Evaluación: MongoDB

Entrega: 26 de Noviembre de 2024

Una empresa de servicios financieros está desarrollando una plataforma para gestionar y analizar transacciones financieras de sus usuarios. El objetivo es almacenar los datos de transacciones en MongoDB para una alta capacidad de almacenamiento y procesamiento, y luego consolidar esta información en MySQL para generar reportes estructurados que faciliten la toma de decisiones y el análisis histórico. **La plataforma también requiere que los reportes en MySQL se actualicen cada vez que haya un cambio en MongoDB.**

Módulo: GESTIÓN DE TRANSACCIONES

La base de datos MongoDB tendrá una colección llamada `transacciones` que contiene los siguientes campos:

- `id_transaccion`: ID único de la transacción (UUIDv4)
- `monto`: valor de la transacción
- `categoria`: categoría de gasto. **Valores permitidos**: "comida", "ropa", "tecnologia", "salud", "educacion", "transporte", "entretenimiento", "servicios", "otros".
- `fecha`: fecha de la transacción
- `descripcion`: breve descripción de la transacción
- `rut_usuario`: rut del usuario al que pertenece la transacción

Servicios a Construir

1. **Registro de Transacciones:**
   Permitir registrar nuevas transacciones. Se debe recibir como entrada una lista de transacciones en un único request. Se deben registrar todas las transacciones en BD pero todas deben pertenecer al mismo usuario.
   Endpoint: `POST /transaccion/bulk`

2. **Modificar Transacciones:**
   Permitir que el usuario pueda modificar el monto, fecha, categoría y/o descripción de una transacción existente.
   Endpoint: `PATCH /transaccion/:id_transaccion`

3. **Eliminar Transacciones:**
   Permitir que el usuario pueda eliminar una transacción que le pertenezca.
   Endpoint: `DELETE /transaccion/:id_transaccion`

4. **Consulta de Transacciones:**
   El usuario puede consultar sus propias transacciones. Debe contar con estos filtros (opcionales): categoria, montoMenorA, montoMayorA, fechaMenorA y fechaMayorA.
   Endpoint: `GET /transaccion?queries=....`

Todos estos servicios deben contener un header donde se indique el rut del usuario propietario de las transacciones. No debe enviarse el rut del usuario en el body, ni en params, ni en un query. Debe ser en el header `rut_usuario`. En caso, que no se envíe este header responder siempre un error 403.

Módulo: CONSOLIDACIÓN Y GENERACIÓN DE REPORTES

La base de datos MySQL está diseñada para almacenar reportes generados a partir de las transacciones en MongoDB.

Habrá tres tablas en MySQL, cada una para un tipo de reporte específico:

1. **Reporte Mensual por Categoría**
   Una tabla llamada `reporte_mensual_categoria` donde se almacena el gasto total de cada categoría por mes y usuario.

Campos: rut_usuario, mes, año, categoria, total_gasto

Endpoint: `GET /reporte/mensual_categoria/:rut_usuario`

El reporte debe entregar la consulta a esta tabla y mostrar primero los más recientes y luego mostrar los mayores gastos primero.

2. **Reporte de Transacciones por Rango de Monto**
   Una tabla llamada `reporte_rango_monto` para agrupar y contar transacciones en rangos de monto por categoría.

Campos: categoria, rango_monto, cantidad_transacciones

`rango_monto` debe tener al menos 4 divisiones:

Ejemplo1: 0-5.000, 5.001-30.000, 30.001-100.000, 100.001-500.000, >500.001

Ejemplo2: 0-10.000, 10.001-100.000, 100.001-200.000, >500.001

Endpoint: `GET /reporte/rango_monto/:categoria`

El reporte debe entregar la consulta a esta tabla y mostrar primero los rangos más bajos y al final los más altos.

3. **Reporte de Gasto Promedio Diario**
   Una tabla llamada `reporte_promedio_diario` para el cálculo del gasto promedio diario por usuario por categoría.

   Campos: usuario_id, categoria, promedio_diario

Endpoint: `GET /reporte/promedio_diario/:categoria`

El reporte debe entregar la consulta a esta tabla filtrada por categoría y mostrar primero los usuarios con promedio de gasto diario más alto.

Sincronización de Reportes

Cada vez que una transacción es creada, actualizada o eliminada en MongoDB, se debe ejecutar una actualización en MySQL para reflejar estos cambios en los reportes.

- Aplicar operaciones de agregación en MongoDB para consolidar los datos necesarios para los reportes.
- Registrar los resultados de la agregación en las tablas correspondientes de MySQL o actualizarlos si ya existen.
- Luego de cualquier cambio en MongoDB (creación, edición o eliminación de transacciones) se debe ejecutar automáticamente la actualización de los reportes en MySQL.

TAREA A REALIZAR

De acuerdo al enunciado anterior.

1. Crear base de datos de MongoDB y la colección: **transacciones.** Importar los datos utilizando el script: **transacciones.json**.
2. Crear base de datos MySQL con utilizando el script: **reportes.sql**
3. Crear un proyecto NestJS, configurar el Swagger y conectarlo a la BD MySQL y a la BD MongoDB (solamente crear el docker-compose.yml para ambiente de desarrollo).
4. Implementar los 4 servicios solicitados del **Módulo: GESTIÓN DE TRANSACCIONES** utilizando Mongoose.
5. Implementar los 3 servicios del **Módulo: CONSOLIDACIÓN Y GENERACIÓN DE REPORTES,** para consultar los reportes solicitados utilizando TypeORM.
6. Crear los métodos, utilizar agregación para obtener la información que se guardará en las tablas de los reportes.
7. Verificar que cada vez que se realicé una operación sobre las transacciones (MongoDB) se actualice la información de los reportes automáticamente (MySQL).

8. Agregar validaciones a los servicios para evitar ir a la base de datos innecesariamente y/o para evitar registrar datos incorrectos o que no tienen un formato válido.

ENTREGABLES

Crear un repositorio en github. Este repositorio debe ser privado con permisos de acceso a jorgemparrah.

En el repositorio se debe subir el proyecto con lo solicitado anteriormente.

En u-cursos, se debe entregar el link al repositorio git y el id del commit donde están los cambios que se evaluarán.