**1. Selección del caso**

Elegimos **detección de fraude en tiempo real**, donde se procesan transacciones financieras a medida que ocurren, buscando patrones sospechosos como montos inusuales, ubicaciones inesperadas o comportamientos atípicos.

**2. Componentes del ecosistema Spark utilizados**

**🔧 Módulos:**

* **Spark Streaming / Structured Streaming**: para recibir y procesar datos en tiempo real desde fuentes como Kafka.
* **Spark SQL**: para aplicar reglas y filtros sobre los datos estructurados.
* **MLlib**: para aplicar modelos de machine learning entrenados previamente que detecten fraude.
* **Spark Core**: para el procesamiento distribuido básico.

**📦 Estructuras de datos:**

* **DataFrames**: para trabajar con datos estructurados (transacciones).
* **RDDs**: si se requiere procesamiento más bajo nivel o personalizado.

**🧪 APIs:**

* API de **Structured Streaming** para manejar flujos de datos.
* API de **MLlib** para aplicar modelos predictivos.

**3. Representación gráfica del procesamiento distribuido**

**🖥️ Arquitectura distribuida:**

1. **Driver Program**:
   * Coordina la ejecución de la aplicación Spark.
   * Define la lógica de negocio para detectar fraude.
2. **Cluster Manager**:
   * Asigna recursos a los nodos del clúster.
   * Puede ser YARN, Mesos, Kubernetes o el gestor nativo de Spark.
3. **Executors**:
   * Ejecutan tareas en paralelo sobre particiones de datos.
   * Aplican transformaciones, filtros y modelos ML.
4. **Particiones**:
   * Los datos se dividen en bloques para ser procesados en paralelo.
5. **Flujo de datos**:
   * Kafka → Spark Streaming → DataFrame → MLlib → Alerta/Almacenamiento

**4. Justificación del uso de Spark**

**✅ Ventajas de Spark frente a otras herramientas:**

* **Procesamiento en memoria**: mucho más rápido que Hadoop (que usa disco).
* **Escalabilidad**: puede manejar grandes volúmenes de datos en tiempo real.
* **Integración de módulos**: permite combinar streaming, SQL y ML en una sola plataforma.
* **Flexibilidad**: soporta múltiples lenguajes (Python, Scala, Java).
* **Structured Streaming**: ideal para flujos de datos continuos como transacciones bancarias.

**❌ Comparación con otras herramientas:**

* **Hadoop**: más lento, no diseñado para tiempo real.
* **Flink**: muy bueno en streaming, pero menos integrado con MLlib.
* **Storm**: especializado en eventos, pero menos flexible para análisis complejos.