# Desvendando o Padrão de Projeto DAO (Data Access Object)

Nome: Luis Eduardo de Sousa

Disciplina: Técnico em Desenvolvimento de Sistemas

## Introdução

Esta atividade apresenta o padrão DAO (Data Access Object), explicando sua razão de ser, suas vantagens e como ele organiza o acesso a dados em aplicações Java com bancos relacionais como MySQL ou PostgreSQL. As respostas abaixo foram escritas com linguagem própria e buscam simular cenários reais de desenvolvimento.

## Cenário 1: O problema sem o DAO

### Resposta 1.1:

Se o código de acesso ao banco estiver espalhado por várias classes e a equipe precisar trocar o banco de MySQL para PostgreSQL no meio do projeto, o impacto será grande. Será necessário localizar todas as partes do sistema que contêm código específico do MySQL (strings de conexão, drivers JDBC, SQL específicos do MySQL, tratamentos de tipos) e modificar cada ocorrência. O trabalho envolveria:  
- Substituir o driver JDBC e ajustar as strings de conexão.  
- Revisar consultas SQL que utilizem sintaxes ou funções específicas do MySQL e adaptá‑las ao PostgreSQL.  
- Ajustar mapeamentos de tipos de dados (por exemplo, tipos de data, auto-incremento).  
- Reexecutar e corrigir testes em toda a aplicação para detectar quebras.  
  
Esse esforço é custoso em tempo e risco de introduzir bugs, principalmente em projetos grandes.

### Resposta 1.2:

Misturar lógica de negócio com lógica de acesso a dados gera vários problemas:  
- Manutenção difícil: alterações no acesso a dados exigem mudanças em várias partes do código.  
- Baixa reutilização: trechos de acesso duplicados aumentam o trabalho e o risco de inconsistência.  
- Testabilidade comprometida: testar a lógica de negócio necessita de acesso ao banco real, dificultando testes automatizados e isolados.  
- Segurança enfraquecida: espalhar credenciais e tratamento de consultas (por exemplo, concatenação de strings para montar SQL) pode aumentar o risco de vazamento de senhas e ataques de SQL Injection.  
- Acoplamento alto: mudanças na infraestrutura (banco, driver) impactam diretamente na lógica de negócio.  
- Violação do princípio de responsabilidade única: cada classe faz mais do que deveria, dificultando leitura e evolução do código.

## Cenário 2: A solução com o DAO

### Resposta 2.1:

A principal função do DAO é encapsular todas as operações de acesso a dados para uma entidade, criando uma camada responsável apenas por persistência e recuperação de informações. O DAO atua como uma ponte entre a aplicação (lógica de negócio) e o banco de dados, expondo métodos de alto nível como salvar, atualizar, excluir e consultar objetos, sem que o restante do sistema precise conhecer detalhes do banco ou do mecanismo de acesso.

### Resposta 2.2:

Com o DAO, a mudança de MySQL para PostgreSQL fica muito mais simples. Em vez de procurar e alterar trechos de código dispersos, a equipe só precisa adaptar (ou reimplementar) as classes de DAO que fazem o acesso ao banco. As principais vantagens dessa abordagem são:  
- Isolamento: a lógica de negócio permanece intacta, porque ela chama métodos do DAO, não SQL diretamente.  
- Menor área de mudança: apenas a camada de persistência é alterada (por exemplo, uma nova implementação de DAO).  
- Possibilidade de múltiplas implementações: é comum ter uma interface ProdutoDAO e implementações MySQLProdutoDAO e PostgreProdutoDAO — facilitando testes e migração gradual.  
- Testes mais fáceis: é possível criar DAOs falsos (mocks) ou em memória para testes unitários da lógica de negócio.  
- Segurança centralizada: tratamento de parâmetros, uso de queries parametrizadas/prepared statements e gerenciamento de conexões ficam concentrados e mais fáceis de auditar.

### Resposta 2.3:

Para a entidade 'Produto', o DAO normalmente oferece operações básicas de CRUD e consultas úteis ao negócio. Exemplos de métodos (nomes e finalidade):

- salvar(Produto produto) — Persiste um novo produto no banco.

- atualizar(Produto produto) — Atualiza os dados de um produto existente.

- excluir(int idProduto) — Remove um produto pelo seu identificador.

- buscarPorId(int idProduto) — Retorna o produto correspondente ao id informado.

- listarTodos() — Retorna uma lista com todos os produtos cadastrados.

- buscarPorCategoria(String categoria) — (opcional) Retorna produtos filtrados por categoria.

- buscarPorNome(String nome) — (opcional) Busca produtos cujo nome contenha o termo informado.

## Exemplo conceitual em Java (sem código)

Uma forma comum de estruturar é definir uma interface ProdutoDAO com as assinaturas dos métodos acima. Em seguida, criar implementações concretas, como MySQLProdutoDAO (que usa JDBC ou um framework específico) e PostgreProdutoDAO. A camada de serviço (por exemplo, ProdutoService) injeta/usa a interface ProdutoDAO, assumindo que não importa qual implementação está sendo usada. Isso permite também usar ORMs como JPA/Hibernate no lugar de JDBC puro, mantendo a mesma interface de DAO para o restante do sistema.

## Conclusão

O padrão DAO melhora a organização, reduz o acoplamento entre regras de negócio e persistência, aumenta a segurança e facilita mudanças na infraestrutura de dados. Para quem trabalha com Java e bancos como MySQL ou PostgreSQL, aplicar o DAO desde as fases iniciais do projeto torna a evolução e a manutenção muito mais sustentáveis.