#### **Cenário 1: O problema sem o DAO**

**Pergunta 1.1:** Se a equipe de desenvolvimento decidir, no meio do projeto, mudar o banco de dados de MySQL para PostgreSQL, qual seria o impacto no seu código? Que tipo de trabalho seria necessário para realizar essa mudança?

A mudança de banco de dados no meio do projeto pode ter impactos significativos, dependendo de como o código foi estruturado.

Vou separar aqui em dois tópicos importantes de serem analisados, impactos no código e trabalho necessário para a mudança:

#### Impactos no código

* Diferenças de sintaxe SQL: MySQL e PostgreSQL têm variações em tipos de dados, funções nativas, e até mesmo na forma como lidam com joins, subqueries e constraints.
* Drivers e bibliotecas: Será necessário trocar o driver de conexão (por exemplo, mysql-connector para psycopg2 no Python).
* ORMs (Object-Relational Mappers): Se estiver usando um ORM como Sequelize, Hibernate ou SQLAlchemy, a troca pode ser mais suave, mas ainda exigirá ajustes na configuração e testes.
* Scripts de migração: Será necessário migrar os dados existentes, o que pode envolver exportação, transformação e importação com cuidado para preservar integridade e consistência.

#### Trabalho necessário

* Refatorar queries específicas que dependem de funções ou comportamentos exclusivos do MySQL.
* Ajustar configurações de conexão e autenticação.
* Testar intensivamente para garantir que não haja regressões ou falhas silenciosas.
* Reescrever stored procedures, triggers e funções, se existirem.
* Validar performance: índices e planos de execução podem se comportar de forma diferente.

**Pergunta 1.2:** Que outros problemas de manutenção, organização e segurança podem surgir ao misturar a lógica de negócio com a lógica de acesso ao banco de dados?

No meu entendimento diante de pesquisa, vou separar aqui também em dois tópicos importante que são os problemas de manutenção, problemas de organização e segurança conforme abaixo:

#### - Problemas de manutenção

* Dificuldade de testes: Testar a lógica de negócio isoladamente se torna quase impossível.
* Baixa reutilização: O código fica acoplado ao banco, dificultando reaproveitamento em outros contextos.
* Refatorações complexas: Qualquer mudança na estrutura do banco pode exigir alterações em múltiplos pontos do código.

#### - Problemas de organização

* Violação da arquitetura em camadas, acaba quebrando o princípio de separação de responsabilidades.
* Código ficará confuso e difícil de ser interpretado, misturando regras de negócio com SQL torna o código menos legível e mais propenso a erros.

#### - Problemas de segurança

* Maior risco de SQL Injection: Se não houver abstração ou sanitização adequada.
* Exposição indevida de dados: Lógica mal estruturada pode permitir acesso não autorizado a informações sensíveis.
* Dificuldade em aplicar políticas de acesso: Sem uma camada intermediária, fica mais difícil controlar quem pode fazer o quê.

#### **Cenário 2: A solução com o DAO**

Pergunta 2.1: O que é um DAO e o que ele faz?

DAO (Data Access Object) é um padrão de projeto que isola a lógica de acesso ao banco de dados da lógica de negócio da aplicação, em termos simples ele centraliza todas as operações de leitura, escrita, atualização e exclusão de dados permitindo que o restante do sistema não precise saber como os dados são armazenados ou recuperados funcionando como uma ponte entre o banco de dados e o código da aplicação.

### Pergunta 2.2: Como o DAO ajuda na troca de banco de dados?

Ao usar DAOs corretamente, a lógica de acesso ao banco fica concentrada em uma camada única, trazendo grandes vantagens como você não precisará alterar o código da lógica de negócio, apenas os DAOs, as queries específicas de MySQL ficam encapsuladas no DAO precisando apenas adaptá-las para o banco que será migrado, no caso o PostgreSQL e uma das principais vantagens está no desacoplamento facilitando a manutenção, testes, migração e até escalabilidade por a aplicação não depender diretamente do tipo do banco de dados.

### Pergunta 2.3: Exemplos de métodos em um DAO de Produto.

Métodos chamados de CRUD (Create, Read, Update, Delete), que é a base da maioria das operações com dados.

Segue abaixo 5 (cinco) métodos típicos que um DAO para a entidade Produto poderia ter:

|  |  |
| --- | --- |
| **MÉTODO** | **FUNCÃO** |
| Salvar (Produto produto) | Insere um novo produto no banco de dados |
| BuscarPorId (int id) | Retorna um produto específico pelo seu ID |
| ListarTodos () | Retorna uma lista com todos os produtos cadastrados |
| Atualizar (Produto produto) | Atualiza os dados de um produto existente |
| Remover (int id) | Exclui um produto do banco com base no ID |