratar datos de tamaños muy grandes, del orden de petabytes, a través del uso de una red de servidores.

- Sistema de almacenamiento con resistencia a pérdidas de información gracias a la replicación de la información de HDFS.
- Sistema de almacenamiento que admite todo tipo de datos (estructurados, semi estructurados y no estructurados), que tiene transacciones ACID y acceso versiones anteriores por la incorporación de Delta Lake.
- Fácil despliegue tanto *on-premises* como en la nube, por la implementación en contenedores Docker.
- Un sistema predictivo que ofrece resultados esperanzadores.

Sin embargo, el desarrollo y evaluación de este prototipo se ha visto limitado por los siguientes condicionantes:

- La complejidad del problema de estimar los precios de la electricidad: por un lado, la dificultad de modelizar factores como la situación política o las regulaciones españolas y europeas. Por otro lado, la dificultad de encontrar datos gratuitos (abiertos) de variables como la demanda de energía, precios del gas natural y carbón, los derechos de emisión de CO2, los impuestos o las subvenciones concedidas.
- El ser ejecutado en un equipo doméstico no ha permitido comprobar el incremento de rendimiento por la paralelización que se lograría con una red de servidores.
- El uso de la librería de Spark no ha permitido explorar otras opciones como las redes neuronales artificiales, debido a la escasa variedad de algoritmos de aprendizaje automático que tiene. Además, no ha podido usarse la potencia completa de la computación por GPU. Librerías de aprendizaje automático que son muy populares como TensorFlow o PyTorch permiten lo anterior.

Por lo tanto, existe un campo bastante amplio para futuras mejoras como:

- La optimización de la configuración de los clusters de Hadoop y Spark con el ajuste de las decenas de parámetros que hay. Por ejemplo: el tamaño mínimo de

los bloques, el número máximo de bloques por archivo, el factor de replicación, número de núcleos por nodo o el tamaño de la memoria por nodo.

- La mejora del sistema predictivo con la incorporación de más variables de entrada y un ajuste fino del modelo. Además, podría considerase el uso de otras librerías usando alguna forma de streaming o procesamiento por lotes.
- La ampliación del periodo de estimación del precio a semanas y meses en el futuro.
- La finalización de todos los apartados de la web y la ampliación de los servicios ofertados.

Por último, cabe resaltar lo interesante que ha resultado adentrarse en el estudio de todas las materias que ha tocado este trabajo. Se ha puesto en evidencia que cada una merecería mucha más atención y profundización.

BIBLIOGRAFÍA

Ahmed, N., Barczak, A. L. C., & M. A., & S. T. (2021). An Enhanced Parallelisation Model for Performance Prediction of Apache Spark on a Multinode Hadoop Cluster. *Big Data and Cognitive Computing* 5, No. 4: 65. https://doi.org/10.3390/bdcc5040065

AleaSoft. (2024). *About*. Recuperado Marzo 7, 2024, de https://aleasoft.com/

Amazon. (2024). *Amazon Web Services*. Recuperado Marzo 7, 2024, de https://aws.amazon.com/

Apache Software Foundation. (2024, Marzo 20). *About*. Recuperado Marzo 7, 2024, de https://www.apache.org/

Balusamy, B., Abirami, R. N., Kadry, S., & Gandomi, A. H. (2021). *Big data: Concepts, technology, and architecture*. John Wiley & Sons, Incorporated.

Chambers, B y Zaharia, M. (2018). *Spark: The Definitive Guide*. O'Reilly Media.

Comisión Nacional de los Mercados y la Competencia. (2024, Marzo 10). *Listado de Distribuidoras de electricidad*. https://sede.cnmc.gob.es/listado/censo/1

Comisión Nacional de los Mercados y la Competencia. (2024, Marzo 10). *Listado de Comercializadoras de electricidad*. https://sede.cnmc.gob.es/listado/censo/2

Comisión Nacional de los Mercados y la Competencia. (2021, Mayo 6). *Reglas de funcionamiento de los mercados diario e intradiario de energía eléctrica para su*

adaptación de los límites de oferta a los límites de casación europeos. https://www.boe.es/buscar/doc.php?id=BOE-A-2021-8362

Comisión Nacional de los Mercados y la Competencia. (2022, Marzo 24). *Boletín anual de mercados a plazo de energía eléctrica en España (balance 2021)*. https://www.boe.es/buscar/doc.php?id=BOE-A-2021-8362

Dardamanis, A., & Δαρδαμάνης, A. (2022). *Evaluation of machine learning models for predicting the system marginal price of an electricity system – italian SMP day-ahead forecasting* . https://doi.org/10.26267/unipi_dione/1944

Dean, J. (2014). *Big data, data mining, and machine learning: Value creation for business leaders and practitioners*. John Wiley & Sons, Incorporated.

Decide. (2024). *Quienes somos*. Recuperado Marzo 7, 2024, de https://decidesoluciones.es/

Enseñat Saavedra, I. (2022, Junio). *El mercado eléctrico mayorista español* TFG Facultad de Ciencias Económicas y Empresariales de la Universidad Pontificia Comillas. https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/56428/TFG-%20Ensenat%20Saavedra%2C%20Isabel.pdf

Fayyad, Usama & Piatetsky-Shapiro, Gregory & Smyth, Padhraic. (1996). *Knowledge Discovery and Data Mining: Towards a Unifying Framework*. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. American Association for Artificial Intelligence https://cdn.aaai.org/KDD/1996/KDD96-014.pdf

Feng, Wenqiang. (2021). *Learning Apache Spark with Python*. https://runawayhorse001.github.io/LearningApacheSpark/pyspark.pdf

Ferré, X. (2015). *Interacción persona-ordenador*. Madrid: Ediciones CEF.

Git. (2024, Marzo 29). https://git-scm.com/

GitHub. (2024, Marzo 29). https://github.com/

Gobierno de España (2024, Marzo 29). *MAGERIT – versión 3.0. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información.* https://administracionelectronica.gob.es/pae_Home/ca/pae_Documentacion/pae_Metodolog/pae_Magerit.html

Grigoryan, H. (2021, Noviembre). *Electricity consumption prediction using energy data, socio-economic and weather indicators. A case study of spain.* https://ieeexplore.ieee.org/document/9646220

Guillermo Castillo Martín, G. (2021 Julio). *Sistema para la extracción del cono cimiento sobre los efectos adversos en las vacunas del COVID 19* TFG Universidad a Distancia de Madrid. https://udimundus.udima.es/bitstream/handle/20.500.12226/1140/guillermo-castillomartin_TFG_Final_20062021.pdf

Google. (2024, Marzo 20). *Google Kubernetes Engine (GKE)*. https://cloud.google.com/

Herrera-Izquierdo, L., & Grob, M. (2017). A performance evaluation between docker container and virtual machines in cloud computing architectures. *Maskana*, 8, 127–133. Recuperado de https://publicaciones.ucuenca.edu.ec/ojs/index.php/maskana/article/view/1457

Iberdrola. (2024). *Contrato de compraventa de energía*. Recuperado Marzo 10, 2024, de https://www.iberdrola.com/conocenos/contrato-ppa-energia

Independent Commodity Intelligence Service. (2024). *About ICIS*. Recuperado Marzo 7, 2024, de https://www.icis.com/

Instituto de Ingenieros Eléctricos y Electrónicos. (1997). *IEEE Std 1074-1997. Standard for Developing Software Life Cycle Processes* https://standards.ieee.org/ieee/1074/1606/

Jankatti, S., Raghavendra, B. K., Raghavendra, S., & Meenakshi, M. (2020). Performance evaluation of Map-reduce jar pig hive and spark with machine learning using big data. *International Journal of Electrical and Computer Engineering*, 10(4), 3811-3818. https://www.proquest.com/scholarly-journals/performance-evaluation-map-reduce-jar-pig-hive/docview/2384919347/se-2

Jefatura de Estado. (2013, Junio 4). *Ley 3/2013 de creación de la Comisión Nacional de los Mercados y la Competencia*. DO BOE 134 https://www.boe.es/buscar/act.php?id=BOE-A-2013-5940

Jefatura de Estado. (2022, Mayo 13). *Real Decreto-ley 10/2022 por el que se establece con carácter temporal un mecanismo de ajuste de costes de producción para la reducción del precio de la electricidad en el mercado mayorista*. DO BOE 115. https://www.boe.es/buscar/act.php?id=BOE-A-2022-7843

Jin-young, C., Cho, M., & Kim, J. (2021). Employing vertical elasticity for efficient big data processing in container-based cloud environments. *Applied Sciences*, 11(13), 6200. https://doi.org/10.3390/app11136200

Kumar, A., Goswami, M. (2023). Performance comparison of instrument automation pipelines using different programming languages. *Scientific Reports* 13, 18579. https://doi.org/10.1038/s41598-023-45849-y

Lara, J. A. (2014). *Minería de datos*. Ed. CEF http://hdl.handle.net/20.500.12226/576

Lee, D., Das, T. y Jainwal, V.(2022). Delta Lake: The Definitive Guide. Ed. O'Reilly Media https://delta.io

Mercado Ibérico de Electricidad. (2024, Marzo 10). *Consejo de Reguladores: Miembros.* https://www.mibel.com/es/consejo-de-reguladores/miembros/

Messias, M., & Rufino, J. (2021*). Electricity price forecasting for MIBEL market: A machine learning approach.*

Microsoft. (2024). *Azure*. Recuperado Marzo 20, 2024, de https://azure.microsoft.com

Ministerio de Industria, Energía y Turismo. (2014, Marzo 28). *Real Decreto 216/2014 por el que se establece la metodología de cálculo de los precios voluntarios para el pequeño consumidor de energía eléctrica y su régimen jurídico de contratación.* https://www.boe.es/diario_boe/txt.php?id=BOE-A-2014-3376

Ministerio de la Transición Ecológica y el Reto Demográfico. (2023 Diciembre). *Balance energético de España 2021-2022.* https://www.miteco.gob.es/es/energia/estrategia-normativa/balances/balances.html

Ministerio de la Transición Ecológica y el Reto Demográfico. (2024). *El comercio de derechos de emisión.* Recuperado Marzo 10, 2024 de https://www.miteco.gob.es/es/cambio-climatico/temas/comercio-de-derechos-de-emision/que-es-el-comercio-de-derechos-de-emision.html

Navarro, F. (2009). *Mercados de futuros. Ed.* El Cid Editor. https://elibro.net/es/ereader/udima/29294

Nvidia (2024), Apache Spark acelerado por GPU. Recuperado Mayo 8, 2024, de https://www.nvidia.com/es-es/deep-learning-ai/solutions/data-science/apache-spark-3/

Olson, D. L., & Olson, D. L. (2018). *Data mining models*, second edition. Business Expert Press.

Operador del Mercado Ibérico de Energía. (2024). *Resultados del Mercado*. Recuperado Mayo 10, 2024, de https://www.omie.es/

Operador del Mercado Ibérico de Energía. (2024). *Mercado de electricidad*. Recuperado Mayo 10, 2024, de https://www.omie.es/es/mercado-de-electricidad

Operador del Mercado Ibérico de Energía. (2024). *Sobre nosotros*. Recuperado Mayo 10, 2024, de https://www.omie.es/es/sobre-nosotros

Organización Internacional de Normalización. (2017). *ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes* https://www.iso.org/es/contents/data/standard/06/37/63712.html

Ortigosa R., Vázquez R. (2009). *Manual de Ingeniería del Software.* Ed: UDIMA.

Osei-Bryson, K., & Barclay, C. (Eds.). (2015). *Knowledge discovery process and methods to enhance organizational performance*. Auerbach Publishers, Incorporated.

Özden-Schilling, C. (2021*). The current economy : Electricity markets and techno-economics. Stanford University Press*. https://www.proquest.com/legacydocview/EBC/6551397?accountid=139267

Pressman, R.S. (2010) "Ingeniería del Software. Un enfoque práctico 7ª Ed,". Ed: McGraw-Hill.

QuantRisk. (2024). *About us*. Recuperado Mayo 7, 2024 de https://quantrisk.com

Red Eléctrica de España. (2024). *Precio mercado spot diario España*. E-sios. https://www.esios.ree.es/es/analisis/600?geoids=3&vis=1&start_date=12-05-2024T00%3A00&end_date=12-05-2024T23%3A55&compare_start_date=11-05-2024&groupby=day

Rubio, J.L. (2021) *Manual de Gestión de Proyectos*. Ed CEF

Santos, M. Y. (2020). *Big data : Concepts, warehousing, and analytics*. River Publishers.

SAS Help Center. (2024, Marzo 20). *Introduction to SEMMA* https://documentation.sas.com/doc/en/emref/15.2/n061bzurmej4j3n1jnj8bbjjm1a2.htm

Shih, W., Yang, C., Ranjan, R., & Chiang, C. (2021). Implementation and evaluation of a container management platform on docker: Hadoop deployment as an example. *Cluster Computing*, 24(4), 3421-3430. https://doi.org/10.1007/s10586-021-03337-w

Sixsigma Institute, (2024). *What is Six Sigma Methodology?* Recuperado Mayo 20, 2024 de https://www.sixsigma-institute.org/What_Is_Six_Sigma.php

Soysal, O. A., & Soysal, H. S. (2020). *Energy for sustainable society : From resources to users.* John Wiley & Sons, Incorporated.

Thaker, P. R. (2022). *Efficient Remote Memory for Parallel and Distributed Data Analytics* . https://www.proquest.com/dissertations-theses/efficient-remote-memory-parallel-distributed-data/docview/2734699337/se-2

Unión Europea. (2024). *Climate and energy indicators for Europe from 1979 to present derived from reanalysis*. Copernicus Climate Change Service. https://cds.climate.copernicus.eu/cdsapp#!/dataset/sis-energy-derived-reanalysis?tab=form

White, T. (2015). *Hadoop: The Definitive Guide, Fourth Edition*. Ed. O'Reilly Media

Yao, J. (2021 Febrero). *Electricity consumption and temperature: Evidence from satellite data.* International Monetary Fund. https://www.imf.org/en/Publications/WP/Issues/2021/02/05/Electricity-Consumption-and-Temperature-Evidence-from-Satellite-Data-50031

Anexos

Anexo A1. Casos de Uso

CU01: Consultar información pública

Actor Principal: Usuario anónimo online

Personal Involucrado e Intereses:

- Usuario anónimo: quiere consultar información que le es de interés sobre evolución de variables relativas a la energía.
- Gerente del contratista: quiere que los usuarios anónimos se sientan atraídos por la información pública que ofrece y conozcan los servicios de pago de la empresa.
- Ingeniero de software: quiere que los usuarios no tengan dificultades técnicas para acceder a la web y a su información pública.
- Científico de datos: quiere que los usuarios accedan a información fiable y de valor.

Precondiciones: el usuario ha conocido de la existencia del portal a través de un buscador o publicidad.

Garantías de éxito (Postcondiciones): el usuario sale del portal o accede a otras secciones.

Escenario principal de éxito (o Flujo Básico):

1- El USUARIO ANÓNIMO accede al portal web.

2- El USUARIO ANÓNIMO accede a la información pública.

3- El USUARIO ANÓNIMO sale del portal.

Extensiones (o Flujos Alternativos):

2 – No se muestra la información

2.a- El USUARIO ANÓNIMO sale del portal

2.b- El USUARIO ANÓNIMO escribe un correo electrónico al servicio técnico.

3- El USUARIO ANÓNIMO accede a la sección de tarifas.

Requisitos especiales: Interfaz disponible en los siguientes idiomas: Español, Inglés, Francés, Portugués y los idiomas regionales de España.
Lista de tecnología y variaciones de datos: La información puede mostrar los precios en diferentes monedas.
Frecuencia: las solicitudes de consulta pueden ser continuas.
Temas abiertos: ¿Se va a proporcionar un chatbot para solucionar consultas o para atraer al usuario a los servicios de pago?

CU02: Consultar precio servicios

Actor Principal: Usuario anónimo online
Personal Involucrado e Intereses:

- Usuario anónimo: quiere consultar las tarifas y servicios que ofrece la empresa.
- Gerente del contratista: quiere que los usuarios anónimos se sientan atraídos por la oferta de servicios de pago de la empresa y se envíen un formulario de solicitud de información personalizada.
- Ingeniero de software: quiere que los usuarios no tengan dificultades técnicas para acceder a la información.

Precondiciones: el usuario ha accedido al portal web.
Garantías de éxito (Postcondiciones): el gerente recibe notificación emitida por el usuario anónimo.
Escenario principal de éxito (o Flujo Básico):
1- El USUARIO ANÓNIMO accede a la sección de tarifas.
2- El USUARIO ANÓNIMO introduce en el formulario el nombre y texto de la consulta.
3- El USUARIO ANÓNIMO introduce en el formulario su correo electrónico.
4- El SISTEMA confirma que ha rellenado los campos obligatorios y que el correo tiene un formato correcto.

5- El SISTEMA envía notificación al GERENTE para que evalúe la consulta y se ponga en contacto con el posible cliente si lo considera oportuno.

Extensiones (o Flujos Alternativos):

4.a– Falta información obligatoria o el correo no tiene formato correcto

1- El SISTEMA notifica al USUARIO ANÓNIMO de la incidencia encontrada.

2- El SISTEMA solicita de nuevo los datos sin borrar los correctos.

5.a- El SISTEMA no envía correctamente la notificación.

1- El SISTEMA notifica al USUARIO de la incidencia y solicita que se envíe de nuevo.

1- El SISTEMA notifica al INGENIERO DE SOFTWARE de la incidencia.

Requisitos especiales: Se requiere que envío de información sea seguro ante posible ataques garantizando la integridad y confidencialidad del mensaje enviado.

Lista de tecnología y variaciones de datos: Bases de datos relacionales basadas en la nube.

Frecuencia: los accesos a la sección pueden ser continuos.

Temas abiertos: ¿Se va a proporcionar información detalla de precios o es un tema negociable con el cliente?

CU03: Acceder a la zona privada

Actores Principales: Usuario del cliente.

Personal Involucrado e Intereses:

- Usuario del cliente (en adelante USUARIO) o el Director de producción del cliente: quieren acceder a la zona privada.
- Gerente del contratista: quiere que los usuarios accedan fácilmente a la zona privada.

- Ingeniero de software: quiere que los usuarios no tengan dificultades técnicas para acceder a la zona privada.

Precondiciones: el usuario ha accedido al portal web y se ha creado su usuario en el sistema.

Garantías de éxito (Postcondiciones): el usuario accede a la información privada.

Escenario principal de éxito (o Flujo Básico):

1- El USUARIO introduce su nombre sus credenciales (nombre de usuario y contraseña).

2- El SISTEMA comprueba que los campos tienen el formato adecuado en el lado del navegador.

3- El SISTEMA comprueba que las credenciales son correctas en el lado del servidor.

4- El SISTEMA muestra la zona privada al USUARIO.

Extensiones (o Flujos Alternativos):

2.a– Falta información obligatoria o no tiene formato correcto

1- El SISTEMA notifica al USUARIO de la incidencia encontrada.

3.a- El SISTEMA no acepta credenciales correctas.

1- El USUARIO se pone en contacto con la empresa para solucionar la incidencia.

Requisitos especiales: Se requiere que envío de información sea seguro ante posibles ataques garantizando la integridad y confidencialidad de las credenciales.

Lista de tecnología y variaciones de datos: Se requiere una capa de seguridad adicional en el sistema.

Frecuencia: los accesos a la zona privada pueden ser continuos.

Temas abiertos: ¿Cuántas veces se puede intentar el acceso desde una dirección IP o con el mismo nombre usuario?

CU04: Acceder a la información

Actores Principales: Usuario del cliente.

Personal Involucrado e Intereses:

- Usuario del cliente: quiere acceder a la información.
- Gerente del contratista: quiere que los usuarios accedan fácilmente a la información de forma segura.
- Ingeniero de software: quiere que los usuarios no tengan dificultades técnicas para acceder a la información.
- Científico de datos: quiere que la información proporcionada sea correcta.

Precondiciones: el usuario ha accedido a la zona privada.

Garantías de éxito (Postcondiciones): el usuario vuelve a la página inicial del portal.

Escenario principal de éxito (o Flujo Básico):

1- El USUARIO navega en la sección.

Extensiones (o Flujos Alternativos):

1.a– El SISTEMA no muestra la información de forma correcta.

1- El USUARIO se pone en contacto con el administrador del sistema (Ingeniero de software).

Requisitos especiales: Se requiere un entorno modular que permita la modificación de su configuración.

Lista de tecnología y variaciones de datos:

Frecuencia: los accesos a la zona privada pueden ser continuos.

Temas abiertos: ¿Qué nivel de detalle se va a mostrar en el panel de control?

CU05: Recibir soporte técnico

Actor Principal: Usuario del cliente.

Personal Involucrado e Intereses:

- Usuario del cliente: quiere poder solicitar soporte técnico.
- Gerente del contratista: quiere que los usuarios accedan fácilmente al soporte técnico.
- Ingeniero de software: quiere que los usuarios no tengan dificultades técnicas para solicitar ayuda técnica.

Precondiciones: el usuario ha accedido a la zona privada.

Garantías de éxito (Postcondiciones): el usuario sale de la zona privada.

Escenario principal de éxito (o Flujo Básico):

1- El USUARIO pulsa el botón de ayuda.

2- EL USUARIO rellena el formulario describiendo la incidencia.

3- El SISTEMA envía al Ingeniero de Software la notificación de incidencia.

4- El INGENIERO DE SOFTWARE resuelve la incidencia.

5- El SISTEMA notifica al USUARIO el resultado positivo de la intervención.

Extensiones (o Flujos Alternativos):

3.a– El SISTEMA no envía el formulario

1- El USUARIO lo reintenta en otro momento.

4.a– El SISTEMA no envía notificación de resolución de incidencia.

1- El SISTEMA notifica al INGENIERO DE SOFTWARE

Requisitos especiales:

Se requiere 24h para resolución de la incidencia.

Se requiere que se preste servicio en horario laboral

Lista de tecnología y variaciones de datos:

Frecuencia: las peticiones de soporte serán esporádicas.

Temas abiertos: ¿Se va a contratar a una persona para que se dedique en exclusiva a las incidencias?

CU06: Gestionar cuentas de los usuarios

Actor Principal: Ingeniero de Software.

Personal Involucrado e Intereses:

- Ingeniero de Software: quiere gestionar las cuentas de los usuarios
- Gerente del contratista: quiere que el sistema funcione correctamente.

Precondiciones: el Director ha accedido al sistema.

Garantías de éxito (Postcondiciones): los usuarios tienen permisos adecuados para el uso del sistema.

Escenario principal de éxito (o Flujo Básico):

1- El INGENIERO entra en la sección de gestión de usuarios.

2- EL INGENIERO gestiona una cuenta (añade, elimina o modifica permisos).

3- EL INGENIERO confirma las modificaciones.

4- El SISTEMA notificación al INGENIERO que la operación se ha realizado con éxito.

Extensiones (o Flujos Alternativos):

2.a– El SISTEMA no permite gestionar cuentas

1- El INGENIERO DE SOFTWARE revisa el código del sistema.

3.a– El SISTEMA no almacena la modificaciones.

1- El INGENIERO DE SOFTWARE revisa el código del sistema.

4.a- El SISTEMA no envía notificación correctamente.

1- El INGENIERO DE SOFTWARE revisa el código del sistema.

Requisitos especiales:

Se requiere un sistema seguro para este tipo de operaciones.

Lista de tecnología y variaciones de datos:

Frecuencia: las gestiones de usuarios serán esporádicas.

Temas abiertos: ¿Qué nivel de detalle tendrán la concesión de permisos?

CU07: Gestionar incidencias del servicio

Actor Principal: Ingeniero de Software.

Personal Involucrado e Intereses:

- Ingeniero de Software: quiere gestionar las cuentas de los usuarios
- Gerente del contratista: quiere que el sistema funcione correctamente.
- Usuarios: quieren que se resuelvan las incidencias.

Precondiciones: el Ingeniero de Software ha accedido al sistema.

Garantías de éxito (Postcondiciones): el solicitante recibe notificación de incidencia corregida.

Escenario principal de éxito (o Flujo Básico):

1- El INGENIERO recibe notificación del SISTEMA.

2- EL INGENIERO gestiona la incidencia.

3- EL INGENIERO confirma la resolución.

4- El SISTEMA notifica al solicitante que la operación se ha realizado con éxito.

Extensiones (o Flujos Alternativos):

4.a- El SISTEMA no envía notificación correctamente.

1- El INGENIERO DE SOFTWARE revisa el código del sistema.

Requisitos especiales: Se requiere un sistema seguro para este tipo de operaciones.

Lista de tecnología y variaciones de datos:

Frecuencia: las gestiones de incidencias serán continuas.

Temas abiertos: ¿Habrá personal de apoyo dedicado a esta tarea?

CU08: Gestionar el sistema inteligente

Actor Principal: Científico de datos.

Personal Involucrado e Intereses:

- Científico de datos: quiere gestionar el sistema inteligente.

- Gerente del contratista: quiere que el científico de datos tenga las herramientas necesarias.
- Ingeniero de software: quiere que el científico de datos no tenga dificultades técnicas para el uso de las herramientas.

Precondiciones: el científico de datos ha accedido a la zona privada.

Garantías de éxito (Postcondiciones): el científico de datos vuelve a la página inicial del portal.

Escenario principal de éxito (o Flujo Básico):

1- El CIENTÍFICO DE DATOS entra en la sección de configuración del sistema inteligente.

2- El CIENTÍFICO DE DATOS modifica la configuración del sistema inteligente.

3- El CIENTÍFICO DE DATOS confirma la modificación del sistema inteligente.

4- El SISTEMA notifica al CIENTÍFICO DE DATOS que se han procesado los cambios

Extensiones (o Flujos Alternativos):

2.a– El SISTEMA no permite realizar modificaciones.

1- El SISTEMA notifica al INGENIERO DE SOFTWARE.

4.a– El SISTEMA no notifica correctamente al CIENTÍFICO DE DATOS.

1- El SISTEMA notifica al INGENIERO DE SOFTWARE.

Requisitos especiales:

Se requiere mejora continua del sistema

Se requiere un entorno modular que permita la modificación del sistema inteligente.

Se requiere poder gestionar los datos de los que se nutre el sistema inteligente.

Se requiere conocimiento de SQL, Python y librerías Ciencia de Datos.

Lista de tecnología y variaciones de datos:

Herramientas de procesado de datos

Bases de datos relacionales basadas en la nube

Frecuencia: los accesos a la zona privada pueden ser continuos.

Anexo A2. Documento ERS del Sistema Predictivo Precios de Electricidad

Documento ERS del Sistema Predictivo Precios de Electricidad

Lista de cambios	
Versión	Descripción del cambio
1.0	Primera versión del documento. 31/3/2024

Índice

---

1. Introducción

1.1. Propósito

El propósito de este documento es establecer un punto de referencia para las siguientes etapas del ciclo de vida del producto. Se describe el sistema, su funcionalidad, características y restricciones. Está dirigido tanto a la dirección de la empresa para que sirva de base a un futuro contrato con el cliente como a los ingenieros y científicos de datos de la empresa para que se pueda desarrollar de forma completa y eficiente.

1.2. Ámbito del Sistema

El sistema se llamará eNeRGy. Realizará predicciones sobre los precios de la electricidad y mostrará estadísticas relacionadas con este ámbito.
Se espera que generará una ventaja competitiva y un ahorro de costes al cliente al tomar decisiones acertadas y en tiempo real en la gestión de sus recursos. Así mismo la aplicación web servirá de punto atractor de clientes nuevos.

1.3. Definiciones, Acrónimos y Abreviaturas

Definiciones

Python	Lenguaje de programación de alto nivel interpretado multiparadigma muy utilizado en Ciencia de Datos.
SQL	Lenguaje de consulta estructurada, enfocado a la gestión de bases de datos relacionales.
Base de datos relacional	Base de datos que cumple con el modelo relacional formado por tablas que reflejan las relaciones entre los datos.
Paradigma OO	Paradigma de programación que se basa en objetos relacionados entre sí para resolver un problema.
Machine learning	Parte de la Inteligencia Artificial donde los ordenadores aplican técnicas de aprendizaje estadístico con el objetivo de identificar automáticamente patrones en los datos.
BigData	Análisis masivo de datos

Acrónimos

GPU	Graphics Processing Unit	Unidad de procesamiento gráfico
TPU	Tensor Processing Unit	Unidad de procesamiento tensorial
API	Application Programming Interface	Interfaz de programación de aplicaciones

1.4. Referencias

- Especificación de requisitos según el estándar de IEEE 830-1998
- Pressman, R.S. (2021) "Ingeniería del Software. Un enfoque práctico"". 9ª ed, McGraw-Hill.
- Ortigosa R., Vázquez R. (2009). "Manual de Ingeniería del Software". Ed: UDIMA.

1.5. Visión General del Documento

Este documento consta de los siguientes apartados:

- Descripción General
- Requisitos específicos

2. Descripción General

En este apartado se describe el contexto, facilitando la posterior comprensión de los requisitos del sistema.

2.1. Perspectiva del Producto

El producto es independiente de otros sistemas.

2.2. Funciones del Producto

Las principales funciones del sistema son:

- Mostrar estadísticas y gráficos de la evolución de diferentes variables relacionadas con el precio de la electricidad.

- Proporcionar estimaciones fiables de los precios de la energía tanto a corto como medio plazo.

2.3. Características de los Usuarios

Existen distintas categorías de usuarios que van a utilizar el sistema:

Usuario anónimo: no necesita formación ni experiencia técnica en ningún campo.

Por el lado del cliente:

- Usuario: se dedicará al estudio de los datos proporcionados por el sistema.

Por el lado del contratista:

- Ingeniero de software: dedicado al desarrollo y mantenimiento del sistema. Tendrá titulación en Ingeniería Informática con amplia experiencia en el desarrollo de sistema informáticos.
- El científico de datos: encargado de la creación y mejora del sistema inteligente predictivo. Tendrá formación especializada y experiencia en el análisis y gestión de datos, así como, en la construcción de sistemas inteligentes. Además, deberá tener conocimientos profundos en el ámbito de la energía eléctrica.

2.4. Restricciones

Siguiendo la política de la empresa, el sistema se desarrollará con el método en cascada y siguiendo una estructura modular orientada a objetos.

Se usarán servicios Cloud de contenedores para el almacenamiento y despliegue de la aplicación.

El sistema será programado con el lenguaje Python. Además, se usará SQL para gestionar a la base de datos y lenguaje Script para el procesamiento por lotes de diferentes módulos.

Es sistema se creará para ofrecer el servicio a través de la web y será desarrollado en un computador con el Sistema Operativo Windows 11.

Se implantará medidas tanto activas como pasivas en la seguridad de todos los componentes del sistema a través de un sistema de monitoreo, un firewall y protocolos de seguridad basados en HTTP.

2.5. Suposiciones y Dependencias

No se espera que haya cambios en los requisitos que impliquen un cambio sustancial en el diseño e implantación original del sistema.

2.6. Requisitos futuros

Se va a necesitar la permanente actualización de los modelos predictivos para aprovechar las innovaciones que vayan surgiendo.

3. Requisitos Específicos

En este apartado se relacionan los requisitos con el detalle suficiente que permitirá al ingeniero de software y al científico de datos diseñar el sistema de forma adecuada.

3.1. Interfaces Externas

FO-03: Interfaz de usurario en español e inglés: el sistema deberá permitir la elección del idioma haciendo visible un icono que muestre el idioma elegido y permita cambiar a otro.
FD-01: Interfaz de usurario en Catalán, Gallego y Euskera: será deseable que se pueda ver la información en cualquiera de estos idiomas.
FD-02: API para el acceso al servicio: será deseable tener disponible una interfaz para acceder a los datos desde otro sistema.
FD-04: Menús contextuales: será deseable ayuda en pantalla accesible desde cada elemento.

3.2. Funciones

En esta sección se ha elegido clasificar los requisitos en función del tipo de usuario, siendo elegido el que está más relacionado o es afectado por cada uno. Con ello se busca facilitar el proceso de validación de los mismo en una etapa posterior haciendo más fácil la identificación de la persona interesada.

Usuario anónimo online

FO-06: Sección de acceso público en el portal
(CU01: Navegar por la zona pública del portal web): la aplicación web mostrará la información básica de forma pública sin necesidad de registro.
CU02: Consultar el precio de otros servicios más avanzados que ofrece la empresa: la aplicación web mostrará la oferta de tarifas para el acceso a servicios más avanzados y de valor.

Usuario que pertenece al cliente

FO-07: Servicio listo para obtener resultados: el sistema deberá proporcionar la información en tiempo real.
CU03: Acceder a la zona privada con el usuario y contraseñas fijadas: el sistema deberá tener una autentificación segura.
CU05: Recibir soporte técnico ante posibles incidencias: se deberá ofrecer asistencia a los usuarios.

Ingeniero de software del contratista

CU03: Acceder a la zona privada con el usuario y contraseñas fijadas: el sistema deberá tener una

autentificación segura.
CU06: Añadir, eliminar, modificar cuentas de usuarios y los permisos que tienen en el sistema: permitirá al administrador que realice las operaciones CRUD (creación, lectura, modificación y eliminación) de todas las cuentas con permisos en el sistema.
CU07: Gestionar incidencias del servicio: el sistema deberá permitir gestionar adecuadamente las incidencias.

Científico de datos del contratista

FO-01: Realizar predicciones en tiempo real: el sistema deberá realizar predicciones cada 24 horas.
CU08: Gestionar el modelo predictor, mejorando y reentrenado el mismo: el sistema deberá permitir la modificación del modelo predictor.

3.3. Requisitos de Rendimiento

NO-17: Capacidad elevada de cómputo para realizar el entrenamiento del modelo predictivo con un uso intensivo de GPUs y TPUs: el sistema deberá contar con un hardware adecuado con gran potencia de cómputo.

3.4. Restricciones de Diseño

NO-04: Sistema modular escalable en Docker
NO-05: Programarse en Python: se deberá usar una versión estable de los lenguajes de programación elegidos.
NO-06: Contratar servicios en la nube relativos al almacenamiento y procesamiento de datos: se deberá contratar servicios en la nube que permitan la escalabilidad del sistema.
NO-07: Almacenamiento y procesado distribuido
NO-08: Seguir la metodología en cascada
NO-10: Bases de datos relacionales basadas en la nube para la etapa de despliegue: se deberá hacer uso de almacenamiento en la nube que permita adaptarse a la capacidad requerida.
NO-11: Conocimiento de SQL, Python y librerías Ciencia de Datos: se deberá acreditar formación y experiencia en estas áreas de conocimiento de los responsables del diseño y desarrollo.
NO-12: Uso de herramientas de procesado de datos: se deberá acreditar formación y experiencia en estas áreas de conocimiento de los responsables del diseño y desarrollo.
NO-16: Capacidad de almacenamiento elevada para gestionar toda la información que nutrirá el modelo predictivo.
ND-01: Utilizar Bases de datos relacionales gratuitas para el desarrollo: será deseable usar software libre relativo a las bases de datos utilizadas.

3.5. Atributos del Sistema

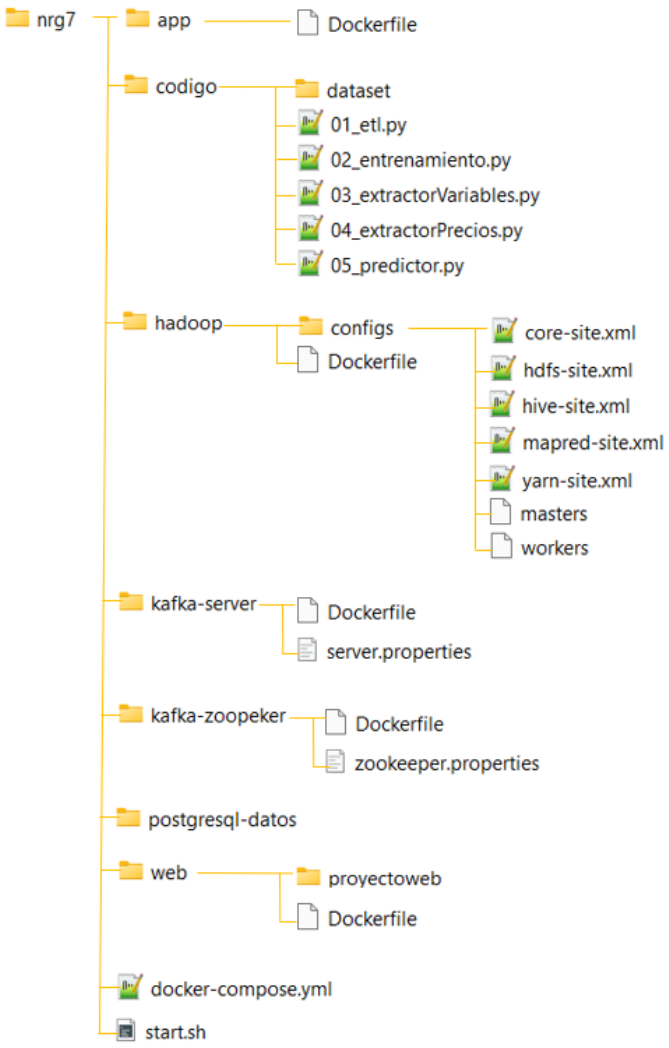
NO-01: 24h para su recuperación ante caídas del sistema: el sistema deberá levantarse en el tiempo prefijado para no perjudicar al cliente.
NO-02: Servicio técnico en horario laboral proporcionado por personal técnico.
NO-09: Énfasis en la seguridad desde el inicio con métodos activos y pasivos: el sistema deberá tener una serie de medidas de seguridad que minimicen los riesgos provocados por ataques informáticos.
NO-13: Mejora continua del sistema con actualizaciones cada 3 meses del modelo de aprendizaje: el sistema deberá ser mejorado de forma regular en el tiempo introduciendo nuevas tecnologías que surjan y hayas sido probadas.

3.6. Otros Requisitos

NO-03: Formación inicial por parte del contratista del sistema completo durante 1 mes: se ofrecerá un curso a los usuarios del cliente que abarque toda la funcionalidad del sistema accesible para el cliente.
NO-14: Compra de datos privados: será necesario comprar datos externos para que el sistema se nutra con información de calidad.
NO-15: Conocimiento técnicas machine learning con experiencia probada de 3 años en puesto similar: se deberá acreditar experiencia en estas áreas de conocimiento de los responsables del diseño y desarrollo.
ND-02: Conocimiento lenguaje script: será deseable conocimiento de este lenguaje para ala automatización de tareas.
EO-01: Tarifa plana con periodo de prueba: se deberá ofrecer una tarifa plana para los servicios ofertados con un periodo de prueba.
EO-02: Plazo máximo de desarrollo 2 meses
EO-03: Cumplir la ley protección de datos: se deberá cumplir en todo momento la legislación vigente en España relativa a la manipulación y almacenamiento de datos personales, en concreto la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.
EO-04: Presupuesto inicial de 7.000 euros
EO-05: Equipo mínimo de 1 persona involucrada directamente en el proyecto
EO-06: Dar formación sobre BigData e Inteligencia artificial: los responsables del diseño y desarrollo del sistema recibirán formación adecuada de estos temas.

Anexo A3. Archivos de configuración y principal código del proyecto

La estructura principal del proyecto es:



En la carpeta principal del proyecto:

docker-compose.yml

```
services:

  postgresql:
    image: postgres
    container_name: postgresql
    ports:
```

```
    - "5432:5432"
  restart: always
  volumes:
     - ./postgresql-datos:/var/lib/postgresql/data
  environment:
    - POSTGRES_PASSWORD=maestropwd
  networks:
    network:
      ipv4_address: 180.20.1.3

postgresqlAdmin:
  image: dpage/pgadmin4
  container_name: postgresqlAdmin
  ports:
    - "82:80"
  restart: always
  environment:
    - PGADMIN_DEFAULT_EMAIL=correo@google.com
    - PGADMIN_DEFAULT_PASSWORD=maestropwd

  networks:
    network:
      ipv4_address: 180.20.1.13

maestro:
  build: ./hadoop
  container_name: maestro
  networks:
    network:
      ipv4_address: 180.20.1.4
  stdin_open: true
  tty: true
  volumes:
     - ./codigo:/codigo
  ports:
    - "55070:50070"
    - "50010:50010"
    - "8088:8088"
    - "8032:8032"
    - "10000:10000"
    - "7077:7077"
    - "9000:9000"
    - "19888:19888"
    - "8080:8080"
    - "18080:18080"
    - "4040:4040"
  extra_hosts:
    - "maestro:180.20.1.4"
    - "esclavo1:180.20.1.5"
    - "esclavo2:180.20.1.6"
    - "esclavo3:180.20.1.7"
    - "esclavo4:180.20.1.8"

esclavo1:
  build: ./hadoop
  container_name: esclavo1
  extra_hosts:
    - "maestro:180.20.1.4"
    - "esclavo1:180.20.1.5"
    - "esclavo2:180.20.1.6"
    - "esclavo3:180.20.1.7"
    - "esclavo4:180.20.1.8"
  volumes:
    - ./codigo:/codigo
  networks:
    network:
      ipv4_address: 180.20.1.5
```

```
    stdin_open: true
    tty: true

  esclavo2:
    build: ./hadoop
    container_name: esclavo2
    extra_hosts:
      - "maestro:180.20.1.4"
      - "esclavo1:180.20.1.5"
      - "esclavo2:180.20.1.6"
      - "esclavo3:180.20.1.7"
      - "esclavo4:180.20.1.8"

    volumes:
      - ./codigo:/codigo
    networks:
      network:
        ipv4_address: 180.20.1.6
    stdin_open: true
    tty: true

  esclavo3:
    build: ./hadoop
    container_name: esclavo3
    extra_hosts:
      - "maestro:180.20.1.4"
      - "esclavo1:180.20.1.5"
      - "esclavo2:180.20.1.6"
      - "esclavo3:180.20.1.7"
      - "esclavo4:180.20.1.8"
    volumes:
      - ./codigo:/codigo
    networks:
      network:
        ipv4_address: 180.20.1.7
    stdin_open: true
    tty: true

  esclavo4:
    build: ./hadoop
    container_name: esclavo4
    extra_hosts:
      - "maestro:180.20.1.4"
      - "esclavo1:180.20.1.5"
      - "esclavo2:180.20.1.6"
      - "esclavo3:180.20.1.7"
      - "esclavo4:180.20.1.8"
    volumes:
      - ./codigo:/codigo
    networks:
      network:
        ipv4_address: 180.20.1.8
    stdin_open: true
    tty: true


  app:
    build: ./app
    container_name: app
    ports:
      - "9005:9002"
    depends_on:
      - maestro
      - esclavo1
    networks:
      network:
        ipv4_address: 180.20.1.10
```

```
    stdin_open: true
    tty: true
    volumes:
       - ./codigo:/codigo

  kafka-zookeper:
    build: ./kafka-zoopeker
    container_name: kafka-zookeper
    ports:
      - "2181:2181"
    restart: always
    networks:
      network:
        ipv4_address: 180.20.1.1
  kafka-server:
    build: ./kafka-server
    container_name: kafka-server
    ports:
      - "9092:9092"
    depends_on:
      - kafka-zookeper
    restart: always
    networks:
      network:
        ipv4_address: 180.20.1.2

  web:
    build: ./web
    container_name: web
    ports:
      - "8000:8000"
    depends_on:
      - postgresql
    networks:
      network:
        ipv4_address: 180.20.1.12
    stdin_open: true
    tty: true
    volumes:
       - ./web/proyectoweb:/proyectoweb


networks:
  network:
    driver: bridge
    ipam:
      config:
        - subnet: 180.20.0.0/16
```

start.sh

```
#!/bin/bash
  echo "----- Formateando hdfs -------"
  docker exec -u hduser -it maestro ./home/hduser/hadoop/bin/hdfs namenode -format
  sleep 5
  echo "----- Arrancando hdfs -------"
  docker exec -u hduser -it maestro start-dfs.sh
  sleep 5
  echo "----- Arrancando yarn -------"
  docker exec -u hduser -d maestro start-yarn.sh
  sleep 5
  echo "----- Arrancando MapReduce-JobHistory Server-------"
  docker exec -u hduser -d maestro mr-jobhistory-daemon.sh start historyserver
```

```
echo "----- Arrancando Spark -------"
docker exec -u hduser -d maestro start-master.sh
docker exec -u hduser -d esclavo1 start-slave.sh maestro:7077
docker exec -u hduser -d esclavo2 start-slave.sh maestro:7077
docker exec -u hduser -d esclavo3 start-slave.sh maestro:7077
docker exec -u hduser -d esclavo4 start-slave.sh maestro:7077
sleep 5
echo "----- Arrancando  History Server -------"
docker exec -u hduser maestro start-history-server.sh
echo "----- Preparando hdfs para hive -----"
docker exec -u hduser -it maestro hdfs dfs -mkdir -p /tmp
docker exec -u hduser -it maestro hdfs dfs -mkdir -p /user/hive/warehouse
docker exec -u hduser -it maestro hdfs dfs -chmod g+w /tmp
docker exec -u hduser -it maestro hdfs dfs -chmod g+w /user/hive/warehouse
sleep 5
echo "----- Arrancando Hive Metastore ----- "
docker exec -u hduser -d maestro hive --service metastore
docker exec -u hduser -d maestro hive --service hiveserver2
echo "Hadoop - cluster     : http://localhost:8088/cluster"
echo "Hadoop - HDFS        : http://localhost:55070/dfshealth.html"
echo "JobHistory Server    : http://localhost:19888"
echo "Spark maestro        : http://localhost:8080"
echo "Spark History Server : http://localhost:18080"
```

En la carpeta “app”:

Dockerfile

```
FROM nrg7-maestro

USER root

RUN apt update
RUN apt install -y  python3-pip

# Instalar apikey de Copernicus
COPY ApiKeys/.cdsapirc /root/
RUN pip install cdsapi

# Librerias necesarias para las aplicaciones
RUN pip install numpy
RUN pip install pyspark
RUN pip install delta-spark==3.1.0
RUN pip install kafka-python
RUN apt-get install -y python3-psycopg2 --fix-missing
RUN pip install pandas
RUN pip install matplotlib

CMD bash
```

En la carpeta “codigo”:

01_etl.py

```
if __name__ == '__main__':
```

```
    print("--------------------------------iniciando-------------------------------------
----------------")

    from pyspark.sql import SparkSession, SQLContext
    from pyspark.sql.functions import col, to_date, max, min, sum, mean
    from delta import *
    import psycopg2
    import json

    builder = SparkSession.builder.master("yarn").appName("app") \
        .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")  \

.config("spark.sql.catalog.spark_catalog","org.apache.spark.sql.delta.catalog.DeltaCatal
og")
    spark = configure_spark_with_delta_pip(builder).getOrCreate()
    sparkContext=spark.sparkContext

    # -----------  Extraer datos  -----------------------

    at  = spark.read.option("inferSchema", "true").option("header", "true")  \
        .csv("file:///codigo/dataset/H_ERA5_ECMW_T639_air_temperature_0002m.csv")
    sl = spark.read.option("inferSchema", "true").option("header", "true")   \
        .csv("file:///codigo/dataset/H_ERA5_ECMW_T639_Sea_Level_Pressure_0000m.csv")
    tp  = spark.read.option("inferSchema", "true").option("header", "true")  \
        .csv("file:///codigo/dataset/H_ERA5_ECMW_T639_Total_Precipitation_0000m.csv")
    ws = spark.read.option("inferSchema", "true").option("header", "true")  \
        .csv("file:///codigo/dataset/H_ERA5_ECMW_T639_WindSpeed_0010m.csv")
    precios = spark.read.option("inferSchema", "true").option("header", "true")  \
        .csv("file:///codigo/dataset/PrecioMercadoSPOTDiario.csv")


    # -----------  Transformar datos  -----------------------
    # --- Agregar datos por horas en dias ------------------------------
    at1  = at.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date"))) \
        .groupBy("fecha").agg(min("ES11").alias("at_min"))
    at2  = at.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date"))) \
        .groupBy("fecha").agg(max("ES11").alias("at_max"))
    at3  = at.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date"))) \
        .groupBy("fecha").agg(mean("ES11").alias("at_med"))
    sl1 = sl.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date")))  \
        .groupBy("fecha").agg(min("ES11").alias("sl_min"))
    sl2 = sl.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date"))) \
        .groupBy("fecha").agg(max("ES11").alias("sl_max"))
    sl3 = sl.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date")))  \
        .groupBy("fecha").agg(mean("ES11").alias("sl_med"))
    tp1 = tp.select('Date','ES11').withColumn("fecha",
to_date(col("Date"))).groupBy("fecha")  \
        .agg(sum("ES11").alias("tp"))
    ws1 = ws.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date")))  \
        .groupBy("fecha").agg(min("ES11").alias("ws_min"))
    ws2 = ws.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date")))  \
        .groupBy("fecha").agg(max("ES11").alias("ws_max"))
    ws3 = ws.select('Date','ES11').filter(col("ES11") > 5).withColumn("fecha",
to_date(col("Date")))  \
        .groupBy("fecha").agg(mean("ES11").alias("ws_med"))


    precios1=precios.withColumn("fecha",
to_date(col("datetime"))).select('fecha','Value').withColumnRenamed('Value','precio')
```

```
# --- Unir datos en una tabla para la base de datos ------------------------------
datos_db1 = at1.join(at2,at1["fecha"] == at2["fecha"]).drop(at2["fecha"])
datos_db2 = datos_db1.join(at3,datos_db1["fecha"] == t3["fecha"]).drop(at3["fecha"])
datos_db3 = datos_db2.join(sl1,datos_db2["fecha"] == l1["fecha"]).drop(sl1["fecha"])
datos_db4 = datos_db3.join(sl2,datos_db3["fecha"] == l2["fecha"]).drop(sl2["fecha"])
datos_db5 = datos_db4.join(sl3,datos_db4["fecha"] == l3["fecha"]).drop(sl3["fecha"])
datos_db6 = datos_db5.join(tp1,datos_db5["fecha"] == p1["fecha"]).drop(tp1["fecha"])
datos_db7 = datos_db6.join(ws1,datos_db6["fecha"] == s1["fecha"]).drop(ws1["fecha"])
datos_db8 = datos_db7.join(ws2,datos_db7["fecha"] == s2["fecha"]).drop(ws2["fecha"])
datos_db9 = datos_db8.join(ws3,datos_db8["fecha"] == s3["fecha"]).drop(ws3["fecha"])



# -----------  Cargar datos en base de datos -----------------------

print("------------ Insertar datos en base de datos    -------------")

# Establecer una conexión a la base de datos PostgreSQL
conn = psycopg2.connect(
            dbname="db_mensajes",
            user="juan",
            password="1234",
            host="postgresql",
            port="5432"
)


# Crear un objeto de cursor usando la conexión.
cur = conn.cursor()

# insertar datos en base de datos

table_name = "panel_tb_datos"
for row in datos_db9.collect():
     cur.execute("INSERT INTO {}
    (fecha,at_min,at_max,at_med,sl_min,sl_max,sl_med,tp,ws_min, \
     ws_max,ws_med) VALUES (%s, %s::real, %s::real, %s::real, %s::real, %s::real,
    %s::real,  \
            %s::real, %s::real, %s::real, %s::real)".format(table_name),
    (row["fecha"], row["at_min"]-273.15,\
            row["at_max"]-273.15, row["at_med"]-273.15, row["sl_min"], row["sl_max"],
    row["sl_med"], row["tp"], \
            row["ws_min"], row["ws_max"], row["ws_med"]))

        table_name = "panel_tb_precios"
for row in precios1.collect():
     cur.execute("INSERT INTO {} (fecha, precio) VALUES (%s, %s)".format(table_name),
    (row["fecha"], row["precio"]))
        conn.commit()

# Cerrar el cursor y la conexión.
cur.close()
conn.close()
print("------------ Datos guardados en base de datos    -------------")

# -----------  Cargar datos en base de hdfs -----------------------

datos_db9.write.format('delta').mode("overwrite").save("hdfs:///datos" )
precios1.write.format('delta').mode("overwrite").save("hdfs:///precios" )

print("---------------- Datos guardados en hdfs  ------------------")

print("---------------- FIN --------------------")
```

02_entrenamiento.py

```
if __name__ == '__main__':

    print("-------------------------------iniciando-----------------------------------
----------------")

    from pyspark.sql import SparkSession, SQLContext
    from pyspark.sql.functions import col, to_date, max, min, mean,year, month,
dayofweek, lag
    from pyspark.sql.window import Window
    from delta import *
    from pyspark.ml.feature import
VectorAssembler,VectorSlicer,StandardScaler,RobustScaler
    from pyspark.ml.evaluation import RegressionEvaluator
    from pyspark.ml.regression import LinearRegression
    from pyspark.ml.regression import GBTRegressor
    from pyspark.ml.regression import RandomForestRegressor
    from pyspark.ml.regression import GeneralizedLinearRegression
    from pyspark.ml.regression import DecisionTreeRegressor
    from pyspark.ml.regression import FMRegressor
    from pyspark.ml import Pipeline
    from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
    import matplotlib.pyplot as plt
    import os

    builder =
SparkSession.builder.master("yarn").appName("app").config("spark.sql.extensions",
"io.delta.sql.DeltaSparkSessionExtension").config("spark.sql.catalog.spark_catalog","org
.apache.spark.sql.delta.catalog.DeltaCatalog")

    spark = configure_spark_with_delta_pip(builder).getOrCreate()
    sparkContext=spark.sparkContext

    print("---------------- cargar datos ------------------")
    datos = spark.read.format("delta").load("hdfs:///datos")
    precios = spark.read.format("delta").load("hdfs:///precios")

    #datos.show()
    #precios.show()

    precios1 = precios.filter(col("fecha") < "2020-01-01")
    #precios1 = precios.filter(col("fecha") < "2024-05-01")

    print("-------- seleccion, limpieza y transformación de los  datos ----------")

    datos1 = precios1.join(datos,datos["fecha"] ==
precios1["fecha"]).drop(datos["fecha"])
    datos2 = datos1.sort(datos1.fecha.asc())
    datos3 = datos2.withColumn("Año", year("fecha")).withColumn("Mes",
month("fecha")).withColumn("Dia", dayofweek("fecha"))

    lag_column = lag("precio", 1).over(Window.orderBy("fecha"))

    datos4 = datos3.withColumn("precio_ant", lag_column).filter(col("fecha") > "2013-12-
31")
    #datos4.show()


    print("---------------- modelado ------------------")

    atributos_a_estandarizar =
["at_min","at_max","at_med","sl_min","sl_max","sl_med","tp","ws_min","ws_max","ws_med"]
    atributos_no_estandarizar = ["Mes","Dia","precio_ant"]

    # obtener datos de entrenamiento y de evaluación
```

```
    train_data, test_data = datos4.randomSplit([0.8, 0.2], seed=1234)

    # crear pipeline
    ensamblador = VectorAssembler( inputCols=atributos_a_estandarizar ,
outputCol="atributos_ensamblados1")

    #escalador = StandardScaler(inputCol="atributos_ensamblados1",
outputCol="atributos_escalados", withMean=True, withStd=True)
    escalador = RobustScaler(inputCol="atributos_ensamblados1",
outputCol="atributos_escalados")

    final_assembler = VectorAssembler(inputCols=atributos_a_estandarizar +
atributos_no_estandarizar, outputCol="atributos_finales")

    algoritmo = 5


    if algoritmo == 1:
        print(" ----------------  LinearRegression   ----------------")
        lr = LinearRegression(featuresCol="atributos_finales", labelCol='precio')
        param_grid = (ParamGridBuilder()
               .addGrid(lr.maxIter, [10, 50, 100])
               .addGrid(lr.regParam, [0.0, 0.01, 0.1])
               .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0])
               .addGrid(lr.solver, ['auto', 'l-bfgs', 'normal'])
               .build())

    elif algoritmo == 2:
        print(" ---------------- GeneralizedLinearRegression   ----------------")
        lr = GeneralizedLinearRegression(featuresCol="atributos_finales",
labelCol='precio',family="gaussian", link="identity")
        param_grid = (ParamGridBuilder()
               .addGrid(lr.maxIter, [10, 50, 100])
               .addGrid(lr.regParam, [0.0, 0.01, 0.1])
               .build())

    elif algoritmo == 3:
        print(" ----------------  GBTRegressor   ----------------")
        lr = GBTRegressor(featuresCol="atributos_finales", labelCol='precio')
        param_grid = (ParamGridBuilder()
               .addGrid(lr.maxDepth, [3, 5, 7])
               .addGrid(lr.maxBins, [32, 64, 128])
               .addGrid(lr.maxIter, [10, 20, 30])
               .addGrid(lr.stepSize, [0.1, 0.05])
               .addGrid(lr.subsamplingRate, [0.8, 1.0])
               .build())
    elif algoritmo == 4:
        print(" ----------------  RandomForestRegressor   ----------------")
        lr = RandomForestRegressor(featuresCol="atributos_finales", labelCol='precio')
        param_grid = (ParamGridBuilder()
               .addGrid(lr.maxDepth, [5, 10, 15])
               .addGrid(lr.numTrees, [10, 20, 30])
               .build())
    else:
        print(" ----------------  DecisionTreeRegressor   ----------------")
        lr = DecisionTreeRegressor(featuresCol="atributos_finales", labelCol='precio')
        param_grid = (ParamGridBuilder()
               .addGrid(lr.maxDepth, [3, 5, 7])
               .addGrid(lr.maxBins, [32, 64, 128])
               .addGrid(lr.minInstancesPerNode, [1, 5, 10])
               .addGrid(lr.minInfoGain, [0.0, 0.1, 0.2])
               .build())


    pipeline = Pipeline(stages=[ensamblador, escalador,final_assembler, lr])
```

```
    # Hacer validación cruzada

    evaluador = RegressionEvaluator(labelCol="precio", predictionCol="prediction",
metricName="rmse")
    crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=param_grid,
                          evaluator=evaluador,
                          numFolds=3)   # 10


    print("---------------- Entrenar datos ------------------")

    # Ajustar el modelo utilizando CrossValidator
    modelo_cv = crossval.fit(train_data)

    # Obtener predicciones del mejor modelo ajustado
    mejor_modelo = modelo_cv.bestModel
    predicciones = mejor_modelo.transform(test_data)
    #predicciones.select(predicciones['*']).show()

    features =
["at_min","at_max","at_med","sl_min","sl_max","sl_med","tp","ws_min","ws_max","ws_med","
Mes","Dia","precio_ant"]
    rmse = evaluador.evaluate(predicciones)

    predicciones.select("prediction", "precio", *features).show(5)
    print("Error cuadrático medio raiz(RMSE) en datos de evaluación = %g euros/MWh" %
rmse)

    # Grabar modelo
    # Si existe borrar =>  hadoop fs -rm -r hdfs:///modelo_aprendizaje

    mejor_modelo.save("hdfs:///modelo_aprendizaje")


    # Obtener los valores de las etiquetas y las predicciones del DataFrame de PySpark
    predicciones_df = predicciones.select("precio", "prediction").toPandas()

    # Extraer las etiquetas y las predicciones
    labels = predicciones_df["precio"]
    predicciones = predicciones_df["prediction"]


    # Crear un gráfico de dispersión
    plt.figure(figsize=(10, 10))
    plt.scatter(labels, predicciones, color='blue', label='Predicciones euros/MWh')

    # Trazar la línea de 45 grados para comparación
    plt.plot(labels, labels, color='green', linestyle='dotted', label='Predicción
perfecta')

    # Etiquetas y título
    plt.xlabel('Precios reales')
    plt.ylabel('Predicciones')
    plt.title('Precios reales vs predicciones')

    # Agregar leyenda
    plt.legend()

    image_path = 'codigo/scatter.png'

    # Verificar si el archivo existe y eliminarlo si es necesario
    if os.path.exists(image_path):
        os.remove(image_path)
        print(f"----------------Eliminado archivo existente: {image_path} -----------")
```

```
    # Guardar la figura como un archivo de imagen
    plt.savefig(image_path, format='png')


    # Mostrar la ruta del archivo de imagen
    print(f"----------------Guardado gráfico en: {image_path} -----------")

    # Mostrar el gráfico
    plt.close()

    print("---------------- FIN ------------------")
```

03_extractorVariables.py

```
from kafka import KafkaProducer
from datetime import datetime
from json import dumps
import time
import pandas as pd
import requests


if __name__ == "__main__":
    print("Inciando extracción de datos ... ")
    print("Esperando 3 segundos.. ")
    time.sleep(3)

    kafka_producer_obj = None
    kafka_producer_obj = KafkaProducer(bootstrap_servers='kafka-
server:9092',value_serializer=lambda x: dumps(x).encode('utf-8'))

    topic = ""

    message = None

    # Configuración petición a AEMET ---------------------------------------------------
----
    url = "https://opendata.aemet.es/opendata/api/observacion/convencional/todas/"

    querystring =
{"api_key":"eyJhbGciOi1NiJ9.eyJzdWIiOiJqdWFuX2Rb0B5YWhvby5lcyIsImp0aSI6IjYzZDgxNzk5LWE2N
jUtNDA5MS1hNGFlLWIxY2Q5NDJhNmlzcyI6IkFFTUVUIiwiaWF0IjTgxNjc1LCJ1ciI2M2Q4MTc5OS1hNjY1LTQw
OTEtYTRhZS1iMWNkOTQyYTZhNDMiLCJyb2xl.Rp3HsjFQ7buQSYC7p_ntd8z2tLdRBubP0lZL2HRxFZo"}

    i = 1

    while True:
        try:

            topic = "Topic_Variables"

            mensaje = {}
            # Datos  ----------------------------------------------------
            mensaje["id"] = i
            mensaje["fecha"] = datetime.now().strftime("%Y-%m-%d")


             # Extracción de datos actuales de ESIOS ---------------------------------
-----------
            headers = {
                'cache-control': "no-cache",
                'fechaIniStr': mensaje["fecha"]+"T00:00:00",
```

```
                'fechaFinStr': mensaje["fecha"]+"T23:59:59"
                }

            peticion_metadatos = requests.request("GET", url, headers=headers,
params=querystring)
            data = peticion_metadatos.json()

            if data['estado'] == 200:

                peticion_datos= requests.get(data['datos'])
                df_variables = pd.DataFrame(peticion_datos.json())

                # Formar mensaje a enviar-------------------------------------------
                mensaje["at_min"] = df_variables['tamin'].mean()
                mensaje["at_max"] = df_variables['tamax'].mean()
                mensaje["at_med"] = df_variables['ta'].mean()
                mensaje["sl_min"] = df_variables['pres_nmar'].min()
                mensaje["sl_max"] = df_variables['pres_nmar'].max()
                mensaje["sl_med"] = df_variables['pres_nmar'].mean()
                mensaje["tp"] = df_variables['prec'].mean()
                mensaje["ws_min"] = df_variables['vv'].min()
                mensaje["ws_max"] = df_variables['vv'].max()
                mensaje["ws_med"] = df_variables['vv'].mean()

                i = i + 1
                envio = kafka_producer_obj.send(topic, mensaje)
                print(mensaje["fecha"]+": " +"Mensaje "+ str(mensaje["id"]) +" enviado
al tema: " + topic)
                print("Temperatura - min: "+ str(mensaje["at_min"])+ ", máx: " +
str(mensaje["at_max"])+ ", media: "+ str(mensaje["at_med"]))
                print("Presión - min: "+ str(mensaje["sl_min"])+ ", máx:
"+str(mensaje["sl_max"])+ ", media: "+ str(mensaje["sl_med"]))
                print("Total precipitaciones: "+ str(mensaje["tp"]))
                print("Velocidad del viento - min: "+ str(mensaje["ws_min"])+ ", máx:
"+str(mensaje["ws_max"])+ ", media: "+ str(mensaje["ws_med"]))
                print("-------------------------------------------")

            else:
                    print("Error: "+str(data['estado']))

        except Exception as ex:
            print("Mensaje no enviado. ")
            print(ex)

        time.sleep(5)

    print("Extracción completada. ")
```

04_extractorPrecios.py

```
from kafka import KafkaProducer
import requests
import pandas as pd
from datetime import datetime, timedelta
from json import dumps
import time
import random
```

```
if __name__ == "__main__":
    print("Inciando extracción de precios ... ")
    print("Esperando 3 segundos.. ")
    time.sleep(3)

    kafka_producer_obj = None
    kafka_producer_obj = KafkaProducer(bootstrap_servers='kafka-
server:9092',value_serializer=lambda x: dumps(x).encode('utf-8'))

    topic = ""
    i = 1

    message = None

   # Configuración petición a ESIOS ----------------------------------------------------
---

    URL_BASE = 'https://api.esios.ree.es/'
    ENDPOINT = 'indicators/'
    INDICATOR = '600'
    url = URL_BASE + ENDPOINT +INDICATOR
    API_TOKEN = 'b48ca50332f75346526a05f23d081f3c5bdfgbbbc73b4622d06178ba3b20de'

    headers = {
        'Host': 'api.esios.ree.es',
        'x-api-key': API_TOKEN
    }



    # Repetir indefinidamente
    while True:
        try:

            ayer = datetime.today() - timedelta(1)
            ayer = ayer.strftime("%Y-%m-%d")

            params = {
                'start_date': ayer + "T00",
                'end_date': ayer +"T23"
            }
            # Extracción de datos de ESIOS ------------------------------------------
----------
            res = requests.get(url, headers=headers, params=params)
            data = res.json()
            df = pd.DataFrame(data['indicator']['values'])
            df = df[['datetime_utc', 'geo_id', 'geo_name', 'value']]
            precio= round(df[df["geo_name"].isin(["España"])]['value'].mean(),2)

            topic = "Topic_Precio"

            mensaje = {}
            # Datos a enviar ------------------------------------------------------
            mensaje["id"] = i
            mensaje["fecha"] = datetime.now().strftime("%Y-%m-%d")
            mensaje["precio"] = precio

            # Envío de datos  -----------------------------------------------------

            i = i + 1
            envio = kafka_producer_obj.send(topic, mensaje)
            print(mensaje["fecha"]+": " +"Mensaje "+ str(mensaje["id"]) +" enviado al
tema: " + topic + " --> "+  str(mensaje["precio"]))


        except Exception as ex:
            print("Mensaje no enviado. ")
```

```
            print(ex)

        time.sleep(10)

    print("Extracción completada. ")
```

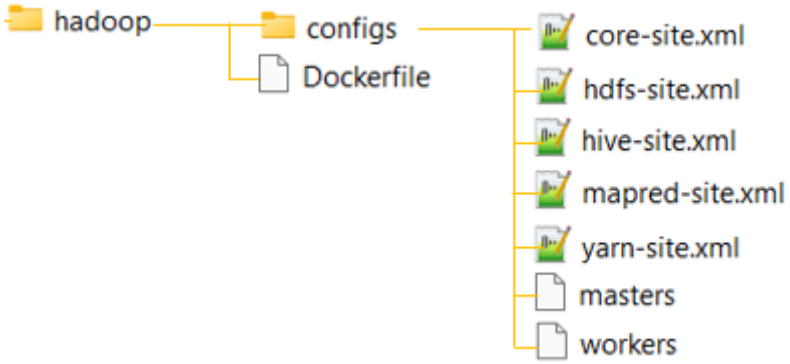
05_predictor.py

```
from kafka import KafkaConsumer
import psycopg2
import json
import time
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.types import StructType, StructField, StringType, IntegerType,
FloatType
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegressionModel
from pyspark.ml import PipelineModel
from delta import *
from datetime import datetime



def create_consumer(topic,group_id):
    bootstrap_servers = ['kafka-server:9092']
    consumer = KafkaConsumer(
        topic,
        bootstrap_servers=bootstrap_servers,
        group_id=group_id,
        auto_offset_reset='latest',
        enable_auto_commit=True,
        max_poll_records = 1,
        value_deserializer=lambda x: x.decode('utf-8')
    )
    return consumer

if __name__ == '__main__':

    builder =
SparkSession.builder.master("yarn").appName("app").config("spark.sql.extensions",
"io.delta.sql.DeltaSparkSessionExtension").config("spark.sql.catalog.spark_catalog","org
.apache.spark.sql.delta.catalog.DeltaCatalog")


    time.sleep(3)
    precio = 0
    at_min = 0
    at_max = 0
    at_med = 0
    sl_min = 0
    sl_max = 0
    sl_med = 0
    tp = 0
    ws_min = 0
    ws_max = 0
    ws_med = 0
    fecha1 = ""
    fecha2 = ""

    topics1 = 'Topic_Precio'
    topics2 = 'Topic_Variables'
```

```
    while True:

            consumer1 = create_consumer(topics1,'group1')
            consumer2 = create_consumer(topics2,'group2')

            msg1 = consumer1.poll(timeout_ms=1000)
            msg2 = consumer2.poll(timeout_ms=1000)

            if msg1:
                for _, messages in msg1.items():
                    for message in messages:
                        print(f'Mensaje recibido de {topics1}: {message.value}')
                        datos = json.loads(message.value)
                        fecha1 = datos['fecha']
                        precio = datos['precio']

                    consumer1.commit()


            if msg2:
                for _, messages in msg2.items():
                    for message in messages:
                        print(f'Mensaje recibido de {topics2}: {message.value}')
                        datos = json.loads(message.value)
                        fecha2 = datos['fecha']
                        at_min = round(float(datos['at_min']),3)
                        at_max = round(float(datos['at_max']),3)
                        at_med = round(float(datos['at_med']),3)
                        sl_min = round(float(datos['sl_min']*100),3)
                        sl_max = round(float(datos['sl_max']*100),3)
                        sl_med = round(float(datos['sl_med']*100),3)
                        tp = round(float(datos['tp']),5)
                        ws_min = round(float(datos['ws_min']),3)
                        ws_max = round(float(datos['ws_max']),3)
                        ws_med = round(float(datos['ws_med']),3)

                    consumer2.commit()
                print(f'{fecha1}, {precio}, {fecha2}, {at_min}, {at_max}, {at_med},
{sl_min}, {sl_max}, {sl_med}, {tp}, {ws_min}, {ws_max}, {ws_med}')

            consumer1.close()
            consumer2.close()

            try:

                fecha3 = datetime.strptime(fecha2, '%Y-%m-%d')
                mes = fecha3.month
                dia = fecha3.weekday()

                # Realizar prediccion
                schema = StructType([
                        StructField("at_min", FloatType(), True),
                        StructField("at_max", FloatType(), True),
                        StructField("at_med", FloatType(), True),
                        StructField("sl_min", FloatType(), True),
                        StructField("sl_max", FloatType(), True),
                        StructField("sl_med", FloatType(), True),
                        StructField("tp", FloatType(), True),
                        StructField("ws_min", FloatType(), True),
                        StructField("ws_max", FloatType(), True),
                        StructField("ws_med", FloatType(), True),
                        StructField("Mes", IntegerType(), True),
                        StructField("Dia", IntegerType(), True),
                        StructField("precio_ant", FloatType(), True)
                    ])
```

```
                datos = [(at_min, at_max, at_med, sl_min,sl_max,sl_med, tp, ws_min,
ws_max, ws_med,mes,dia, precio)]

                spark = configure_spark_with_delta_pip(builder).getOrCreate()
                sparkContext=spark.sparkContext
                lr_model = PipelineModel.load("hdfs:///modelo_aprendizaje")

                df = spark.createDataFrame(datos, schema)

                assembler = VectorAssembler(
inputCols=["at_min","at_max","at_med","sl_min","sl_max","sl_med","tp","ws_min","ws_max",
"ws_med","Mes","Dia","precio_ant"], outputCol="features_vector")

                df = assembler.transform(df)
                predictions = lr_model.transform(df)
                prediccion = predictions.select("prediction").collect()
                print("---- La predicción del precio es: "+
str(prediccion[0].prediction))
                precio_estimado = prediccion[0].prediction
                spark.stop()


                # Establecer una conexión a la base de datos PostgreSQL
                conn = psycopg2.connect(
                    dbname="db_mensajes",
                    user="juan",
                    password="1234",
                    host="postgresql",
                    port="5432"
                )

                table_name = "panel_tb_datos"

                # Crear un objeto cursor usando la conexión.
                cur = conn.cursor()

                # insertar datos en base de datos


                cur.execute("INSERT INTO {}
(fecha,at_min,at_max,at_med,sl_min,sl_max,sl_med,tp,ws_min,ws_max,ws_med,prediccion)
VALUES (%s, %s::real, %s::real, %s::real, %s::real, %s::real, %s::real, %s::real,
%s::real, %s::real, %s::real, %s::real) ON CONFLICT (fecha) DO UPDATE SET at_min =
EXCLUDED.at_min, at_max = EXCLUDED.at_max, at_med = EXCLUDED.at_med, sl_min =
EXCLUDED.sl_min, sl_max = EXCLUDED.sl_max, sl_med = EXCLUDED.sl_med, tp = EXCLUDED.tp,
ws_min = EXCLUDED.ws_min,ws_max = EXCLUDED.ws_max, ws_med = EXCLUDED.ws_med, prediccion
= EXCLUDED.prediccion".format(table_name), (fecha1,
at_min,at_max,at_med,sl_min,sl_max,sl_med,tp,ws_min,ws_max,ws_med, precio_estimado))

                conn.commit()
                # Cerrar el cursor y la conexión.
                cur.close()
                conn.close()

                time.sleep(10)

            except Exception as ex:
                print("Mensaje no enviado. ")
                print(ex)
```

En la carpeta “hadoop” se tienen los siguientes archivos:



Dockerfile

```
FROM ubuntu:latest

# ------------  Instalar hadoop -------------------------------------------

USER root

RUN apt-get update
RUN apt-get install -y openjdk-8-jre openjdk-8-jdk openssh-server iputils-ping

# añadir usuario hduser (hadoop user) para gestionar hadoop,hdfs y yarn
RUN useradd -m -s /bin/bash hduser

# set pubkey authentication
RUN echo "PubkeyAuthentication yes" >> /etc/ssh/ssh_config
RUN mkdir -p /home/hduser/.ssh
RUN echo "PubkeyAcceptedKeyTypes +ssh-dss" >> /home/hduser/.ssh/config
RUN echo "PasswordAuthentication no" >> /home/hduser/.ssh/config


# copiar clave
ADD configs/id_rsa.pub /home/hduser/.ssh/id_rsa.pub
ADD configs/id_rsa /home/hduser/.ssh/id_rsa
RUN chmod 400 /home/hduser/.ssh/id_rsa
RUN chmod 400 /home/hduser/.ssh/id_rsa.pub
RUN cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
RUN chown hduser -R /home/hduser/.ssh

COPY configs/hadoop-3.3.6.tar.gz /home/hduser/
#RUN wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz -P
/home/hduser/
RUN tar -xzvf /home/hduser/hadoop-3.3.6.tar.gz -C /home/hduser/
RUN rm /home/hduser/hadoop-3.3.6.tar.gz
RUN mv /home/hduser/hadoop-3.3.6 /home/hduser/hadoop

# establecer variables de entorno
ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64
ENV HADOOP_HOME /home/hduser/hadoop
ENV HADOOP_CONF_DIR $HADOOP_HOME/etc/hadoop
ENV HADOOP_MAPRED_HOME $HADOOP_HOME
ENV HADOOP_COMMON_HOME $HADOOP_HOME
ENV HADOOP_HDFS_HOME $HADOOP_HOME
ENV HADOOP_COMMON_LIB_NATIVE_DIR $HADOOP_HOME/lib/native
ENV YARN_HOME $HADOOP_HOME
ENV PATH $JAVA HOME/bin:$PATH
ENV PATH $HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

```
RUN echo "export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64" >>
/home/hduser/hadoop/etc/hadoop/hadoop-env.sh
RUN echo "export HADOOP_HOME=/home/hduser/hadoop" >>
/home/hduser/hadoop/etc/hadoop/hadoop-env.sh
RUN echo "export HADOOP_CONF_DIR=/home/hduser/hadoop/etc/hadoop" >>
/home/hduser/hadoop/etc/hadoop/hadoop-env.sh
RUN echo "export YARN_CONF_DIR=/home/hduser/hadoop/etc/hadoop" >>
/home/hduser/hadoop/etc/hadoop/hadoop-env.sh
RUN echo "export HDFS_NAMENODE_USER=hduser" >> /home/hduser/hadoop/etc/hadoop/hadoop-
env.sh
RUN echo "export HDFS_DATANODE_USER=hduser" >> /home/hduser/hadoop/etc/hadoop/hadoop-
env.sh
RUN echo "export HDFS_SECONDARYNAMENODE_USER=hduser" >>
/home/hduser/hadoop/etc/hadoop/hadoop-env.sh

# crear carpetas
RUN mkdir -p /home/hduser/data/nameNode /home/hduser/data/dataNode
/home/hduser/data/nameNodeSecondary /home/hduser/data/tmp
RUN mkdir -p /home/hduser/hadoop/logs
ADD configs/core-site.xml $HADOOP_HOME/etc/hadoop/core-site.xml
ADD configs/hdfs-site.xml $HADOOP_HOME/etc/hadoop/hdfs-site.xml
ADD configs/mapred-site.xml $HADOOP_HOME/etc/hadoop/mapred-site.xml
ADD configs/yarn-site.xml $HADOOP_HOME/etc/hadoop/yarn-site.xml
ADD configs/workers $HADOOP_HOME/etc/hadoop/workers
ADD configs/masters $HADOOP_HOME/etc/hadoop/masters



# permisos
RUN chown hduser -R /home/hduser/data
RUN chown hduser -R /home/hduser/hadoop

# ------------  Instalar Spark  -----------------------------------------


# coger fuentes

COPY configs/spark-3.5.1-bin-hadoop3.tgz /home/hduser/
#RUN wget https://archive.apache.org/dist/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.tgz
-P /home/hduser/
RUN tar -xzvf /home/hduser/spark-3.5.1-bin-hadoop3.tgz -C /home/hduser/
RUN rm /home/hduser/spark-3.5.1-bin-hadoop3.tgz
RUN mv /home/hduser/spark-3.5.1-bin-hadoop3 /home/hduser/spark

RUN mkdir /home/hduser/spark/logs
RUN chown hduser -R /home/hduser/spark/logs

# establecer variables de entorno

ENV SPARK_HOME /home/hduser/spark
ENV SPARK_LOG_DIR /home/hduser/spark/logs
ENV SPARK_WORKER_DIR /home/hduser/spark/work
RUN export SPARK_DIST_CLASSPATH=$(hadoop classpath)
ENV PATH $SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
RUN mv /home/hduser/spark/conf/spark-env.sh.template /home/hduser/spark/conf/spark-
env.sh
RUN echo "export SPARK_DIST_CLASSPATH=$(hadoop classpath)" >>
/home/hduser/spark/conf/spark-env.sh
RUN echo "export SPARK_LOG_DIR=/home/hduser/spark/logs" >>
/home/hduser/spark/conf/spark-env.sh
RUN echo "export HADOOP_CONF_DIR=/home/hduser/hadoop/etc/hadoop" >>
/home/hduser/spark/conf/spark-env.sh
RUN echo "export YARN_CONF_DIR=/home/hduser/hadoop/etc/hadoop" >>
/home/hduser/spark/conf/spark-env.sh
RUN echo "export SPARK_WORKER_DIR=/home/hduser/spark/work" >>
/home/hduser/spark/conf/spark-env.sh
```

```
RUN mv /home/hduser/spark/conf/spark-defaults.conf.template
/home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.eventLog.dir file:/home/hduser/spark/logs" >>
/home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.history.provider org.apache.spark.deploy.history.FsHistoryProvider" >>
/home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.history.fs.update.interval  10s" >> /home/hduser/spark/conf/spark-
defaults.conf
RUN echo "spark.history.ui.port 18080" >> /home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.eventLog.enabled true" >> /home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.history.fs.logDirectory file:/home/hduser/spark/logs" >>
/home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.master yarn" >> /home/hduser/spark/conf/spark-defaults.conf
RUN echo "spark.io.compression.codec snappy" >> /home/hduser/spark/conf/spark-
defaults.conf
ADD configs/workers /home/hduser/spark/conf/workers


RUN chown hduser -R /home/hduser/spark

# ------------  Instalar Delta lake  ------------------------------------------

RUN mkdir /home/hduser/delta
COPY configs/delta-core_2.13-2.4.0.jar /home/hduser/delta
COPY configs/delta-hive-assembly_2.12-3.1.0.jar /home/hduser/delta
#RUN wget https://repo1.maven.org/maven2/io/delta/delta-core_2.13/2.4.0/delta-core_2.13-
2.4.0.jar -P /home/hduser/

ENV DELTA_HOME /home/hduser/delta
ENV PATH $DELTA_HOME:$PATH

EXPOSE 9000 50010 50020 50070 50075 50090 4040
EXPOSE 9871 9870 9820 9869 9868 9867 9866 9865 9864
EXPOSE 8030 8031 8032 8033 8040 8042 8088 8188


CMD service ssh start && bash
```

core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hduser/data/tmp</value>
    </property>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://maestro:9000</value>
    </property>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://maestro:9000</value>
    </property>
       <property>
           <name>hadoop.proxyuser.username.groups</name>
        <value>*</value>
       </property>
       <property>
              <name>hadoop.proxyuser.username.hosts</name>
              <value>*</value>
       </property>
```

```
            <property>
           <name>hadoop.proxyuser.hue.groups</name>
         <value>*</value>
        </property>
        <property>
                <name>hadoop.proxyuser.hue.hosts</name>
                <value>*</value>
        </property>
</configuration>
```

hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
                <name>dfs.permissions</name>
                <value>false</value>
       </property>
    <property>
            <name>dfs.namenode.name.dir</name>
            <value>/home/hduser/data/nameNode</value>
    </property>
    <property>
            <name>dfs.datanode.data.dir</name>
            <value>/home/hduser/data/dataNode</value>
    </property>
    <property>
                <name>dfs.datanode.max.transfer.threads</name>
                <value>8192</value>
       </property>
    <property>
          <name>dfs.namenode.checkpoint.dir</name>
          <value>/home/hduser/data/nameNodeSecondary</value>
    </property>
       <property>
                <name>dfs.namenode.http-address</name>
                <value>maestro:50070</value>
       </property>
    <property>
                <name>dfs.namenode.secondary.http-address</name>
                <value>maestro:50090</value>
       </property>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
    </property>
       <property>
                <name>dfs.webhdfs.enabled</name>
                <value>true</value>
       </property>
       <property>
                <name>dfs.client.use.datanode.hostname</name>
                <value>true</value>
       </property>
       <property>
                <name>dfs.datanode.use.datanode.hostname</name>
                <value>true</value>
       </property>
</configuration>
```

mapred-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.job.tracker</name>
        <value>maestro:9001</value>
    </property>
    <property>
            <name>mapreduce.jobhistory.address</name>
            <value>maestro:10020</value>
       </property>
       <property>
            <name>mapreduce.jobhistory.webapp.address</name>
            <value>maestro:19888</value>
       </property>
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>1536</value>
    </property>
    <property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
        <value>1536</value>
    </property>
    <property>
        <name>yarn.scheduler.minimum-allocation-mb</name>
        <value>128</value>
    </property>
       <property>
        <name>yarn.app.mapreduce.am.resource.memory-mb</name>
        <value>128</value>
    </property>
       <property>
        <name>mapreduce.map.resource.memory-mb</name>
        <value>128</value>
    </property>
       <property>
        <name>mapreduce.reduce.resource.memory-mb</name>
        <value>128</value>
    </property>
    <property>
        <name>yarn.nodemanager.vmem-check-enabled</name>
        <value>false</value>
    </property>
</configuration>
```

yarn-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
    <property>
        <name>yarn.acl.enable</name>
        <value>0</value>
    </property>
       <property>
            <name>yarn.application.classpath</name>
            <value>/home/hduser/hadoop/etc/hadoop,
/home/hduser/hadoop/share/hadoop/common/*,
```

```
/home/hduser/hadoop/share/hadoop/common/lib/*, /home/hduser/hadoop/share/hadoop/hdfs/*,
/home/hduser/hadoop/share/hadoop/hdfs/lib/*,
/home/hduser/hadoop/share/hadoop/mapreduce/*,
/home/hduser/hadoop/share/hadoop/mapreduce/lib/*,
/home/hduser/hadoop/share/hadoop/yarn/*,
/home/hduser/hadoop/share/hadoop/yarn/lib/*</value>
        </property>
    <property>
            <name>yarn.resourcemanager.hostname</name>
            <value>180.20.1.4</value>
    </property>
       <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
            <name>yarn.nodemanager.aux-services</name>
            <value>mapreduce_shuffle</value>
    </property>
       <property>
              <name>yarn.log-aggregation-enable</name>
              <value>true</value>
       </property>
       <property>
              <name>yarn.log-aggregation.retain-seconds</name>
              <value>86400</value>
       </property>
    <property>
            <name>yarn.nodemanager.resource.memory-mb</name>
            <value>1536</value>
    </property>
    <property>
            <name>yarn.scheduler.maximum-allocation-mb</name>
            <value>1536</value>
    </property>
    <property>
            <name>yarn.scheduler.minimum-allocation-mb</name>
            <value>128</value>
    </property>
    <property>
              <name>yarn.nodemanager.resource.cpu-vcores</name>
              <value>2</value>
       </property>
    <property>
            <name>yarn.nodemanager.vmem-check-enabled</name>
            <value>false</value>
    </property>
    <property>
            <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-
percentage</name>
            <value>99</value>
    </property>
    <property>
            <name>yarn.nodemanager.pmem-check-enabled</name>
            <value>false</value>
    </property>


</configuration>
```

masters

180.20.1.4

workers

```
180.20.1.5
180.20.1.6
180.20.1.7
180.20.1.8
```

En la carpeta “kafka-server”

Dockerfile

```
# Utilizar Ubuntu como base
FROM ubuntu:latest

# estbalecer a root como usuario
USER root

# abrir puerto
EXPOSE 9092

# Instalar dependencias
RUN apt-get update
RUN apt install -y wget openjdk-8-jdk

# coger recursos
RUN wget https://downloads.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz
RUN tar -zxvf kafka_2.13-3.7.0.tgz
RUN rm kafka_2.13-3.7.0.tgz
RUN mv /kafka_2.13-3.7.0 /kafka

RUN mkdir /kafka-logs
ADD server.properties /kafka/config/

#  Ejecutar servidor kafka
CMD /kafka/bin/kafka-server-start.sh /kafka/config/server.properties
```

server.properties

```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#
# This configuration file is intended for use in ZK-based mode, where Apache ZooKeeper
is required.
# See kafka.server.KafkaConfig for additional details and defaults
#
```

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will be
equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port
9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
#advertised.listeners=PLAINTEXT://your.host.name:9092

# Maps listener names to security protocols, the default is for them to be the same. See
the config documentation for more details
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEX
T,SASL_SSL:SASL_SSL

# The number of threads that the server uses for receiving requests from the network and
sending responses to the network
num.network.threads=3

# The number of threads that the server uses for processing requests, which may include
disk I/O
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against
OOM)
socket.request.max.bytes=104857600


############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=/kafka-logs

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1

# The number of threads per data directory to be used for log recovery at startup and
flushing at shutdown.
# This value is recommended to be increased for installations with data dirs located in
RAID array.
num.recovery.threads.per.data.dir=1

############################# Internal Topic Settings  #############################
# The replication factor for the group metadata internal topics "__consumer_offsets" and
"__transaction_state"
# For anything other than development testing, a value greater than 1 is recommended to
ensure availability such as 3.
```

```
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1

############################# Log Flush Policy #############################

# Messages are immediately written to the filesystem but by default we only fsync() to
sync
# the OS cache lazily. The following configurations control the flush of data to disk.
# There are a few important trade-offs here:
#    1. Durability: Unflushed data may be lost if you are not using replication.
#    2. Latency: Very large flush intervals may lead to latency spikes when the flush
does occur as there will be a lot of data to flush.
#    3. Throughput: The flush is generally the most expensive operation, and a small
flush interval may lead to excessive seeks.
# The settings below allow one to configure the flush policy to flush data after a
period of time or
# every N messages (or both). This can be done globally and overridden on a per-topic
basis.

# The number of messages to accept before forcing a flush of data to disk
#log.flush.interval.messages=10000

# The maximum amount of time a message can sit in a log before we force a flush
#log.flush.interval.ms=1000

############################# Log Retention Policy #############################

# The following configurations control the disposal of log segments. The policy can
# be set to delete segments after a period of time, or after a given size has
accumulated.
# A segment will be deleted whenever *either* of these criteria are met. Deletion always
happens
# from the end of the log.

# The minimum age of a log file to be eligible for deletion due to age
log.retention.hours=168

# A size-based retention policy for logs. Segments are pruned from the log unless the
remaining
# segments drop below log.retention.bytes. Functions independently of
log.retention.hours.
#log.retention.bytes=1073741824

# The maximum size of a log segment file. When this size is reached a new log segment
will be created.
#log.segment.bytes=1073741824

# The interval at which log segments are checked to see if they can be deleted according
# to the retention policies
log.retention.check.interval.ms=300000

############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=180.20.1.1:2181

# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=18000


############################# Group Coordinator Settings #############################
```

```
# The following configuration specifies the time, in milliseconds, that the
GroupCoordinator will delay the initial consumer rebalance.
# The rebalance will be further delayed by the value of group.initial.rebalance.delay.ms
as new members join the group, up to a maximum of max.poll.interval.ms.
# The default value for this is 3 seconds.
# We override this to 0 here as it makes for a better out-of-the-box experience for
development and testing.
# However, in production environments the default value of 3 seconds is more suitable as
this will help to avoid unnecessary, and potentially expensive, rebalances during
application startup.
group.initial.rebalance.delay.ms=0
```

En la carpeta “kafka-zoopeker”

Dockerfile

```
FROM ubuntu:latest

USER root

# abrir puerto
EXPOSE 2181

RUN apt-get update
RUN apt install -y wget openjdk-8-jdk


# coger recursos
RUN wget https://downloads.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz
RUN tar -zxvf kafka_2.13-3.7.0.tgz
RUN rm kafka_2.13-3.7.0.tgz
RUN mv /kafka_2.13-3.7.0 /kafka

RUN mkdir /zoopeeker-data
ADD zookeeper.properties /kafka/config/


CMD /kafka/bin/zookeeper-server-start.sh /kafka/config/zookeeper.properties
```

zookeeper.properties

```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# the directory where the snapshot is stored.
dataDir=/zookeeper-data
# the port at which the clients will connect
clientPort=2181
```

```
# disable the per-ip limit on the number of connections since this is a non-production
config
maxClientCnxns=0
# Disable the adminserver by default to avoid port conflicts.
# Set the port to something non-conflicting if choosing to enable this
admin.enableServer=false
# admin.serverPort=8080
```

En la carpeta web

Están, entre otros, estos ficheros más específicos para este proyecto:

Dockerfile

```
FROM ubuntu:20.04

USER root

# abrir puerto
EXPOSE 8000

RUN apt-get update
RUN apt-get install -y python3-pip
RUN apt-get install -y libpq-dev
RUN apt-get install -y python3-psycopg2 --fix-missing
RUN python3 -m pip install django
RUN python3 -m pip install Pillow
RUN python3 -m pip install django-crispy-forms
RUN python3 -m pip install pytz --upgrade
RUN python3 -m pip install tzdata --upgrade
RUN python3 -m pip install crispy-bootstrap4
```

settings.py

```
"""
Django settings for proyectoweb project.

Generated by 'django-admin startproject' using Django 4.2.3.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path
import os
from django.contrib import messages as mensajes_error

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = 'django-insecure-grqiysv)fgm19@42wd7)rgll^i38#4!_uzb=!+i2g8goz&o)29'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'web',
    'servicios',
    'avisos',
    'blog',
    'contacto',
    'panel',
    'autenticacion',
    'crispy_forms',
    'crispy_bootstrap4',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'proyectoweb.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'proyectoweb.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

# DATABASES = {
#     "default": {
#         "ENGINE": "django.db.backends.sqlite3",
#         "NAME": "mydatabase",
#     }
```

```
# }

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'db_mensajes',
        'USER': 'juan',
        'PASSWORD': '1234',
        "HOST": "postgresql",
        "PORT": 5432,

    }
}


# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'es-eu'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/


STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]

MEDIA_URL='/media/'
MEDIA_ROOT=os.path.join(BASE_DIR,'media')

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


CRISPY_TEMPLATE_PACK='bootstrap4'
CRISPY_ALLOWED_TEMPLATE_PACK = 'bootstrap4'

MESSAGE_TAGS={
```

```
    mensajes_error.DEBUG: 'debug',
    mensajes_error.INFO: 'info',
    mensajes_error.SUCCESS: 'success',
    mensajes_error.WARNING: 'warning',
    mensajes_error.ERROR: 'danger',

}
```

urls.py

```
"""
URL configuration for proyectoweb project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('web.urls')),
    path('servicios/', include('servicios.urls')),
    path('avisos/', include('avisos.urls')),
    path('blog/', include('blog.urls')),
    path('contacto/', include('contacto.urls')),
    path('panel/', include('panel.urls')),
    path('registro/', include('autenticacion.urls')),
]
```

base.html

```
<html>

<head>

  {% load static %}

    <!-- Bootstrap -->
    <link href="{% static 'vendor/bootstrap/css/bootstrap.min.css'%}" rel="stylesheet">

     <!-- Fonts -->

     <link href="https://fonts.googleapis.com/css2?family=Raleway:wght@300&display=swap"
rel="stylesheet">
     <link href="https://fonts.googleapis.com/css?family=Lora:400,400i,700,700i"
rel="stylesheet">


    <!-- Styles -->
    <link href="{% static 'css/gestion.css'%}" rel="stylesheet">
```

```
</head>

<body>


 <!-- Navbar -->
 <nav class="navbar navbar-expand-lg navbar-dark py-lg-1" id="mainNav">
    <div class="container-fluid justify-content-start">
      <div class="navbar-brand mr-auto" href="#"><img src="../static/img/logo.jpg"
style="width: 150px; height: 70px;" class="rounded"> </div>
      <div  id="navbarResponsive">
        <ul class="nav navbar-nav ">

          <li class="nav-item {% if request.path == '/' %}active{% endif %} px-lg-4">

            <a class="nav-link  text-expanded" href="{% url 'Home' %}">Inicio</a>
          </li>
          <li class="nav-item {% if request.path == '/servicios/' %}active{% endif %}
px-lg-2">
            <a class="nav-link  text-expanded" href="{% url 'Servicios'
%}">Servicios</a>
          </li>
          <li class="nav-item {% if request.path == '/blog/' %}active{% endif %} px-lg-
2">
            <a class="nav-link text-expanded" href="{% url 'Blog' %}">Noticias</a>
          </li>
          <li class="nav-item {% if request.path == '/contacto/' %}active{% endif %} px-
lg-2">
            <a class="nav-link text-expanded" href="{% url 'Contacto' %}">Contacto</a>
          </li>
          {% if user.is_authenticated %}
          <li class="nav-item {% if request.path == '/panel/' %}active{% endif %} px-lg-
2">
            <a class="nav-link  text-expanded mr-4" href="{% url 'Panel' %}">Panel</a>
          </li>
          {% endif %}
          <li class="nav-item px-lg-4 text-center"><div class="d-flex align-items-center
ml-4">{% if user.is_authenticated %}<span class="ml-auto " style="color:aliceblue;
text-align:right"><strong>{{user.username}}</strong></span><a class="nav-link  text-
expanded ml-2" href="{% url 'cerrar_sesion' %}" >Cerrar sesión</a> {% else %}
            <a class="nav-link  text-expanded" href="{% url 'logear' %}">Acceso
clientes</a>    <a class="nav-link  text-expanded" href="{% url 'Registro'
%}">Regístrate</a>
            {% endif %}</div></li>
            </li>

        </ul>

      </div>
    </div>
  </nav>

    <!-- Contenido personalizado -->
     {% block content %}



     {% endblock %}

    <!-- Footer -->
    <footer class="footer text-faded text-center ">
      <div class="container">
        <p class="m-2 ">
              <a href="{% url 'Avisos' %}" class="link">Política de privacidad</a> ·
              <a href="{% url 'Avisos' %}" class="link">Aviso legal</a> ·
```

```
                <a href="{% url 'Avisos' %}" class="link">Cookies</a>
            <span class="m-0 mbt1">&copy; Juan del Rio 2024</span>
                </p>

        </div>
          </footer>

    <!-- Bootstrap -->
    <script src="{% static 'vendor/jquery/jquery.min.js'%}"></script>
    <script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js'%}"></script>

</body>


</html>
```

models.py

```
from django.db import models

# Create your models here.
class tb_datos(models.Model):
    fecha = models.DateField(primary_key=True)
    at_min = models.FloatField()
    at_max = models.FloatField()
    at_med = models.FloatField()
    sl_min = models.FloatField()
    sl_max = models.FloatField()
    sl_med = models.FloatField()
    tp = models.FloatField()
    ws_min = models.FloatField()
    ws_max = models.FloatField()
    ws_med = models.FloatField()
    prediccion = models.FloatField()

class tb_precios(models.Model):
    fecha = models.DateField(primary_key=True)
    precio = models.FloatField()
```

views.py

```
from django.shortcuts import render
from django.http import JsonResponse
from panel.models import tb_datos
from panel.models import tb_precios
from django.core import serializers
from django.contrib.auth.decorators import login_required
import json

# Create your views here.

@login_required(login_url="/registro/logear")
def panel(request):

    # Tabla datos
    datos = tb_datos.objects.filter(fecha__year__gte=1978).order_by('fecha')
    fechas = [dato.fecha.strftime('%Y-%m-%d') for dato in datos]
    at_min = [dato.at_min for dato in datos]
    at_max = [dato.at_max for dato in datos]
    at_med = [dato.at_med for dato in datos]
    sl_min = [dato.sl_min for dato in datos]
    sl_max = [dato.sl_max for dato in datos]
    sl_med = [dato.sl_med for dato in datos]
```

```
    tp = [dato.tp for dato in datos]
    ws_min = [dato.ws_min for dato in datos]
    ws_max = [dato.ws_max for dato in datos]
    ws_med = [dato.ws_med for dato in datos]

    ultimo_registro = datos.last()
    if (ultimo_registro):
        ultimo_valor = round(ultimo_registro.prediccion,2)
    else:
        ultimo_valor = 0.0

    # Tabla precios
    datos_precios = tb_precios.objects.order_by('fecha')
    fechas_precios = [dato.fecha.strftime('%Y-%m-%d') for dato in datos_precios]
    valores_precios = [dato.precio for dato in datos_precios]

    return   render(request,   'panel.html',   {'fechas':   json.dumps(fechas),'at_min':
json.dumps(at_min),'at_max':  json.dumps(at_max),'at_med':  json.dumps(at_med),'sl_min':
json.dumps(sl_min),'sl_max':    json.dumps(sl_max),'sl_med':    json.dumps(sl_med),'tp':
json.dumps(tp),'ws_min':    json.dumps(ws_min),'ws_max':    json.dumps(ws_max),'ws_med':
json.dumps(ws_med),'prediccion':                          ultimo_valor,'fechas_precios':
json.dumps(fechas_precios),'valores_precios': json.dumps(valores_precios)})
```

panel.html

```
{% extends "base.html" %}
{% load static %}
{% block content %}
<head>
    <title>Panel</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        h2 {
            color: white;
            text-align: center;
        }
        form{
            color: rgba(250, 250, 250, 0.614);
            text-align: center;
        }
    canvas {
        max-width: 80%;
        margin: 0 auto;
        display: block;
    }

    .chart-container {
            display: grid;
            grid-template-columns: repeat(2, 1fr);
            grid-gap: 20px;

        }
```

```
</style>
</head>

<body>
    <br>
    <h2 >Estimación del próximo precio de la electricidad: {{ prediccion }}
euros/MWh</h2>
    <br>
    <br>
    <form id="dateRangeForm">
        <label for="startDate">Fecha de inicio:</label>
        <input type="date" id="startDate" name="startDate" value="2024-01-01"
style="background-color: rgb(120, 120, 120);">
        <label for="endDate">Fecha de fin:</label>
        <input type="date" id="endDate" name="endDate" value="2024-04-30"
style="background-color: rgb(120, 120, 120);">
        <button type="submit" style="background-color: rgb(120, 120, 120);">Actualizar
Gráficos</button>
    </form>

    <div><canvas id="chart5" width="800" height="400"></canvas></div>
    <br>
    <div class="chart-container">
    <div><canvas id="chart1" width="800" height="400"></canvas></div>
    <div><canvas id="chart2" width="800" height="400"></canvas></div>
    <div><canvas id="chart3" width="800" height="400"></canvas></div>
    <div><canvas id="chart4" width="800" height="400"></canvas></div>
    </div>
    <script>
        var ctx1 = document.getElementById('chart1').getContext('2d');
        var ctx2 = document.getElementById('chart2').getContext('2d');
        var ctx3 = document.getElementById('chart3').getContext('2d');
        var ctx4 = document.getElementById('chart4').getContext('2d');
        var ctx5 = document.getElementById('chart5').getContext('2d');
        var fechas = JSON.parse('{{ fechas|escapejs }}');
        var ws_med = JSON.parse('{{ ws_med|escapejs }}');
        var ws_max = JSON.parse('{{ ws_max|escapejs }}');
        var at_min = JSON.parse('{{ at_min|escapejs }}');
        var at_max = JSON.parse('{{ at_max|escapejs }}');
        var sl_min = JSON.parse('{{ sl_min|escapejs }}');
        var sl_max = JSON.parse('{{ sl_max|escapejs }}');
        var tp = JSON.parse('{{ tp|escapejs }}');
        var fechas_precios = JSON.parse('{{ fechas_precios|escapejs }}');
        var valores_precios = JSON.parse('{{ valores_precios|escapejs }}');
        var chart1, chart2,chart3,chart4,chart5;
```

```
function updateChart1(startDate, endDate) {
    var filteredFechas = fechas.filter(function(fecha) {
        return fecha >= startDate && fecha <= endDate;
    });
    var startIndex = fechas.indexOf(filteredFechas[0]);
    var endIndex = fechas.indexOf(filteredFechas[filteredFechas.length - 1]);

    var at_min_filtrados = at_min.slice(startIndex, endIndex + 1);
    var at_max_filtrados = at_max.slice(startIndex, endIndex + 1);

    if (chart1) {
        chart1.destroy();
    }

    chart1 = new Chart(ctx1, {
        type: 'line',
        data: {
            labels: filteredFechas,
            datasets: [{
                label: 'Temperatura mínima (°C)',
                data: at_min_filtrados,
                backgroundColor: 'rgba(14, 244, 198, 0.9)',
                borderColor: 'rgba(14, 244, 198, 1)',
                borderWidth: 1
            },
            {
                label: 'Temperatura máxima (°C)',
                data: at_max_filtrados,
                backgroundColor: 'rgba(244, 125, 14, 0.9)',
                borderColor: 'rgba(244, 125, 14, 1)',
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    }
                }]
            }
        }
    });
}
function updateChart2(startDate, endDate) {
        var filteredFechas = fechas.filter(function(fecha) {
            return fecha >= startDate && fecha <= endDate;
        });
        var startIndex = fechas.indexOf(filteredFechas[0]);
```

```
            var endIndex = fechas.indexOf(filteredFechas[filteredFechas.length -
1]);

            var min_filtrados = sl_min.slice(startIndex, endIndex + 1);
            var max_filtrados = sl_max.slice(startIndex, endIndex + 1);

            if (chart2) {
                chart2.destroy();
            }

            chart2 = new Chart(ctx2, {
                type: 'line',
                data: {
                    labels: filteredFechas,
                    datasets: [{
                        label: 'Presión a nivel del mar mínima (Pa)',
                        data: min_filtrados,
                        backgroundColor: 'rgba(99, 99, 99, 0.9)',
                        borderColor: 'rgba(99, 99, 99, 1)',
                        borderWidth: 1
                    },
                    {
                        label: 'Presión a nivel del mar máxima (Pa)',
                        data: max_filtrados,
                        backgroundColor: 'rgba(220, 220, 220, 0.9)',
                        borderColor: 'rgba(220, 220, 220, 1)',
                        borderWidth: 1
                    }]
                },
                options: {
                    scales: {
                        yAxes: [{
                            ticks: {
                                beginAtZero: true
                            }
                        }]
                    }
                }
            });
        }

        function updateChart3(startDate, endDate) {
            var filteredFechas = fechas.filter(function(fecha) {
                return fecha >= startDate && fecha <= endDate;
            });
            var startIndex = fechas.indexOf(filteredFechas[0]);
            var endIndex = fechas.indexOf(filteredFechas[filteredFechas.length -
1]);

            var min_filtrados = ws_med.slice(startIndex, endIndex + 1);
```

```
            var max_filtrados = ws_max.slice(startIndex, endIndex + 1);

            if (chart3) {
                chart3.destroy();
            }

            chart3 = new Chart(ctx3, {
                type: 'line',
                data: {
                    labels: filteredFechas,
                    datasets: [{
                        label: 'Velocidad media del viento (m/s)',
                        data: min_filtrados,
                        backgroundColor: 'rgba(211, 164, 10, 1)',
                        borderColor: 'rgba(211, 164, 10, 1)',
                        borderWidth: 1
                    }]
                },
                options: {
                    scales: {
                        yAxes: [{
                            ticks: {
                                beginAtZero: true
                            }
                        }]
                    }
                }
            });
        }

        function updateChart4(startDate, endDate) {
            var filteredFechas = fechas.filter(function(fecha) {
                return fecha >= startDate && fecha <= endDate;
            });
            var startIndex = fechas.indexOf(filteredFechas[0]);
            var endIndex = fechas.indexOf(filteredFechas[filteredFechas.length -
1]);

            var filtrados = tp.slice(startIndex, endIndex + 1);


            if (chart4) {
                chart4.destroy();
            }

            chart4 = new Chart(ctx4, {
                type: 'line',
                data: {
                    labels: filteredFechas,
                    datasets: [{
```

```
                        label: 'Precipitaciones (m)',
                        data: filtrados,
                        backgroundColor: 'rgba(47, 139, 251, 0.9)',
                        borderColor: 'rgba(47, 139, 251, 1)',
                        borderWidth: 1
                    }]
                },
                options: {
                    scales: {
                        yAxes: [{
                            ticks: {
                                beginAtZero: true
                            }
                        }]
                    }
                }
            });
        }



        function updateChart5(startDate, endDate) {
            var filteredFechas = fechas_precios.filter(function(fecha) {
                return fecha >= startDate && fecha <= endDate;
            });
            var startIndex = fechas_precios.indexOf(filteredFechas[0]);
            var endIndex =
fechas_precios.indexOf(filteredFechas[filteredFechas.length - 1]);

            var filtrados = valores_precios.slice(startIndex, endIndex + 1);


            if (chart5) {
                chart5.destroy();
            }

            chart5 = new Chart(ctx5, {
                type: 'line',
                data: {
                    labels: filteredFechas,
                    datasets: [{
                        label: 'Precios (euros/MWh)',
                        data: filtrados,
                        backgroundColor: 'rgba(91, 244, 25, 0.9)',
                        borderColor: 'rgba(91, 244, 25, 1)',
                        borderWidth: 1
                    }]
                },
                options: {
                    scales: {
```

```
                        yAxes: [{
                            ticks: {
                                beginAtZero: true
                            }
                        }]
                    }
                }
            });
        }


        document.getElementById('dateRangeForm').addEventListener('submit',
function(event) {
            event.preventDefault();
            var startDate = document.getElementById('startDate').value;
            var endDate = document.getElementById('endDate').value;
            updateChart1(startDate, endDate);
            updateChart2(startDate, endDate);
            updateChart3(startDate, endDate);
            updateChart4(startDate, endDate);
            updateChart5(startDate, endDate);
        });

        var startDate = document.getElementById('startDate').value;
        var endDate = document.getElementById('endDate').value;
        updateChart1(startDate, endDate);
        updateChart2(startDate, endDate);
        updateChart3(startDate, endDate);
        updateChart4(startDate, endDate);
        updateChart5(startDate, endDate);


        </script>


        <br>
        <br>
        <br>


{% endblock %}
```