2º DAM – Acceso a Datos

Examen 1º Evaluación: Gestión de Plataforma de Streaming

En este examen, desarrollarás una aplicación Java que integra múltiples fuentes de datos para poblar una base de datos relacional y una base de datos NoSQL.

Contexto

La empresa StreamIt ofrece una plataforma de streaming donde los usuarios pueden explorar y reproducir contenidos.

Para mejorar la experiencia de los clientes, StreamIt ha decidido migrar los datos de usuarios y reproducciones almacenados en archivos JSON a sistemas de bases de datos: una base relacional en PostgreSQL y una base no relacional en DynamoDB.

Tu tarea es desarrollar una solución que procese los datos existentes, los almacene en las bases de datos correspondientes y permita realizar consultas sobre los mismos para analizar el comportamiento de los usuarios y la popularidad de los contenidos.

Descripción del sistema

Tienes dos ficheros como fuentes de datos:

usuarios.json: Contiene los nombres completos y correos electrónicos de los usuarios.

```
[
  {
    "nombre": "Paulina Encinas",
    "email": "paulinaencinas@gmail.com"
  },
```

reproducciones.json: Contiene los nombres completos y correos electrónicos de los usuarios.

```
[
  {
    "usuario": "paulinaencinas@gmail.com",
    "reproducciones": [
      { "contenido": "Breaking Bad", "fecha": "2024-11-01T20:30:00" },
      { "contenido": "Game of Thrones", "fecha": "2024-11-02T18:45:00" },
      { "contenido": "Friends", "fecha": "2024-11-03T22:15:00" }
    ]
  },
```

En Amazon Web Service tendrás **ya creadas** dos bases de datos donde almacenar la información:

PostgreSQL

Tienes una base de datos con el nombre de tu correo GVA (sin números). Por ejemplo, si mi email de alumno es jaiforbal3@alu.edu.gva.es, mi base de datos se llama **jaiforbal**.

La tabla **usuarios** contendrá la información básica del usuario:

```
CREATE TABLE IF NOT EXISTS usuarios (
  id_usuario SERIAL PRIMARY KEY,
  nombre_completo VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE
);
```

La tabla contenidos_reproducidos almacena las reproducciones asociadas a un usuario:

```
CREATE TABLE IF NOT EXISTS contenidos_reproducidos (
  id SERIAL PRIMARY KEY,
  id_usuario INT NOT NULL,
  contenido VARCHAR(100) NOT NULL,
  fecha TIMESTAMP NOT NULL,
  FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario));
);
```

DynamoDB

La tabla se llama **usuarioGVA-historial**, sin número. En mi caso sería jaiforbal-historial, y su estructura es la siguiente:

- Clave principal: idUsuario (generado por PostgreSQL en la tabla usuarios)
- Atributos: email y reproducciones (lista de objetos con contenido y fecha).

Requisitos de la aplicación

Tu aplicación debe cumplir los siguientes **requerimientos**:

- Lectura de ficheros:
  - Lee y procesa los dos archivos de entrada con la biblioteca Jackson mediante el uso de clases y anotaciones para convertir los datos leídos en objetos Java.
- PostgreSQL:
  - Almacena la información leída de usuarios y sus reproducciones asociadas.
  - Ejecuta **consultas** para:
    - Obtener los contenidos reproducidos para un usuario específico.
      List<String> obtenerContenidosPorUsuario(String email)
    - Listar los emails de usuarios que han reproducido un contenido determinado.
      List<String> obtenerUsuariosPorContenido(String contenido)
- DynamoDB:
  - Almacena el historial completo de reproducciones en formato no relacional.
  - Ejecuta **consultas** para:
    - Obtener el historial completo de un usuario.
      List<Reproduccion> obtenerHistorialPorUsuario(String email);
    - Contar las veces que un contenido específico ha sido reproducido.
      int contarReproduccionesPorContenido(String contenido);
- Funcionalidad adicional:
  - Borra los usuarios y sus reproducciones en ambas bases de datos, respetando las relaciones y referencias.
    void borrarUsuarioPorEmail(String email);
  - Gestiona las conexiones a bases de datos de manera eficiente, intentando utilizar try-with-resources cuando sea posible.
  - Diseña un código modular y organizado que facilite su legibilidad.

Entrega

Comprime el proyecto realizado en un fichero .zip y adjúntalo a la tarea correspondiente en la plataforma AULES.

Rúbrica

Aspecto	10 (Excelente)	4 (Regular)	0 (Mal)
Lectura de ficheros (2,5 pts)			
Procesado de CSV y JSON.	Los ficheros se leen sin errores y los datos se cargan correctamente en las clases	Algunos datos no se leen o están incompletos por errores en el código o falta de validaciones	No se leen los ficheros o el proceso genera fallos continuos que impiden cargar los datos.
Herramientas (OpenCSV, Jackson).	Se utilizan correctamente las librerías asignadas para simplificar el procesamiento de ficheros.	Se implementa manualmente parte del procesamiento sin necesidad, afectando eficiencia o claridad.	No se utiliza ninguna de las librerías mencionadas, o el código no logra leer los ficheros.
Manejo de excepciones y recursos.	El código usa try-with-resources y gestiona adecuadamente las excepciones (sin interrumpir el flujo).	Hay intentos de manejo de excepciones, pero el código es inconsistente o provoca fallos.	No hay manejo de excepciones o no se cierran recursos, generando fugas o fallos graves.
PostgreSQL (2,5 pts)			
Inserción de datos en PostgreSQL.	Todos los usuarios y reproducciones se insertan correctamente, respetando las relaciones.	Solo se insertan parcialmente los datos, con múltiples errores o relaciones inconsistentes.	Los datos no se insertan, no se respetan las relaciones o el código falla al ejecutar las operaciones.
Consultas (PostgreSQL).	Las consultas devuelven los resultados correctos.	Una consulta devuelve resultados incompletos o erróneos.	Las consultas no funcionan o devuelven datos incorrectos.
Eficiencia y buenas prácticas.	El código es eficiente, seguro y evita operaciones innecesarias.	Hay alguna ineficiencia significativa.	El código es ineficiente, inseguro o no sigue estándares mínimos.
DynamoDB (3 pts)			
Inserción de datos en DynamoDB.	Los historiales se insertan correctamente, respetando la estructura propuesta.	Solo se inserta parte de la información, o se generan inconsistencias en los datos.	Los datos no se insertan en DynamoDB, o el proceso genera errores graves continuos.
Consultas (DynamoDB).	Las consultas devuelven datos correctos implementando patrones eficientes.	Una consulta devuelve resultados incompletos o erróneos.	Las consultas no funcionan o devuelven datos incorrectos.
Uso del SDK.	El código usa correctamente las clases y métodos del SDK de AWS v2.	La implementación es confusa o ineficiente, aunque funcional.	No se usa correctamente el SDK, o el código genera fallos al interactuar con DynamoDB.
Otras funcionalidades y calidad del código (2 pts)			
Eliminación de datos.	Los usuarios y reproducciones se eliminan correctamente en ambas bases de datos.	Se eliminan solo algunos datos, dejando registros inconsistentes o huérfanos.	No se logra la eliminación correcta o el proceso genera errores graves.
Estructura y modularidad.	El proyecto está bien organizado, con carpetas y clases diferenciadas por responsabilidades.	La estructura es confusa, con clases poco cohesivas o mal organizadas.	La estructura es caótica o todo el código está en una sola clase de forma desordenada.