UNIVERSIDADE ESTÁCIO DE SÁ

DESENVOLVIMENTO FULL STACK

MUNDO 3 – NÍVEL 3

RPG0016 - BACKEND SEM BANCO NÃO TEM

ALUNO: ERB LUCIANO MARIZ DE BARROS FILHO

Objetivos da prática:

Tem como objetivo, o desenvolvimento de sistemas de cadastro bem estruturados, incluindo funcionalidades de inclusão, alteração, exclusão e consulta de registros, assim manipulando os dados inseridos no banco de dados relacional, além de aprofundar o uso do *middleware* JDBC e da aplicação eficiente do padrão DAO em aplicativos Java. Compreender e aplicar conceitos de mapeamento objeto-relacional para facilitar a interação entre objetos Java e as tabelas do banco de dados e refletir sobre as melhores práticas de desenvolvimento de software em Java, abordando aspectos como organização do código, tratamento de exceções e boas práticas de programação.

Resultados:

Consegui como resultados da prática a realização do CRUD, ou seja, quatro etapas principais ao lidar com os dados: criação, leitura, atualização e exclusão.

Na criação, consegui inserir dados novos no banco de dados, através de classes específicas, como PessoaFisicaDAO e PessoaJuridicaDAO. Os dados inseridos incluem informações como nome, endereço, telefone, CPF/CNPJ, entre outros.

Na leitura, consegui buscar e mostrar os dados que foram inseridos no banco de dados, realizado por meio dos métodos de leitura das classes PessoaFisicaDAO e PessoaJuridicaDAO, apresentando os dados ao usuário para visualização.

Na atualização, consegui editar os dados que já haviam sido inseridos no banco de dados utilizando um referencial para saber quem estou atualizando, sendo possível realizar a alteração de dados como nome, endereço, telefone, CPF/CNPJ, entre outros.

Na deleção, consegui excluir os dados do banco de dados, através de um referencial, id, por exemplo.

Além disso, também testada a opção de finalização do programa, obtendo os resultados esperados de interrupção do *loop* e encerramento da execução do programa.

A importância dos componentes de *middleware*, como o JDBC:

Os componentes de *middleware*, como o JDBC (Java Database Connectivity), desempenham um papel fundamental na integração de aplicativos Java com bancos de dados relacionais. O JDBC fornece uma interface padronizada para acessar e manipular dados no banco de dados, independentemente do sistema de gerenciamento de banco de dados subjacente. Isso permite que os desenvolvedores escrevam código Java portável que possa ser executado em diferentes ambientes de banco de dados,

sem a necessidade de modificar a lógica de acesso aos dados. Além disso, o JDBC oferece recursos avançados, como suporte a transações, consultas parametrizadas e manipulação eficiente de conjuntos de resultados, contribuindo para o desenvolvimento de aplicativos robustos e escaláveis.

A diferença no uso de *Statement* ou *PreparedStatement* para a manipulação de dados:

A diferença principal entre Statement e PreparedStatement reside na forma como os comandos SQL são tratados e executados. Um objeto Statement é usado para executar comandos SQL estáticos, onde o texto do comando é definido diretamente no código Java. Isso pode levar a vulnerabilidades de segurança, como ataques de injeção de SQL, se os dados de entrada não forem adequadamente validados e sanitizados. Por outro lado, um objeto PreparedStatement é usado para executar comandos SQL parametrizados, onde os parâmetros do comando são definidos como espaços reservados e preenchidos dinamicamente com valores durante a execução. Isso torna o PreparedStatement mais seguro e eficiente, pois os parâmetros são tratados separadamente do comando SQL, evitando ataques de injeção de SQL e permitindo o reuso eficiente de consultas preparadas.

A melhora a manutenibilidade do software com o padrão DAO:

O padrão DAO (*Data Access Object*) melhora a manutenibilidade do software ao promover a separação de responsabilidades entre a lógica de negócios e a lógica de acesso a dados. Ele encapsula a lógica de acesso ao banco de dados em classes específicas, reduzindo o acoplamento entre os componentes do sistema e facilitando a modificação e extensão do código. Com o padrão DAO, é possível alterar a fonte de dados subjacente (por exemplo, de um banco de dados relacional para um serviço *web*) sem afetar outras partes do sistema. Além disso, o padrão DAO facilita a realização de testes unitários e a aplicação de práticas de desenvolvimento ágil, como a refatoração de código e a melhoria contínua.

O reflexo da herança no banco de dados, quando lidamos com um modelo estritamente relacional:

Quando lidamos com um modelo estritamente relacional, a herança pode ser refletida no banco de dados de várias maneiras, dependendo da abordagem escolhida para modelar a hierarquia de classes no sistema. Uma abordagem comum é usar uma tabela para cada classe concreta na hierarquia de herança, onde cada tabela contém os atributos específicos da classe, além dos atributos herdados das classes pai. Isso é conhecido como mapeamento de tabela por classe (ou tabela por subclasse). Outra abordagem é usar uma única tabela para todas as classes na hierarquia de herança, onde uma coluna é usada para identificar o tipo de objeto representado por cada linha. Isso é conhecido como mapeamento de tabela por hierarquia (ou tabela única para hierarquia). Ambas as abordagens têm vantagens e desvantagens, e a escolha entre elas depende dos requisitos específicos do sistema e das preferências do desenvolvedor.