Prueba Práctica: API REST para Gestión de Empleados

Objetivo General

Desarrollar una **API REST en Java** (sin frameworks) que permita gestionar empleados, utilizando **procedimientos almacenados (Stored Procedures)** en una base de datos **MySQL alojada en AWS RDS**. La API debe desplegarse en **AWS Lambda** y exponerse mediante **Amazon API Gateway**. Además, se debe incluir un servicio adicional que utilice **hilos (threads)** para procesar datos de empleados desde tres archivos JSON y devolver a los 10 empleados con los salarios más altos.

Requisitos Previos

1. **Herramientas y Configuración Local:**
   - **Java JDK 17** o superior.
   - **Maven** para gestionar dependencias y construir el proyecto.
   - Cliente **MySQL** para gestionar la base de datos.
   - Una cuenta **Free Tier AWS** para desplegar los servicios.

Requerimientos

Funcionalidad de la API REST

1. **Recursos:**
   - /employees**:**
     - **POST**: Crear un empleado.
     - **GET**: Obtener todos los empleados.
   - /employees/{id}**:**
     - **GET**: Obtener la información de un empleado específico.
     - **PUT**: Actualizar información de un empleado.
     - **DELETE**: Eliminar un empleado por ID.
2. **Recurso Adicional:** /employees/salary/top

   GET:
   - Procesa los datos de empleados desde tres fuentes (archivos JSON).
   - Devuelve una lista de los **10 empleados con los salarios más altos** ordenados de forma descendente.

Persistencia

- Los datos deben almacenarse en una base de datos **MySQL alojada en AWS RDS**.
- La tabla employees debe incluir los siguientes campos:
    1. **id**: Identificador único del empleado (clave primaria).
    2. **name**: Nombre completo del empleado.
    3. **position**: Puesto o cargo que ocupa en la organización.
    4. **salary**: Salario del empleado.
    5. **hire_date**: Fecha de contratación del empleado.
    6. **department**: Departamento al que pertenece el empleado.

Procedimientos Almacenados (Stored Procedures)

- Se deben crear procedimientos almacenados en **MySQL** para realizar todas las operaciones de la API (CRUD).

Despliegue en AWS

1. **AWS RDS:** La base de datos debe alojarse en **AWS RDS**.
2. **AWS Secrets Manager:** Las credenciales de la base de datos deben gestionarse mediante **AWS Secrets Manager**.
3. **AWS Lambda:** La lógica de la API debe implementarse como una función **AWS Lambda**.
4. **Amazon API Gateway:** La API debe ser expuesta públicamente mediante **API Gateway**, configurada como proxy.

Buenas Prácticas

1. Uso de principios **SOLID**.
2. Separación de responsabilidades en el código.
3. Manejo adecuado de excepciones.
4. Código limpio y bien comentado.

Recurso Adicional: /employees/salary/top

1. **Descripción:**
   - Procesa los datos de empleados desde tres fuentes (archivos JSON).
   - Devuelve una lista de los **10 empleados con los mayores salarios**, ordenados de forma descendente.
2. **Flujo de Trabajo:**
   - Cada archivo JSON es leído por un hilo independiente.
   - Los resultados son combinados y ordenados en el hilo principal.
   - La API responde con los 10 empleados con los mayores salarios.

Documentación Requerida

1. **Capturas de Pantalla:**
   - Configuración de **AWS RDS**.
   - Creación del secreto en **AWS Secrets Manager**.
   - Configuración de **API Gateway**.
   - Logs de prueba en **AWS Lambda**.
   - Ejecución de los endpoints mediante **Postman** o **cURL**.
2. **Archivos a adjuntar para evaluación de la práctica:**
   - **Postman Collection** con ejemplos de consumo y respuestas de los endpoints.
   - Código SQL de los stored procedures y el DDL para la tabla employees.

Archivos JSON Adjuntos

Se adjuntan los siguientes archivos JSON con datos de empleados para el recurso /employees/salary/top:

1. **employees_data1.json**
2. **employees_data2.json**
3. **employees_data3.json**

Especificaciones para configuración de servicios AWS (Free Tier)

1. **AWS RDS (Free Tier)**

Instancia: Hasta 750 horas al mes de uso (equivalente a una instancia en ejecución 24/7).

- Tipo de Instancia: db.t2.micro o db.t3.micro.
- Almacenamiento: Hasta 20 GB para almacenamiento de base de datos.
- Backup: Hasta 20 GB para snapshots y backups automáticos.

2. **AWS Secrets Manager (Free Tier)**

Límites Gratuitos

- Primero 1,000,000 de solicitudes mensuales gratis:
- Esto incluye operaciones como:
- GetSecretValue
- ListSecrets
- CreateSecret
- UpdateSecret
- DeleteSecret

3. **AWS Lambda (Free Tier)**

Límites Gratuitos

- 750,000 segundos de cómputo por mes:
- Esto incluye la suma del tiempo que Lambda ejecuta tu código en todas las invocaciones.
- Por ejemplo:
- Si tu función tarda 100 ms y se ejecuta 100,000 veces al mes, usas solo 10,000 segundos.
- 1 millón de invocaciones gratuitas por mes:
- Aplica a todas las invocaciones de funciones Lambda, ya sean iniciadas directamente o a través de otros servicios (como API Gateway).

4. AWS API Gateway (Free Tier)

Amazon API Gateway ofrece una capa gratuita que te permite construir y probar tus APIs sin costos iniciales. Los límites de la Free Tier son los siguientes:

1. 1 Millón de Llamadas API por Mes (REST y HTTP APIs)
   - Aplica a REST APIs y HTTP APIs.
   - Incluye tanto invocaciones internas como públicas.
2. 1 Millón de Mensajes para WebSocket APIs por Mes
   - Aplica a mensajes enviados y recibidos a través de WebSocket APIs.
3. 750,000 Millisegundos de Cómputo por Mes
   - Para APIs que usen integración con AWS Lambda.
   - Esto se calcula en base al tiempo que lleva a API Gateway procesar las solicitudes.