Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Curso: Ingeniería de Software 1 (2016701)

Patrones

Nicolás Andrés Castellanos Rico
Juan Felipe Quiroga Medina
Erfán Andrés Triana Duque
Juan David Yopasa Maldonado

Singleton:

Para este parte en los documentos de proyecto se realizó la integración del patron:

database.ts

Este archivo define la configuración y estructura principal para la conexión con la base de datos utilizando TypeORM. En primer lugar, se importan los módulos necesarios y se define un enumerador UserRole que representa los distintos roles de usuario posibles: user, admin y guest. A continuación, se declara la entidad AuthUser, que mapea la tabla auth_user en la base de datos a una clase de TypeScript. Luego, se implementa un patrón Singleton para crear y administrar una única instancia del DataSource de TypeORM, configurado para conectarse a una base de datos MySQL local y sincronizar automáticamente la estructura

de tablas. La instancia resultante se exporta como AppDataSource para ser utilizada en otros archivos TypeScript.

test.ts

Este archivo realiza una prueba que inicia el DataSource con AppDataSource, obtiene el repositorio correspondiente a la entidad AuthUser y crea un nuevo usuario con datos de ejemplo. Luego, guarda este usuario en la base de datos y muestra en la consola tanto la información del nuevo registro como la lista completa de usuarios existentes. Al final, muestra un mensaje que indica si la operación fue exitosa o no.

Observer:

Cuando un usuario crea, edita o borra una transacción, el sistema debe actualizar el saldo, guardar un snapshot histórico, registrar la notificación y avisar al frontend en tiempo real. El controlador solo persiste la transacción, y el ObservadorTransaccion se ocupa de las tareas derivadas.

Actores
Usuario, frontend React, backend Node TypeScript con TypeORM, ObservadorTransaccion, base de datos MySQL, clientes conectados por sockets

Precondiciones
Existe una cuenta en account con account_id correspondiente. La conexión a la base de datos está inicializada en FuenteDatos o AppDataSource, si se usan sockets, io está creado y listo. Se debe verificar si hay triggers o procedimientos en la base de datos que ya realicen snapshots o notificaciones, para evitar duplicados

Trigger
La acción inicial es el guardado de una transacción mediante POST en el endpoint de transacciones de prueba

Flujo

1. El frontend envía la petición con los campos cuentaId, amount, type y description
2. El backend válida y persiste la fila en transaction_tbl, y responde al cliente con el objeto transacción y estado 201
3. TypeORM dispara el hook afterInsert o el hook correspondiente para update o delete
4. El ObservadorTransaccion recibe el evento, toma la cuenta afectada y calcula la suma de transacciones con una consulta agregada sobre transaction_tbl, tambien el ObservadorTransaccion actualiza account.cached_balance con la suma calculada si la columna existe en el esquema.
5. El ObservadorTransaccion inserta un snapshot en balance_history con total_balance y un payload que incluye meta datos como accion y transactionId, y tambien

ObservadorTransaccion crea una fila en notification_log con status pending y payload que describe la operación.

6. ObservadorTransaccion emite en memoria un evento balanceCuentaActualizado con el payload que contiene cuentaId, balance y meta
7. El servidor escucha balanceCuentaActualizado y reemite por sockets para que los clientes conectados actualicen la interfaz sin recargar

Cambios esperados en la base de datos

- transaction_tbl tendrá la nueva fila de la transacción
- account mostrará cached_balance actualizado a la suma calculada si esa columna forma parte del esquema
- balance_history contendrá al menos un snapshot reciente con total_balance igual al nuevo saldo
- notification_log mostrará una entrada reciente con status pending y payload con transactionId y contexto

Pruebas manuales mínimas

- Asegurar cuenta de prueba con account_id igual a 1 en la base de datos.
- Enviar la petición POST al endpoint de pruebas para crear la transacción.
- Esperar brevemente y comprobar las tablas con consultas SQL que consulten cached_balance, balance_history y notification_log.
- Confirmar en logs del servidor que el observador corrió y que se emitió el evento