2.7.1 Experiência prática com Transactions em um produto de crédito

Situação: Em um projeto relacionado a produtos de crédito, estava implementando a camada de serviço responsável por processar diversas etapas de uma operação de crédito. O processo envolvia várias etapas sequenciais, como validação de dados, verificação de crédito, atualização do histórico do cliente, aprovação e emissão de contrato. Era crítico garantir que, caso qualquer uma dessas etapas falhasse, nenhuma mudança fosse persistida no banco de dados.

Tarefa: Minha tarefa era garantir a integridade dos dados ao processar essas etapas de forma sequencial, assegurando que o sistema permanecesse consistente, mesmo em caso de falhas durante o processamento de uma das etapas.

Ação: Para isso, utilizei **transactions** na camada de serviço. Agrupei as operações de validação, verificação e atualização dentro de uma transação. Se, por exemplo, a etapa 3 falhasse (atualização do histórico do cliente), as etapas anteriores (validação e verificação de crédito) seriam revertidas automaticamente, impedindo que o banco de dados ficasse com dados inconsistentes.

Resultado: Essa abordagem garantiu que, em casos de falha, as transações anteriores fossem revertidas, mantendo a integridade dos dados e evitando qualquer inconsistência no sistema. O processo de crédito foi realizado de forma mais segura e robusta.

2.7.2 Experiência prática com Triggers para Auditoria

Situação: Em um projeto de sistema financeiro, a equipe foi solicitada a implementar uma solução para auditar todas as operações realizadas no sistema, desde o login do usuário até todas as operações CRUD realizadas, incluindo chamadas a APIs de terceiros. O cliente exigia a implementação de uma auditoria robusta e sem impacto na performance, além de garantir o rastreamento de todas as modificações e acessos no sistema.

Tarefa: Minha tarefa era projetar e implementar um sistema de auditoria que registrasse as ações dos usuários de forma detalhada e segura, sem comprometer o desempenho do sistema. As auditorias deveriam incluir login de usuários, inserção, atualização, exclusão de registros e também todas as interações com APIs externas.

Ação: Para isso, utilizei **triggers** no banco de dados para capturar todas as operações de **CRUD** e registrar as informações necessárias, como o ID do usuário, a operação realizada, a tabela afetada e a data/hora da ação. Criei triggers para todos os tipos de operações que precisavam ser auditadas e também para registrar as chamadas às APIs de terceiros, incluindo os dados de entrada e saída. Para garantir que o desempenho não fosse impactado, as informações de auditoria eram registradas no ElasticSearch.

Resultado: A implementação das triggers de auditoria permitiu rastrear e monitorar todas as ações críticas do sistema de forma automática e eficiente, atendendo aos requisitos de segurança e conformidade. A solução foi bem-sucedida em fornecer relatórios detalhados e em tempo real, sem afetar significativamente o desempenho do sistema, o que resultou em uma maior confiança e controle sobre as operações do sistema.