SISTEMA DE MONITORIZACIÓN Y VISUALIZACIÓN DE DATOS METEOROLÓGICOS EN TIEMPO REAL

Memoria del Proyecto de Final de Ciclo

Nombre del alumno/a: Josep Ferrer Ferragud
Centro educativo: IES Jaume II El Just
Ciclo formativo: Curso Especialización Inteligencia Artificial y Big Data
Fecha presentación: 28 de Mayo de 2025

IES Jaume II El Just

Sumario

1. Introducción.

1.1 Objetivos del proyecto.

Este es un proyecto realizado por Josep Ferrer Ferragud para los módulos de Big Data Aplicado y Sistemas de Big Data. El objetivo de este proyecto es poder tener un control meteorológico de las principales ciudades de España, entre ellas Valencia, Madrid y Barcelona.

Para ello he decidido usar las siguientes tecnologías: NodeRed, Elastic, MongoDB, Grafana, Kafka, ZooKeeper y Kibana. Para poder alcanzar los objetivos de este proyecto, he decidido usar una metodología **Scrum** en la que el proyecto se divide en pequeños bloques o Sprints con el objetivo de ir revisando y mejorando la fase anterior.

NodeRed se emplea para consumir datos de la API del clima de forma periódica, procesar los datos iniciales y redirigirlos hacia Kafka o bases de datos como MongoDB y ElasticSearch.

Kafka actúa como intermediario entre la ingesta de datos y su procesamiento/almacenamiento.

Zookeeper se encarga de coordinar los brokers de Kafka, gestionar su configuración y facilitar el equilibrio de carga y la recuperación ante fallos.

MongoDB se utiliza para almacenar datos históricos del clima en formato JSON. Es ideal para consultas flexibles, persistencia a largo plazo y para acceder a los datos de forma estructurada.

ElasticSearch permite realizar búsquedas rápidas y análisis de datos meteorológicos, como encontrar patrones por fecha, ubicación, condiciones climáticas, etc. También sirve como backend de visualización para Kibana.

Kibana se utiliza para crear dashboards interactivos que muestran tendencias, alertas y métricas climatológicas procesadas en tiempo real.

Grafana complementa a Kibana al permitir visualizaciones más personalizadas y orientadas a métricas además de alertas y paneles dinámicos.

Docker es una tecnología indispensable para poder exportar de manera rápida un proyecto y usar menos recursos en el dispositivo.

Se usaron otras herramientas como **Visual Studio Code** para escribir código y **GitHub** para tener un control de versiones.

1.2 Alcance

El proyecto se centra en la creación de una arquitectura funcional para la ingesta, almacenamiento, procesamiento y visualización de datos meteorológicos en tiempo real.

El sistema desarrollado permite:

- Obtener información climática periódica desde una API pública.
- Procesar los datos mediante flujos automáticos.
- Almacenar los datos tanto para consulta estructurada (MongoDB) como para análisis y visualización (Elasticsearch).
- Visualizar las métricas climáticas mediante dashboards en Kibana y Grafana.

El proyecto no abarca la creación de modelos de predicción climática, ni la integración con dispositivos físicos IoT, aunque se considera que podría extenderse en el futuro con dichas funcionalidades.

1.3 Metodología

Para llevar a cabo el proyecto se ha utilizado la metodología ágil **Scrum**, que permite organizar el trabajo en bloques de tiempo llamados **sprints**. Esta forma de trabajar facilita dividir el desarrollo en partes más pequeñas y manejables, y permite hacer mejoras continuas durante el proceso.

Cada sprint ha tenido una duración de una o dos semanas, dependiendo de la carga de trabajo. Al inicio de cada sprint se definieron las tareas principales a realizar, como por ejemplo: la conexión con la API del clima, la configuración de Kafka, el almacenamiento en MongoDB o la creación de dashboards en Grafana. Al finalizar cada sprint, se revisaron los avances y se ajustó la planificación para el siguiente.

Esta metodología ha facilitado una buena organización del trabajo, permitiendo adaptarse rápidamente a posibles problemas y cambios.

2. Descripción general del sistema

2.1 Visión general del flujo de datos

El sistema desarrollado tiene como objetivo principal la recolección, procesamiento, almacenamiento y visualización de datos meteorológicos en tiempo real. Para ello, se ha diseñado una arquitectura distribuida basada en tecnologías de Big Data que permite trabajar con flujos de datos continuos y gestionar información de forma eficiente.

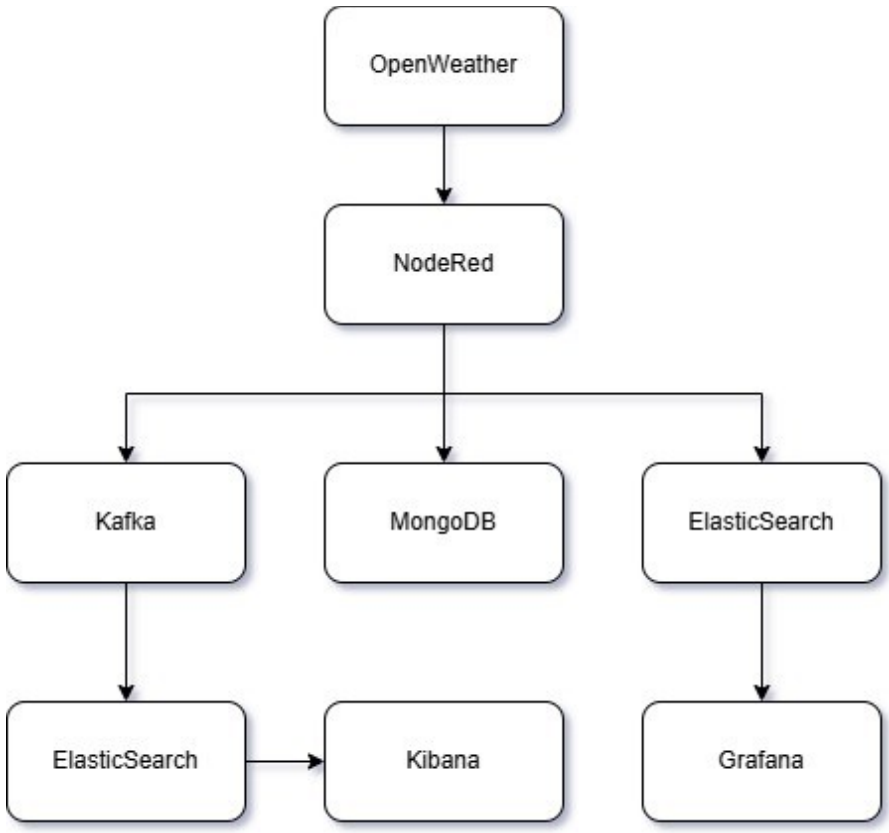
El flujo general funciona de la siguiente manera:

- Node-RED obtiene datos periódicamente desde una API del clima.
- Los datos son enviados a Apache Kafka, que actúa como sistema de mensajería para distribuir la información.
- Desde Kafka, los datos son procesados y almacenados en dos bases de datos:
    1. MongoDB, para almacenamiento flexible y consultas estructuradas.
    2. Elasticsearch, para indexación y búsquedas rápidas.
- Finalmente, los datos se visualizan en tiempo real mediante Grafana y Kibana, mostrando métricas como temperatura, humedad, viento, etc.

Este sistema permite una gestión eficiente de los datos meteorológicos, tanto en tiempo real como en modo histórico.

2.2 Diagrama de componentes

Aquí se muestra una representación del flujo general del sistema:



Este diagrama refleja que **Node-RED actúa como punto central** de entrada de datos, y distribuye la información a distintas partes del sistema. Grafana y Kibana es la interfaz final para la visualización de los datos obtenidos.

3. Tecnologías utilizadas

3.1 Node-RED

Se utiliza para conectarse a la API del clima mediante nodos HTTP request, obtener datos de forma periódica y enviarlos simultáneamente a Kafka, MongoDB y Elasticsearch. Node-RED permite definir esta lógica de forma visual y rápida mediante elementos llamados nodos, lo que facilita las pruebas, la integración y la comprensión del flujo de datos.

3.2 Apache Kafka

Node-RED envía los datos meteorológicos a Kafka, donde se almacenan como eventos. Estos datos son procesados y almacenados en un índice específico que se utiliza para gestionar alertas meteorológicas, como temperaturas anormalmente elevadas.

3.3 MongoDB

Se emplea para guardar un histórico de los datos recibidos de la API del clima. Esto permite realizar consultas por fecha, ciudad u otros parámetros en el futuro.
Además, se organiza la información en diferentes colecciones para mantenerla más estructurada y fácil de consultar.

3.4 Elasticsearch

Node-RED envía los datos a Elasticsearch, donde se indexan automáticamente. Esto permite consultar y visualizar fácilmente las variables meteorológicas.
Además, facilita la integración con Grafana, ya que Elasticsearch es una de las fuentes de datos compatibles.

3.5 Grafana

Grafana se conecta directamente a los índices de Elasticsearch para mostrar en tiempo real gráficos con la evolución de variables como temperatura, humedad, presión atmosférica, velocidad del viento entre otros datos relevantes para el proyecto.

3.6 ApacheZooKeeper

Se utiliza junto con Kafka para mantener la configuración y coordinar los brokers, asegurando la estabilidad del sistema de mensajería.
En la versión de Kafka utilizada en el proyecto, su presencia es necesaria para que funcione correctamente.

3.7 Kibana

Se utiliza como una interfaz gráfica para crear nuevos índices, visualizar datos y poder trabajar con ellos. En el proyecto se utiliza para poder gestionar las alertas por temperaturas elevadas en las diferentes localidades.

4. Desarrollo del sistema

4.1 Obtención de datos meteorológicos

El sistema se inicia con Node-RED, donde se ha configurado un nodo HTTP request que se conecta periódicamente a una API de datos meteorológicos. Este nodo realiza peticiones cada 10 minutos para obtener información actualizada sobre el clima de distintas ciudades.

Los datos recibidos incluyen parámetros como temperatura, humedad, presión atmosférica, velocidad del viento, entre otros. Una vez obtenidos, se procesan mediante nodos function para transformarlos al formato requerido y se preparan para su envío simultáneo a distintas plataformas de almacenamiento y análisis.

4.2 Procesamiento y envío a múltiples destinos

Tras recibir los datos, Node-RED los envía a tres destinos principales:

1. **Kafka**: Los datos se publican en un tópico de Kafka, que actúa como sistema de mensajería distribuido. Esto permite que otros sistemas puedan consumirlos más adelante de forma asíncrona.

2. **MongoDB**: Se almacenan todos los datos como histórico, permitiendo búsquedas por fecha, ciudad y otros filtros. Ideal para análisis detallados o futuras ampliaciones del sistema.

3. **Elasticsearch**: Los datos también se envían a Elasticsearch, donde se indexan automáticamente. Esto permite una búsqueda eficiente y conexión directa con Grafana para la visualización.

Este envío simultáneo permite aprovechar lo mejor de cada tecnología: robustez, persistencia y análisis rápido.

4.3 Visualización en tiempo real con Grafana

Grafana se configura para conectarse directamente a los índices de Elasticsearch. Desde ahí, se crean dashboards que muestran en tiempo real la evolución de las variables meteorológicas.

Cada gráfico representa un parámetro: temperatura, humedad, presión, etc., y se actualizan automáticamente a medida que se reciben nuevos datos. Se han configurado paneles por ciudad, por variable y por rango de tiempo, facilitando el análisis visual de tendencias climáticas.

4.4 Gestión de alertas meteorológicas

Además de la visualización, el sistema permite configurar alertas por condiciones extremas, como temperaturas superiores a ciertos umbrales.

Esto se ha implementado mediante el análisis de los datos que pasan por Kafka: si una temperatura supera el umbral establecido, se genera un evento de alerta y se almacena en un índice separado en Elasticsearch.

Desde Kibana, se pueden revisar estas alertas y visualizar en qué ciudades y momentos se han superado ciertos límites, lo que aporta una capa de prevención o información crítica en tiempo real.

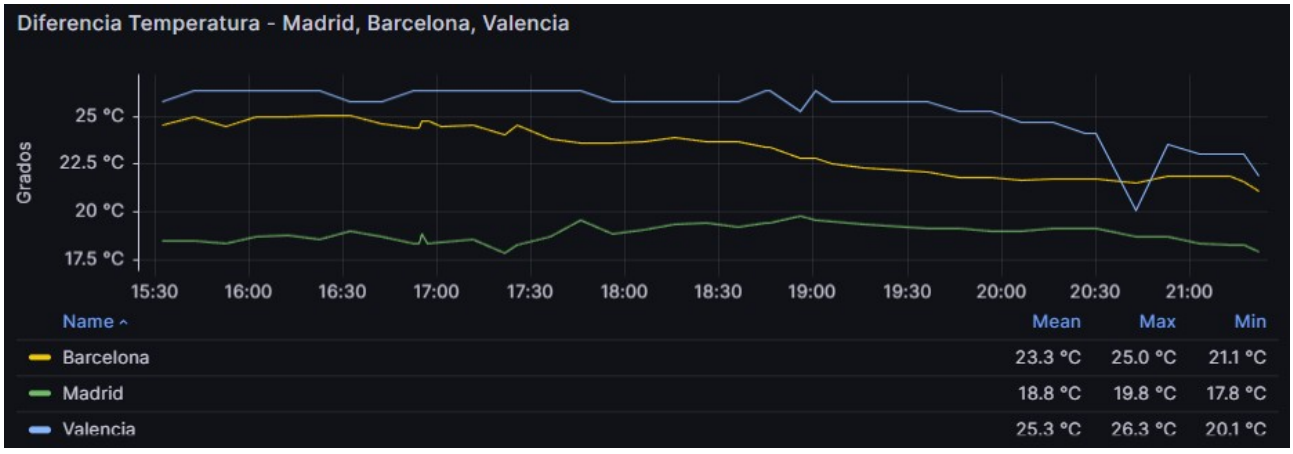
4.5 Pruebas y validación del sistema

Se han realizado pruebas funcionales para verificar que:

- La API responde correctamente y los datos llegan en el formato esperado.
- Node-RED realiza las peticiones en los intervalos establecidos.
- Los datos se almacenan correctamente en MongoDB y Elasticsearch.
- Grafana muestra las gráficas en tiempo real sin errores.
- Las alertas se activan correctamente al superar los valores límite.
-

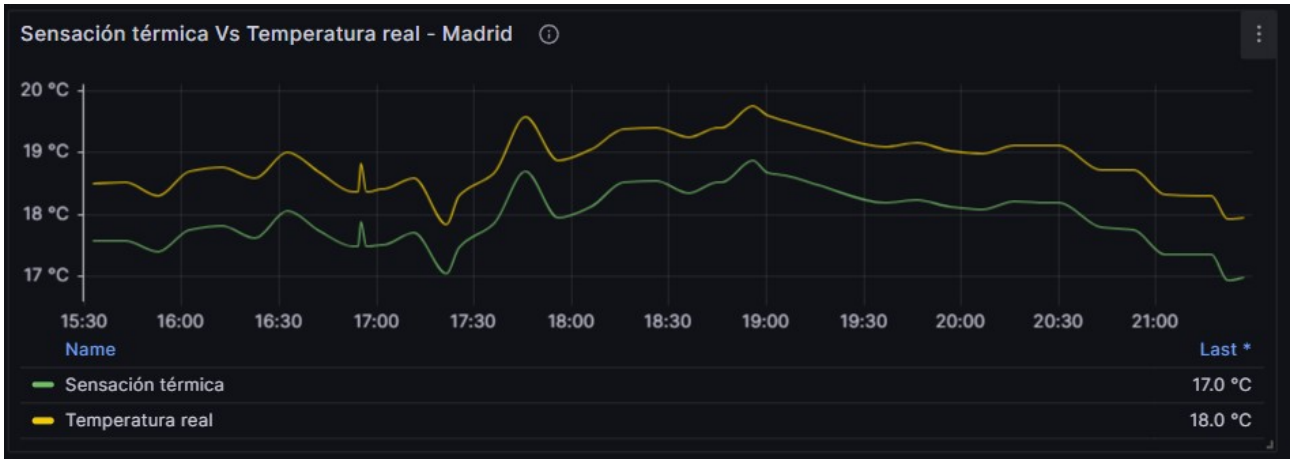
5. Resultados y visualizaciones

5.1 Capturas de dashboards

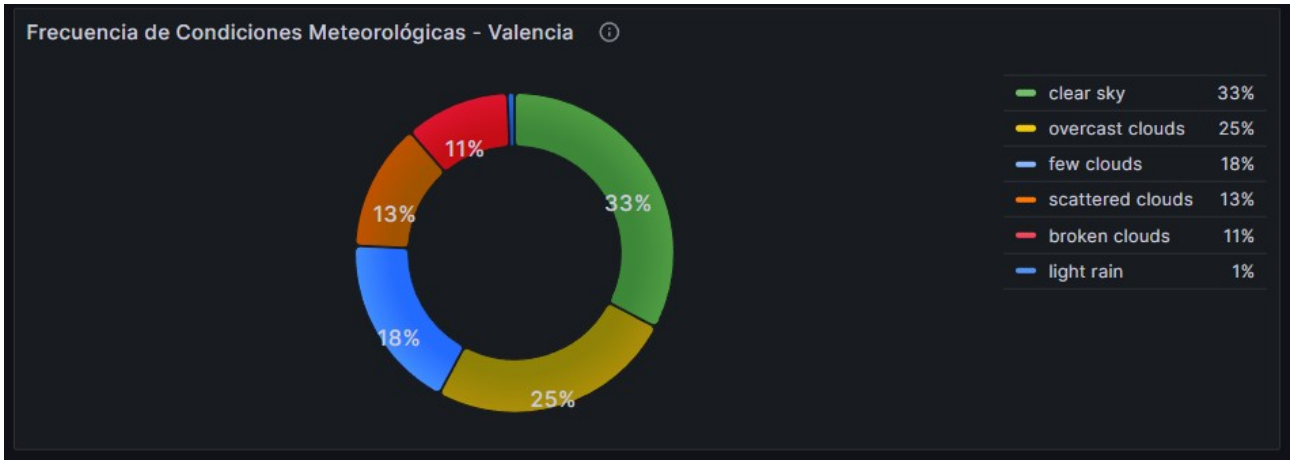


En esta gráfica se puede observar la diferencia entre la temperatura de las principales provincias

de España. Además también se puede ver la temperatura máxima, mínima y la media del tiempo seleccionado. En el caso de esta gráfica, las últimas 6 horas.



Mediante la gráfica anterior, podemos apreciar la diferencia entre la temperatura real y la sensación térmica en Madrid. Gracias a esta podemos decir que, mayoritariamente, la temperatura real en la capital de España es siempre 1 grado mayor a la sensación térmica.



En esta gráfica, se puede ver el porcentaje de cada condición meteorológica en la ciudad de Valencia en los últimos 2 días. Podemos deducir que Valencia es una ciudad en la que predominan los cielos despejados y un tanto nublados aunque en algún momento llueva suavemente.

En la captura anterior se aprecia cómo sería el dashboard con un intervalo de 12 horas.

Como hemos visto, Grafana es una herramienta muy potente y atractiva visualmente que permite al usuario seleccionar un intervalo de tiempo y ver información acerca de este.

En Kibana también es posible generar dashboards. Para tener la funcionalidad separada, he decidido implementar aquí el panel de las alertas. En el siguiente gráfico se puede observar información sobre las alertas de la ciudad de Barcelona.

Se observa un gráfico que muestra el número de alertas en la última semana. Se puede ver también cuando fue la última alerta junto con la temperatura que se detectó. Está configurado para que el umbral sea 23 °C.

5.2 Análisis de datos

Tras un periodo de recolección continua de datos meteorológicos, se han identificado varias tendencias y comportamientos interesantes:

- **Temperatura**: Se ha podido observar que la temperatura desciende drásticamente al caer la noche y su punto álgido es a mediodía.
- **Humedad**: En las zonas costeras, como Valencia o Barcelona, la humedad relativa es significativamente más alta durante la madrugada y primeras horas del día.
- **Presión atmosférica:** En general se mantiene estable, pero se han registrado descensos bruscos antes de cambios climáticos importantes, como tormentas o lluvias.
- **Viento:** Se ha identificado un aumento de velocidad de viento por la tarde en algunas zonas montañosas, lo que podría tener impacto en actividades al aire libre o agrícolas.

Estos patrones han sido fácilmente detectables gracias a los dashboards de Grafana, que permiten filtrar por ciudad y fecha, y comparar variables en tiempo real.

5.3 Casos de uso destacados

Durante el desarrollo del proyecto, se han identificado varios casos prácticos donde el sistema ha demostrado su utilidad:

- **Generación de alertas por altas temperaturas:** Cuando la temperatura en algunas ciudades superó los 25 °C, el sistema generó eventos de alerta automáticamente a través de Kafka y los almacenó en un índice específico en Elasticsearch. Estos eventos se visualizaron en tiempo real con Kibana y se representaron gráficamente en Grafana.
- **Consulta histórica de datos:** Gracias al almacenamiento en MongoDB, se pudieron hacer consultas específicas por fecha y ciudad para comparar las condiciones meteorológicas entre distintos días. Por ejemplo, se consultaron datos de Madrid durante la última semana de abril para estudiar una anomalía térmica registrada.
- **Análisis preventivo:** Se usó el sistema para analizar si había correlación entre presión atmosférica baja y la llegada de tormentas en ciertas zonas. Esto permitió anticipar eventos meteorológicos con mayor precisión y visualizar claramente las caídas de presión antes de los cambios.

Estos casos evidencian la versatilidad del sistema para combinar recolección, análisis y visualización de datos en un entorno de big data distribuido.

6. Implementación del proyecto

El proyecto completo se encuentra disponible en el siguiente repositorio de GitHub: https://github.com/FerrerJosep/CEIABD-ClimasES.git

Para implementar el sistema en un entorno local, se deben seguir los siguientes pasos:

6.1 Clonación del repositorio

Primero, abrimos una terminal en la carpeta donde queremos alojar el proyecto y ejecutamos el siguiente comando para clonar el repositorio:

git clone https://github.com/FerrerJosep/CEIABD-ClimasES.git

6.2 Ejecución de contenedores con Docker

Dentro del directorio clonado, encontraremos el archivo docker-compose.yml. Para iniciar todos los servicios necesarios (Kafka, Zookeeper, Elasticsearch, MongoDB, etc.), ejecutamos:

docker-compose up -d

Esto levantará los contenedores en segundo plano. Una vez activos, podremos acceder a las diferentes herramientas desde el navegador.

6.3 Configuración de Node-RED

Accedemos a Node-RED desde: http://localhost:1880

Importamos el archivo flowsNodeRed.json desde el menú de Node-RED.

Aseguramos que están instalados los nodos necesarios:

- **node-red-contrib-kafka-client**
- **node-red-node-mongodb**

Ejecutamos el nodo inject para comenzar la ingesta periódica de datos meteorológicos desde la API.

6.4 Configuración de Kibana

Accedemos a Kibana desde: http://localhost:5601

En el menú lateral izquierdo, navegamos a:

Stack Management → Index Patterns.

Creamos un nuevo patrón de índice para visualizar los datos almacenados en Elasticsearch.

6.5 Configuración de Grafana

Accedemos a Grafana desde: http://localhost:3000

Importamos el archivo **grafanaDashboard.json** desde el menú de dashboards.

Con esta importación, tendremos disponible el panel de control con las gráficas configuradas.

Pasados unos minutos, los gráficos comenzarán a mostrar datos actualizados, ya que la API se consulta cada 5 minutos mediante Node-RED.

6.6 Funcionalidades adicionales

Si se desea visualizar alertas meteorológicas (por ejemplo, temperaturas extremas), se puede importar el panel desde **kibanaDashboard.ndjson.**

Para mantener un registro completo de los datos históricos recogidos durante el desarrollo del proyecto, se puede importar los archivos **weatherdb.datosBarcelona.json, weatherdb.datosValencia.json** y **weatherdb.datoMadrid.json** en MongoDB mediante una herramienta como MongoDB Compass o la línea de comandos.

7. Conclusiones y trabajo futuro

Este proyecto ha permitido aplicar tecnologías clave del ecosistema Big Data como Kafka, Elasticsearch, MongoDB y Grafana en un caso real: la monitorización de datos meteorológicos. Gracias al uso de Node-RED, fue posible integrar diferentes componentes de forma visual y rápida, automatizando la obtención, el envío y el almacenamiento de datos desde una API externa.

Entre los logros principales destacan:

- La ingesta continua de datos meteorológicos cada 5 minutos.
- El almacenamiento organizado en múltiples sistemas según su propósito (MongoDB para histórico, Elasticsearch para visualización, Kafka para eventos).
- La visualización clara y en tiempo real de variables climáticas mediante Grafana.
- La detección y alerta de condiciones extremas.

7.1 Trabajo futuro

Aunque el sistema ya es funcional, existen mejoras posibles:

- Integrar una capa de notificaciones (por ejemplo, email o Telegram) para alertas en tiempo real.
- Aplicar análisis predictivo con herramientas de machine learning sobre los datos históricos.
- Implementar una interfaz gráfica propia para usuarios no técnicos.
- Almacenamiento en la nube como AWS.

8. Referencias

1. **Node-RED** – Documentación oficial
   https://nodered.org/docs/
2. **Apache Kafka** – Documentación oficial
   https://kafka.apache.org/documentation/
3. **MongoDB** – Manual oficial
   https://www.mongodb.com/docs/
4. **Elasticsearch** – Guía oficial
   https://www.elastic.co/guide/en/elasticsearch/reference/
5. **Grafana** – Documentación oficial
   https://grafana.com/docs/
6. **Kibana** – Documentación oficial
   https://www.elastic.co/guide/en/kibana/current/index.html
7. **Apache ZooKeeper** – Documentación oficial

https://zookeeper.apache.org/doc/

8. **API meteorológica:**

   https://openweathermap.org/api

9. **Docker** – Documentación de Docker Compose

   https://docs.docker.com/compose/

10. **GitHub del proyecto**

    https://github.com/FerrerJosep/CEIABD-ClimasES