I.E.S. Fernando Aguilar Quignon
C/Conil de la Frontera, 3
CP 11010, Cádiz

5075. Big Data aplicado - 1ª Evaluación (RA 1 – CE b, c)

Unidad Didáctica 1. Estrategias de ingestión y almacenamiento de datos en Big Data

SmartParking Flow: Monitorización inteligente de plazas con Kafka, NiFi, MongoDB, Flask y Dremio.

Como nuevo miembro del equipo de **SmartCity Cádiz**, tu primer reto como ingeniero de datos será desarrollar e implementar **SmartParking Flow**, un sistema avanzado para la recolección, almacenamiento y visualización en tiempo real de los datos generados por los sensores de un aparcamiento inteligente.

El proyecto tiene dos objetivos principales:

En primer lugar, optimizar el flujo de datos en tiempo real utilizando **Apache Kafka**, que gestionará la recepción continua de los mensajes enviados por los sensores instalados en cada plaza de aparcamiento. Estos sensores transmitirán información sobre si cada plaza está **ocupada o libre**, así como otros datos relevantes, como la temperatura, el nivel del parking o el estado de la batería del dispositivo. A continuación, presentamos un ejemplo del documento asociado:

```
{
  "bay_id": "L1-A-023",
  "parking_id": "PK-CADIZ-01",
  "level": "L1",
  "occupied": true,
  "last_event_ts": "2025-10-07T10:15:30Z",
  "metrics": {
    "temperature_c": 23.4,
```

```
    "battery_pct": 78
  },
  "updated_at": "2025-10-07T10:15:31Z"
}
```

En segundo lugar, deberás implementar un sistema que almacene y gestione estos datos en **MongoDB**, una base de datos NoSQL orientada a documentos que permitirá tanto mantener un histórico de eventos como consultar el estado actual de cada plaza de forma ágil y eficiente.

Tu papel será clave en la orquestación del flujo de datos utilizando **Apache NiFi**, que automatizará el proceso completo desde la ingestión de mensajes de Kafka hasta la inserción de documentos en MongoDB. NiFi garantizará que los datos fluyan correctamente, se validen y actualicen el estado de cada plaza de forma continua. Para ello, el flujo deberá distinguir entre dos tipos de almacenamiento:

- Una colección de **eventos históricos**, donde se registran todas las detecciones de los sensores.
- Una colección de **estado actual**, que se actualiza mediante operaciones *upsert* para reflejar en cada momento si la plaza está libre u ocupada.

Además, como parte de la capa de presentación, desarrollarás una **aplicación web ligera en Python con Flask**, que mostrará un **mapa visual del parking** donde cada plaza aparecerá **en verde si está libre** o **en rojo si está ocupada**. La web se actualizará automáticamente cada pocos segundos, consultando los datos de MongoDB y ofreciendo así una visión en tiempo real de la disponibilidad del aparcamiento.

Por último, se integrará **Dremio** como herramienta de análisis para realizar consultas descriptivas (mínimo 5 consultas significativas y distintas) y extraer información agregada sobre la ocupación del parking, las horas de mayor demanda y la eficiencia en el uso de las plazas, complementando la visualización proporcionada por la web.

Objetivos del Proyecto:

- **Configurar y desplegar Apache Kafka** para gestionar la transmisión en tiempo real de los datos generados por los sensores del aparcamiento, garantizando una comunicación continua, fiable y de baja latencia entre los dispositivos y el sistema central.
- **Orquestar la ingesta y el procesamiento de datos con Apache NiFi**, automatizando el flujo desde los tópicos de Kafka hasta su almacenamiento en MongoDB, y asegurando que cada mensaje se valide, se transforme y se inserte correctamente en las colecciones correspondientes.
- **Configurar MongoDB** como sistema de almacenamiento principal, estructurando dos colecciones diferenciadas: una para el **histórico de eventos** y otra para el **estado actual** de cada plaza, optimizando las operaciones de lectura y actualización en tiempo real.
- **Desarrollar una aplicación web con Flask** que consuma los datos almacenados en MongoDB y muestre un **mapa visual del aparcamiento**, donde cada plaza aparezca en color verde o rojo según su disponibilidad, actualizándose automáticamente cada pocos segundos.
- **Integrar Dremio como plataforma de análisis de datos**, permitiendo ejecutar consultas descriptivas sobre la ocupación del parking, el uso por niveles o franjas horarias, y otros indicadores útiles para la gestión y optimización del espacio.
- **Garantizar un flujo de datos robusto, escalable y automatizado**, desde los sensores hasta la visualización y el análisis, asegurando la fiabilidad del sistema y facilitando la monitorización en tiempo real del aparcamiento.

Implementación técnica:

El proyecto integrará diversas tecnologías clave en el ámbito del Big Data y la analítica en tiempo real para garantizar la **captura, transmisión, almacenamiento y visualización** de los datos generados por los sensores del aparcamiento inteligente.

• **Apache Kafka:** actuará como el componente central para la transmisión de datos en tiempo real. Este sistema de mensajería distribuido se configurará para recibir los mensajes enviados por los sensores instalados en cada plaza del aparcamiento. Cada sensor publicará información sobre el estado de la plaza —ocupada o libre—, además de posibles métricas adicionales como la temperatura o el nivel de batería del dispositivo. Kafka garantizará una **comunicación continua y tolerante a fallos** entre los sensores y los sistemas de ingesta, permitiendo manejar flujos constantes de mensajes sin pérdida de información.

• **Apache NiFi:** desempeñará un papel fundamental en la orquestación del flujo de datos desde Kafka hasta MongoDB. Gracias a su interfaz visual y a su capacidad para manejar datos en tiempo real, NiFi permitirá **automatizar y monitorizar** todo el proceso de ingesta. A través de sus procesadores, NiFi validará, transformará y enrutará los mensajes recibidos, almacenando una copia en la colección de **eventos históricos** y actualizando en paralelo la colección de **estado actual** mediante operaciones *upsert*. Esta doble estrategia permitirá mantener tanto el registro completo de eventos como la información en tiempo real sobre la disponibilidad de las plazas. NiFi también facilitará la **escalabilidad** del sistema y la **gestión centralizada** de los flujos de datos.

• **MongoDB:** será la plataforma de almacenamiento principal del proyecto. Se elegirá por su flexibilidad y su modelo orientado a documentos, que se adapta perfectamente a la naturaleza variable de los datos de sensores. En MongoDB se configurarán dos colecciones: events, que almacenará todos los mensajes históricos recibidos desde Kafka, y bays, que mantendrá el estado actual de cada plaza. Esta estructura permitirá consultas rápidas, operaciones de actualización eficientes y un almacenamiento escalable sin necesidad de un esquema rígido. Además, MongoDB servirá como fuente directa de datos tanto para la web desarrollada en Flask como para Dremio.

• **Flask:** será el framework web encargado de ofrecer una **interfaz visual e interactiva** del aparcamiento. Se desarrollará una aplicación ligera en Python que mostrará un **mapa dinámico de las plazas**, donde cada una se representará con un color según su estado: verde si está libre y rojo si está ocupada. La aplicación consultará periódicamente los datos de MongoDB, actualizando la vista, cada pocos segundos, para reflejar los cambios en tiempo real. Su sencillez permitirá un despliegue rápido, bajo consumo de recursos y facilidad de mantenimiento.

• **Dremio:** se implementará como la plataforma de **análisis y consulta de datos** del sistema. A través de su conector nativo para MongoDB, Dremio permitirá ejecutar consultas SQL interactivas sobre los datos almacenados, facilitando la obtención de información agregada sobre el uso del aparcamiento, los niveles de ocupación o la eficiencia del sistema. Su motor de ejecución acelerado ofrecerá una exploración fluida de los datos sin necesidad de replicarlos ni transformarlos previamente, integrándose así de forma natural con el flujo NiFi → MongoDB.

• **Optimización y pruebas:** se validará el rendimiento de MongoDB bajo cargas concurrentes, y se verificará que tanto la web Flask como Dremio reflejan correctamente los cambios en tiempo real.

Con esta arquitectura técnica se logrará un **sistema robusto, escalable y automatizado**, capaz de ofrecer una visión completa y actualizada del estado del aparcamiento. Los usuarios podrán **monitorizar la disponibilidad de plazas en tiempo real** y realizar análisis descriptivos sobre la ocupación, garantizando una gestión eficiente del espacio y una experiencia mejorada para los conductores.

Además de la implementación, se deberá entregar un vídeo de 5 minutos máximo mostrando el funcionamiento de la plataforma. También, se desarrollará una memoria del proyecto (mínimo 50 páginas y desarrollado con LaTeX) que constará de los siguientes apartados:

1. Introducción
    a. Contextualización
    b. Justificación
    c. Objetivos
    d. Alcance del proyecto
    e. Planificación
2. Marco teórico
3. Fase de análisis
4. Fase de diseño
5. Fase de implementación
6. Fase de pruebas
7. Conclusiones y líneas futuras
8. Bibliografía
9. Diario de trabajo (anexo)
10. Anexos varios

Cualquier sospecha y comprobación de plagio será penalizada, obteniendo el trabajo la calificación de 0 puntos.