▶️ 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).