Enunciado de proyecto integral para ingeniería de datos. El proyecto sigue el ciclo de vida **KDD** (*Knowledge Discovery in Databases*) y utiliza el ecosistema **Apache** en sus versiones más recientes (2025-2026).

Proyecto:

1. Contexto del Proyecto

Una empresa ficticia necesita una plataforma de **Big Data** capaz de monitorizar su red de transporte global. El objetivo es cruzar datos de sensores en tiempo real con información histórica y grafos de rutas para predecir cuellos de botella y optimizar la logística.

Objetivos Técnicos:

- Implementar una arquitectura **Lambda/Kappa** 100% basada en Apache.
- Automatizar el ciclo KDD: Selección, Preprocesamiento, Transformación, Minería e Interpretación.
- Garantizar la escalabilidad mediante **HDFS** y **YARN**.

---

2. Requisitos Técnicos (Stack Apache 2026)

Se exige el uso de las siguientes versiones **(o superiores)**:

- **Ingesta:** Apache NiFi 2.6.0 & Apache Kafka 3.9.1 (KRaft mode).
- **Procesamiento:** Apache Spark 3.5.x (con Spark SQL, Structured Streaming y GraphFrames).
- **Orquestación:** Apache Airflow 2.10.x.
- **Almacenamiento:** HDFS 3.4.2 & Apache Cassandra 5.0 (NoSQL) / Apache Hive (SQL).
- **Gestión de Recursos:** YARN.

---

3. Fases del Proyecto (Ciclo KDD)

Fase I: Ingesta y Selección (NiFi + Kafka)

1. **Fuentes Externas:** Configurar NiFi para consumir datos de una API pública (ej: OpenWeather o FlightRadar24) y archivos logs simulados de GPS.
2. **Streaming:** Publicar los eventos en temas de **Kafka**. Se deben diferenciar temas para "Datos Crudos" y "Datos Filtrados".
3. **Registro:** Almacenar una copia "raw" en **HDFS** para auditoría.

Fase II: Preprocesamiento y Transformación (Spark)

1. **Limpieza:** Utilizar **Spark SQL** para normalizar formatos, gestionar nulos y eliminar duplicados.
2. **Enriquecimiento:** Cruzar el streaming de Kafka con datos maestros almacenados en **Hive**.
3. **Análisis de Grafos:** Usar **GraphFrames** para modelar la red de transporte (Nodos: Almacenes; Aristas: Rutas) y calcular el camino más corto o detectar comunidades críticas.

Fase III: Minería y Acción (Streaming + ML)

1. **Ventanas de Tiempo:** Implementar Structured Streaming para calcular la media de retrasos en ventanas de 15 minutos.
2. **Carga Multicapa:** * **Relacional (Hive):** Datos agregados para reporting histórico.
    - **NoSQL (Cassandra):** Último estado conocido de cada vehículo para consultas de baja latencia.

Fase IV: Orquestación (Airflow)

- Crear un DAG que coordine el re-entrenamiento mensual del modelo de grafos y la limpieza de tablas temporales en HDFS.

---

4. Rúbrica de Evaluación

Criterio	Excelente (10)	Adecuado (6-7)	Insuficiente (<5)
Arquitectura de Ingesta	NiFi y Kafka integrados con back-pressure y manejo de errores.	Ingesta funcional pero sin control de fallos robusto.	Fallos en la conexión o pérdida de datos.
Procesamiento Spark	Uso avanzado de GraphFrames, SQL y Streaming. Optimización de joins.	Procesamiento básico sin uso de grafos o mal optimizado.	El código Spark es ineficiente o falla en YARN.

Persistencia	Uso correcto de Cassandra (NoSQL) y Hive (SQL) según el caso de uso.	Solo utiliza un tipo de base de datos.	Errores en la escritura o esquema mal diseñado.
Orquestación Airflow	DAGs complejos con reintentos, alertas y dependencias claras.	DAG lineal simple sin gestión de errores.	No utiliza Airflow o el DAG no funciona.
Documentación	Detalla cada etapa del KDD, diagramas de flujo y justificación técnica.	Documentación técnica básica.	Documentación escasa o inexistente.

---

5. Ejemplo de Referencia

Un proyecto similar de referencia es el **"NYC Taxi & Uber Data Pipeline"**.

- **Datos:** Datasets de viajes de NYC (disponibles en Kaggle/NYC Open Data).
- **Implementación:** 1. NiFi recoge archivos de APIs.
  2. Kafka distribuye las coordenadas de los viajes.
  3. Spark calcula en tiempo real las zonas con más demanda (Heatmaps).
  4. **GraphFrames** identifica las intersecciones más congestionadas.
  5. Airflow genera un reporte diario de ingresos y lo guarda en Hive.