EAD - Polo Santa Luiza – Vitória – ES

DESENVOLVIMENTO FULL STACK

RELATÓRIO DE PRÁTICA
MUNDO 3 – NÍVEL 3 – MISSÃO PRÁTICA
RPG0016 - BackEnd sem banco não tem

CARLOS ALTOMARE CATÃO

Matrícula: 202403460912

Semestre Letivo: 2025/1 - 3º Período

Data: 2025/05

🎯 Objetivos da Prática

Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

Etapas:

1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.
5. No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

📌 Requisitos

- JDK (*Java Development Kit*);
- IDE *NetBeans*;
- Banco de dados SQL Server;
- SQL Server Management Studio (**SSMS**);
- *Driver* **JDBC** (*Java Database Connectivity*) para SQL Server;
- O Banco de Dados gerado na Missão Prática do Nível 2 (*loja*);
- Navegador para Internet, como o Chrome;
- Computador com acesso à Internet.

▶️ Procedimento 1 - Mapeamento Objeto-Relacional e DAO

✅ Objetivos:

O projeto da Missão Prática foi desenvolvido a título de um exercício completo de integração entre uma aplicação Java padrão (modelo Ant) e um banco de dados SQL Server. Ao longo da implementação, foi possível explorar conceitos fundamentais de JDBC, POO (Programação Orientada a Objetos) e manipulação de dados relacionais com persistência em tabelas normalizadas.

📦 Estrutura do Projeto

O projeto foi dividido de forma modular, observando-se a separação de responsabilidades:

- **Modelo de domínio** (Pessoa, PessoaFisica, PessoaJuridica)
- **DAO (Data Access Object)** para encapsular a lógica de acesso aos dados
- **Utilitários** (ConectorBD, SequenceManager) para conexão e controle de sequência
- **Classe de testes** (CadastroBDTeste) para validar todas as operações CRUD

⚙️ Funcionalidades

1. **Conexão com o banco**: A comunicação com o banco de dados SQL Server foi estabelecida por intermédio do *driver* **mssql-jdbc-12.10.0.jre11.jar**.
2. **Criação de objetos de negócio**: As classes de entidade foram estruturadas com herança e polimorfismo.
3. **Operações CRUD**:
    - Inclusão de pessoas físicas e jurídicas (tabelas Pessoa, Pessoa_Fisica, Pessoa_Juridica)
    - Alteração e exclusão dos dados persistidos
    - Listagem geral das pessoas cadastradas
4. **Controle de sequência**: O *SequenceManager* foi utilizado para gerenciar as sequências da *chave primária* no banco de dados SQL Server.

📄 Códigos

Os códigos foram desenvolvidos com a IDE NetBeans e se encontram em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_3_Missao_Pratica/tree/main/Procedimento-1/CadastroBD.

🧪 Testes

Foram explorados testes de inclusão, alteração, exclusão e listagem. Como aprendizado importante há que se destacar o aspecto relativo às permissões necessárias para se efetuar as transações junto ao banco de dados.

- O uso do conn.setAutoCommit(false) com commit() e rollback() permitiu controle transacional, evitando corrupção de dados em falhas intermediárias.
- A listagem no console (via exibir()) confirmou a integridade das operações exploradas nos testes.

📜 Resultados

Os resultados da execução dos códigos se encontram ilustrados no arquivo *Resultados.pdf* que se encontra em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_3_Missao_Pratica/blob/main/Procedimento-1/RESULTADOS.pdf.

⚠️ Desafios Enfrentados

- **Permissões no banco**: Inicialmente, utilizando o usuário *loja* ocorreram erros de permissão para transações como INSERT, UPDATE e DELETE; que impediram a execução correta dos comandos. A solução encontrada consistiu na aplicação de GRANTs apropriados, utilizando a credencial do administrador do banco de dados.
- **Erros de índice em PreparedStatement**: Em alguns métodos incluir, o número de parâmetros definidos não coincidia com os placeholders da query, gerando erros do tipo "*índice fora do intervalo*".
- **Gerenciamento de sequência**: O *SequenceManager* foi de extrema importância para o tratamento das chaves primárias quando da inclusão de dados no banco.

📌 Conclusão

O projeto **CadastroBD** demonstrou como aplicações Java podem interagir de forma robusta com bancos de dados relacionais, aplicando boas práticas de POO e JDBC. A modularização em pacotes, o uso de DAO, o controle de transações e a organização do código tornam o sistema facilmente escalável e compreensível, o que facilita em muito a depuração do código e sua manutenabilidade.

O exercício permitiu consolidar conhecimentos técnicos importantes, como:

- Conexão com SQL Server via JDBC
- Estruturação de aplicações com camadas separadas
- Tratamento de exceções e recursos (fechamento de ResultSet, Statement, Connection)
- Operações CRUD usando SQL nativo

❓ Questionamentos

Qual a importância dos componentes de middleware, como o JDBC?

O **JDBC** é responsável por intermediar a comunicação (*middleware*) entre aplicações Java e bancos de dados relacionais, garantindo eficiência, segurança e padronização no acesso a dados.

Qual a diferença no uso de *Statement* ou *PreparedStatement* para a manipulação de dados?

Statement executa comandos SQL simples, enquanto *PreparedStatement* permite comandos **parametrizados**, oferecendo maior segurança contra injeções de SQL, melhor desempenho em consultas repetidas.

Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO isola a lógica de acesso a dados do restante da aplicação, facilitando futuras manutenções, a possível troca do **SGBD** (Sistema Gerenciador do Banco de Dados) e a organização do código.

Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

No modelo relacional, a herança é geralmente implementada usando tabelas separadas para cada classe, utilizando chaves estrangeiras para representar a relação entre as tabelas.

▶️ Procedimento 2 - Alimentando a Base

✅ Objetivos

O Procedimento 2 teve como objetivo principal desenvolver uma aplicação Java de linha de comando, utilizando o JDBC, para realizar o gerenciamento de dados de pessoas físicas e jurídicas armazenadas em um banco de dados SQL Server. A aplicação foi desenvolvida na IDE NetBeans, utilizando o modelo de projeto **Java Application (Ant)** e estruturada em pacotes organizados por responsabilidades: modelos de dados, utilitários e classes DAO.

🧱 Estrutura do Projeto

A estrutura utilizada para o procedimento 2 é idêntica a que foi apresentada no procedimento 1.

⚙️ Funcionalidades

O sistema foi implementado com um menu interativo que permite as seguintes operações:

- **[1] Incluir** – Cadastro de novas pessoas físicas ou jurídicas.
- **[2] Alterar** – Edição de dados existentes.
- **[3] Excluir** – Remoção de registros com base no ID.
- **[4] Exibir pelo ID** – Consulta de dados específicos.
- **[5] Exibir todos** – Listagem completa de registros.
- **[0] Sair** – Finalização do sistema.

A opção entre Pessoa Física ou Jurídica é selecionada pelo usuário a cada operação, garantindo, assim, flexibilidade e controle. A entrada dos dados ocorre via teclado, com *feedback* sobre as ações realizadas exibido no console.

📄 Códigos

Os códigos foram desenvolvidos com a IDE NetBeans e se encontram em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_3_Missao_Pratica/tree/main/Procedimento-2/CadastroBD.

🛑 Tratamento de Erros

Durante o desenvolvimento, foi aplicada a técnica de tratamento de exceções para capturar falhas em:

- Conexão com o banco de dados.
- Inserção, alteração ou exclusão de registros.
- Leitura de dados inválidos do teclado.

Esse cuidado garante maior segurança e robustez ao sistema, permitindo que ele a execução prossiga mesmo diante de erros pontuais.

🧪 Testes

Os testes foram realizados na IDE NetBeans, validados pelo SQL Server Management Studio e, também, pela aba "*Databases*" do NetBeans, de forma a assegurar que:

- Os dados eram corretamente persistidos nas tabelas.
- As operações refletiam em tempo real no banco.
- A integridade preservada entre as tabelas Pessoa, Pessoa_Fisica e Pessoa_Juridica.
- A sequência utilizada (seq_pessoa) gerava códigos únicos.

📜 Resultados

Os resultados da execução dos códigos se encontram ilustrados no arquivo *Resultados.pdf* que se encontra em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_3_Missao_Pratica/blob/main/Procedimento-2/RESULTADOS.pdf.

⚠ Desafios Enfrentados

Durante o desenvolvimento desta atividade prática observou-se diversos desafios técnicos, entre os quais pode-se destacar:

- A implementação correta da lógica de exclusão em um banco de dados relacional, respeitando as restrições de integridade referencial.
- A garantia de que os dados fossem manipulados de forma segura e validada, especialmente, nas operações de alteração e consulta por código de pessoa.
- O ajuste da estrutura das queries SQL para garantir que os filtros fossem aplicados corretamente.

Essas dificuldades exigiram uma análise criteriosa e cuidadosa da modelagem do banco de dados e reforçaram a importância do planejamento da arquitetura relacional, validações consistentes e uso adequado de transações. Superar esses obstáculos contribuiu significativamente para o aprendizado prático e o amadurecimento técnico no uso do JDBC, SQL Server e boas práticas de programação em Java.

📌 Conclusão

O projeto atendeu satisfatoriamente aos objetivos propostos, proporcionando uma experiência completa de interação com banco de dados via Java e JDBC. A separação por camadas (modelo, DAO, utilitário) contribuiu para um código limpo, reutilizável e de fácil manutenção.

A implementação do modo texto tornou a aplicação mais amigável ao usuário, garantindo uma melhor usabilidade, e desta forma foi ideal para fins didáticos e como base para futuras evoluções.

Esse exercício proporcionou aprendizado prático valioso sobre acesso a banco de dados, estruturação de sistemas e tratamento de dados em aplicações Java.

❓ Questionamentos

Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

 Arquivos são mais simples e usados em projetos pequenos. Os dados são gravados em formatos como texto ou binário.

 Bancos de dados são mais organizados, permitem consultas com SQL, suportam muitos usuários ao mesmo tempo e garantem integridade dos dados.

 Arquivos exigem mais código para buscar e organizar os dados, enquanto bancos de dados já oferecem ferramentas para isso.

Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?

Antes do Java 8, a impressão de valores exigia loops explícitos como:

```
for (Pessoa p : lista) {
    System.out.println(p.getNome());
}
```

Com **expressões lambda**, isso pode ser simplificado:

```
lista.forEach(p -> System.out.println(p.getNome()));
```

 Código mais **conciso**, **legível** e **funcional**.

 Facilita o uso de APIs como Streams para filtragem, ordenação e transformação de dados.

Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?

O método main é o ponto de entrada da aplicação Java, e ele próprio é static. Isso significa que:

- Ele é chamado **sem instanciar a classe**.
- Portanto, qualquer método invocado diretamente por main **também precisa ser static**, pois métodos não estáticos só podem ser acessados através de um objeto (instância da classe).