# Examen Segunda evaluación

Tendréis que gestionar datos provenientes de distintas fuentes, integrarlos y analizarlos en un clúster de Hadoop. Tendréis que utilizar Python para manipular archivos, interactuar con MySQL, MongoDB y Neo4j y almacenar resultados en HDFS. Además, tendréis que trabajar con Pig para limpiar y analizar la información y Sqoop para exportar resultados.

# Parte 1: Generación de Datos

## 1.1 Generar documentos CSV, TXT y JSON

Crear un archivo llamado **fileGen.py** que genere todos estos documentos.

### Archivo CSV - Horarios de Líneas (horarios.csv) **100 filas**

**Ubicación:** ./datos/horarios.csv

| Columna | Tipo | Descripción |
|---|---|---|
| linea | STRING | Identificador de la línea [L1,L100] |
| hora_inicio | STRING | Hora de inicio del servicio |
| hora_fin | STRING | Hora de finalización del servicio |

**Ejemplo de contenido (horarios.csv):**

```
L1,06:00,23:00
L2,05:30,22:30
L3,07:00,21:45
```

### Archivo TXT - Conductores y Vehículos (conductores.txt) **2000 filas**

**Ubicación:** ./datos/conductores.txt

| Columna | Tipo | Descripción |
|---|---|---|
| id_conductor | STRING | Identificador único del conductor |
| nombre | STRING | Nombre del conductor |

| | | |
|---|---|---|
| vehiculo | STRING | [Bus_1,Bus_100] o [Tranvia_1,Tranvia_100] |
| linea | STRING | Identificador de la línea [L1,L100] |

**Importante:** un vehículo puede ser operado por más de un conductor, independientemente que sea la misma línea u otra.

**Ejemplo de contenido (conductores.txt):**

```
C001,Juan Pérez,Bus_11,L1
C002,Ana López,Tranvia_25,L2
C003,Carlos Ruiz,Bus_30,L3
```

## Archivo JSON - Tarifas por Línea (tarifas.json) **100 filas**

**Ubicación:** ./datos/tarifas.json

| Campo | Tipo | Descripción |
|---|---|---|
| linea | STRING | Identificador de la línea [L1,L100] |
| precio | FLOAT | Precio del billete por una parada |
| descuento | FLOAT | Descuento aplicado (por ejemplo, para estudiantes) |

**Ejemplo de contenido (tarifas.json):**

```
[
  {"linea": "L1", "precio": 1.50, "descuento": 0.20},
  {"linea": "L2", "precio": 2.00, "descuento": 0.30},
]
```

# 1.2 Bases de Datos

## Base de Datos MongoDB **2000 documentos**

Crear un archivo llamado **fileGen.py** que genere todos estos documentos.

Ubicación: MongoDB (localhost:27017, db: transporte, colección: posiciones)

| Campo | Tipo | Descripción |
|---|---|---|
| _id | ObjectId | ID único de MongoDB |
| vehiculo | STRING | [Bus_1,Bus_100] o [Tranvia_1,Tranvia_100] |
| latitud | DOUBLE | Posición geográfica |
| longitud | DOUBLE | Posición geográfica |
| timestamp | DATETIME | Fecha y hora de la posición |

**Ejemplo de documento en MongoDB (posiciones):**

```
{
    "_id": ObjectId("65abcd123456"),
    "vehiculo": "Bus_101",
    "latitud": 42.465,
    "longitud": -2.437,
    "timestamp": "2025-02-14T08:30:00Z"
}
```

## Base de Datos MySQL **2000 documentos**

Crear un archivo llamado **MysqlGen.py** que genere todos estos documentos.

Ubicación: MySQL (localhost:3306, db: transporte)

Tabla tickets - Gestión de Pagos y Tickets

| Columna | Tipo | Descripción |
|---|---|---|
| id | INT (PK) | Identificador único del ticket |
| vehiculo | STRING | [Bus_1,Bus_100] o [Tranvia_1,Tranvia_100]. Importante: un vehículo puede operar en distintas líneas. |
| linea | STRING | Identificador de la línea [L1,L100] |
| monto | FLOAT | Precio del ticket, tiene en cuenta todas las paradas que se harán en el viaje |
| fecha | DATETIME | Fecha de la compra |

```
INSERT INTO tickets (id, vehiculo, linea, monto, fecha) VALUES
(1, 'Bus_101', 'L1', 1.50, '2025-02-14 08:00:00'),
(2, 'Tranvia_205', 'L2', 2.00, '2025-02-14 08:15:00');
```

## Base de Datos Neo4j

Ya existe un **neo4jGen.py** que genera los nodos y relaciones.

Ubicación: Neo4j (bolt://localhost:7687)

**Nodos**

- (:Parada {id, nombre}) → Representa una parada de autobús o tranvía.
- (:Vehiculo {id, linea}) → Representa un vehículo de transporte.

**Relaciones**

- (:Vehiculo)-[:PARA_EN]->(:Parada) → Indica que un vehículo hace parada en una ubicación.

**IMPORTANTE:** Solo hay **30** paradas, IDs posibles vehículos: [Bus_1,Bus_100] o [Tranvia_1,Tranvia_100], IDS posiblers líneas [L1,L100]

# Parte 2: Integración y Análisis de Datos

Los resultados de estas preguntas se tendrán que almacenar en formato CSV en Hadoop mediante WebHDFS y Python y los guardará en /data/raw.

- **Pregunta 1:** Por cada línea ¿Cuáles son los horarios de inicio y fin de la línea, qué conductores está asignado a esa línea, cuántos vehículos tiene asignado esa línea , y cuál es el precio y descuento medio del billete por línea?
    - `linea`: **string** (Identificador de la línea de transporte)
    - `hora_inicio`: **string** (Hora de inicio del servicio, en formato HH:MM)
    - `hora_fin`: **string** (Hora de fin del servicio, en formato HH:MM)
    - `conductores`: **list of strings** (Lista de nombres del conductor asignado)
    - `vehiculos`: **list of strings** (Lista de vehículos de la línea)
    - `precio`: **float** (Precio del billete para esa línea, en formato decimal)
    - `descuento`: **float** (Descuento aplicado al billete, en formato decimal)

- **Pregunta 2:** Por cada vehículo ¿Cuáles son las posiciones actuales, qué líneas están operando, y qué paradas tienen disponibles estos vehículos?
    - `vehiculo`: **string** (Identificador o nombre del vehículo)
    - `latitud`: **float** (Posición geográfica del vehículo en coordenadas de latitud)
    - `longitud`: **float** (Posición geográfica del vehículo en coordenadas de longitud)
    - `timestamp`: **string** (Marca temporal del momento en que se obtuvo la posición, en formato ISO 8601)
    - `lineas`: **list of strings** (Lista de líneas asociadas al vehículo)
    - `paradas`: **list of strings** (Lista de paradas asociadas al vehículo)

- **Pregunta 3:** Devuelve los 50 vehículos con mayor número de paradas,el importe generado en todas sus líneas, el importe total generado por todos los vehículos asociados a sus líneas y los nombres de conductores que lo utilizan.
    - `vehiculo`: **string** (Identificador o nombre del vehículo)

- `numParadas`: **float** (num de paradas)
- `importeLinea`: **float**(importe de todas sus líneas)
- `importeTodosVehiculos`: **float**(importe de todas sus líneas)
- `conductores`: **list of strings** (Lista de conductores que lo utilizan)

# Parte 3: Procesamiento en Hadoop

## 3.1 Limpieza de Datos con Pig

Crear un script en Pig llamado **Clean.pig** que elimine filas con datos vacíos o nulos en los 3 archivos CSV que se han almacenado en el apartado anterior y los vuelve a guardar /data/raw de HDFS.

## 3.2 Consulta con Pig

De la Pregunta 3, crea un archivo llamado **ConductorTop.pig** devuelve el nombre del conductor que más veces aparece en el archivo 3 y la fecha actual y lo guardáis en la siguiente carpeta de HDFS resultados/conductor_mas_repetido.

**PISTA:** fecha_actual = FOREACH conductor_mas_repetido GENERATE conductor, CURRENT_TIME() AS fecha;

## 3.3 Exportación con Sqoop

Exporta a MySQL utilizando Sqoop el resultado de la consulta anterior en Pig a una Database llamada **DataWarehouse** y una tabla llamada **TopConductor**. Crea un archivo llamado **Sqoop.txt** donde esté el comando.