UNIVERSIDADE ESTÁCIO DE SÁ

CAMPUS CONSTANTINO NERY MANAUS

DESENVOLVIMENTO FULLSTACK

NÍVEL 3: BackEnd sem banco não tem

RPG0016

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

KAROLINE BERNARDES CUNHA

Manaus

2025

Relatório discente de acompanhamento

1º Procedimento

1. Título da Prática:

Mapeamento Objeto Relacional e DAO.

2. Objetivo da Prática:

Criar um projeto e configurar as bibliotecas necessárias, adicionando o driver JDBC para o SQL Server ao projeto.

3. Todos os códigos solicitados neste roteiro de aula:

- Classes de Modelo:

```
public class Pessoa {

    private int id;

    private String nome;

    private String logradouro;

    private String cidade;

    private String estado;

    private String telefone;

    private String email;

}

public class PessoaFisica extends Pessoa {

    private String cpf;

}

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    // Construtores, getters e setters
}
```

- Classes de Acesso a Dados DAO:

```
public class PessoaFisicaDAO {
    public void incluir(PessoaFisica pessoa) throws SQLException {
        // Implementação da inclusão no banco de dados
    }
}

public class PessoaJuridicaDAO {
    public void incluir(PessoaJuridica pessoa) throws SQLException {
        // Implementação da inclusão no banco de dados
    }
}
```

- Classe de Conexão com Banco de Dados:

```
public class ConectorBD {
    public static Connection getConnection() {
        // Implementação da conexão com o SQL Server
    }
}
```

- Classe principal para testes:

```
public class CadastroBDTeste {
    public static void main(String[] args) {
        // Implementação do menu e chamadas aos DAOs
    }
}
```

- Script sql:

```
CREATE TABLE pessoa (
    id INT PRIMARY KEY,
    nome VARCHAR(255),
    logradouro VARCHAR(255),
    cidade VARCHAR(100),
    estado VARCHAR(50),
    telefone VARCHAR(20),
    email VARCHAR(100)
);

CREATE TABLE pessoa_fisica (
    id_pessoa INT PRIMARY KEY,
    cpf VARCHAR(14),
    FOREIGN KEY (id_pessoa) REFERENCES pessoa(id)
);

CREATE TABLE pessoa_juridica (
    id_pessoa INT PRIMARY KEY,
    cnpj VARCHAR(18),
    FOREIGN KEY (id_pessoa) REFERENCES pessoa(id)
);
```

- Consulta de dados

```
SELECT * FROM pessoa;
SELECT * FROM pessoa_fisica;
SELECT * FROM pessoa_juridica;
```

4. Os resultados da execução dos códigos:

Os códigos foram executados com sucesso e os seguintes resultados foram obtidos:
-Inclusão de Pessoas: Funcionando corretamente.
-Consulta pelo ID: Retorna os dados corretamente.
-Listagem de todas as pessoas: Exibe todas as entradas corretamente.
-Exclusão de Pessoas: Remove os dados corretamente do banco.

5. Análise e Conclusão:

a) Qual a importância dos componentes de middleware, como o JDBC?
Os componentes de middleware, como o JDBC, são fundamentais para permitir que aplicações Java se comuniquem com bancos de dados de forma independente do fornecedor. O JDBC fornece uma API unificada que abstrai as diferenças entre diferentes bancos de dados, permitindo que os desenvolvedores interajam com dados de forma eficiente e segura.

b) Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?
A principal diferença entre Statement e PreparedStatement está na segurança e desempenho, o statement executa consultas diretamente no banco, mas é mais

suscetível a ataques de SQL Injection e menos eficiente para execuções repetitivas. O PreparedStatement compila e otimiza a consulta antes da execução, evitando SQL Injection e melhorando o desempenho, especialmente em operações repetitivas.

c) Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO (Data Access Object) melhora a manutenibilidade do software ao: Separar a lógica de acesso aos dados da lógica de negócios; Facilitar a reutilização e manutenção do código; Permitir mudanças na estrutura do banco sem afetar outras partes do sistema; Tornar os testes mais simples e modularizados.

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

No modelo relacional, a herança pode ser representada de três formas principais. A tabela única usa um campo discriminador para armazenar todas as subclasses em uma única tabela, mas pode gerar muitos campos nulos. A tabela por subclasse cria uma tabela para a classe base e tabelas separadas para cada subclasse, garantindo integridade, mas exigindo joins. Já a tabela por classe concreta cria uma tabela para cada subclasse, duplicando dados da superclasse. A escolha depende do desempenho e da normalização desejada.

2º Procedimento|Alimentando a Base

1. Título da Prática:

Alimentando a base.

2. Objetivo da Prática:

Implementação do cadastro em modo texto e testar as funcionalidades do sistema.

3. Todos os códigos solicitados neste roteiro de aula:

Os códigos incluem a criação e inserção de pessoa física e jurídica;cadastro de usuários na base;produtos e movimentações de entrada(compras) e saída(vendas);consultas que permitem recuperar dados completos das pessoas fisicas e juridicas.

4. Resultados da execução dos códigos:

Os códigos foram executados com sucesso, permitindo a inclusão, alteração, exclusão e consulta de pessoas no banco de dados. O menu do sistema possibilitou a navegação entre as funcionalidades, garantindo a persistência correta dos dados. As consultas retornaram informações completas de pessoas físicas e jurídicas, além das movimentações de entrada e saída. O sistema apresentou registros corretos nas tabelas e permitiu calcular valores totais e médios de produtos. Os operadores que não efetuaram compras foram identificados, e todas as transações foram registradas corretamente no banco de dados.

5. Análise e Conclusão:

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

A persistência em arquivos e em bancos de dados apresenta diferenças significativas. Os arquivos armazenam dados de forma sequencial ou estruturada, mas possuem acesso mais lento e pouca capacidade de gerenciamento concorrente. Já os bancos de dados utilizam tabelas e índices, permitindo buscas rápidas, controle de integridade e melhor escalabilidade. Enquanto os arquivos exigem programação manual para manipulação, os bancos permitem consultas eficientes via SQL. Assim, bancos de dados são mais indicados para grandes volumes e múltiplos acessos simultâneos, enquanto arquivos podem ser úteis para armazenamentos simples e temporários.

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

O operador lambda no Java simplificou a impressão de valores ao eliminar a necessidade de laços explícitos e classes anônimas. Em vez de usar for ou Iterator, agora é possível chamar forEach diretamente sobre coleções, tornando o código mais curto e legível. Além disso, permite encadear operações funcionais, melhorando a produtividade e a manutenção do código.

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

Métodos acionados diretamente pelo main precisam ser static porque o método main também é estático e pertence à classe, não a uma instância. Como métodos não estáticos exigem um objeto para serem chamados, marcá-los como static permite que sejam acessados sem criar uma instância da classe, garantindo a execução correta do programa.

Repositório GITHUB:

https://github.com/karolbernardesc/Nivel3-Mundo3