UNIVERSIDADE ESTÁCIO DE SÁ POLO CENTRO – IGREJINHA – RS

Curso: Desenvolvimento Full Stack

Disciplina: BackEnd sem banco não tem (RPG0016)

Turma: 2024.1 | 3º semestre

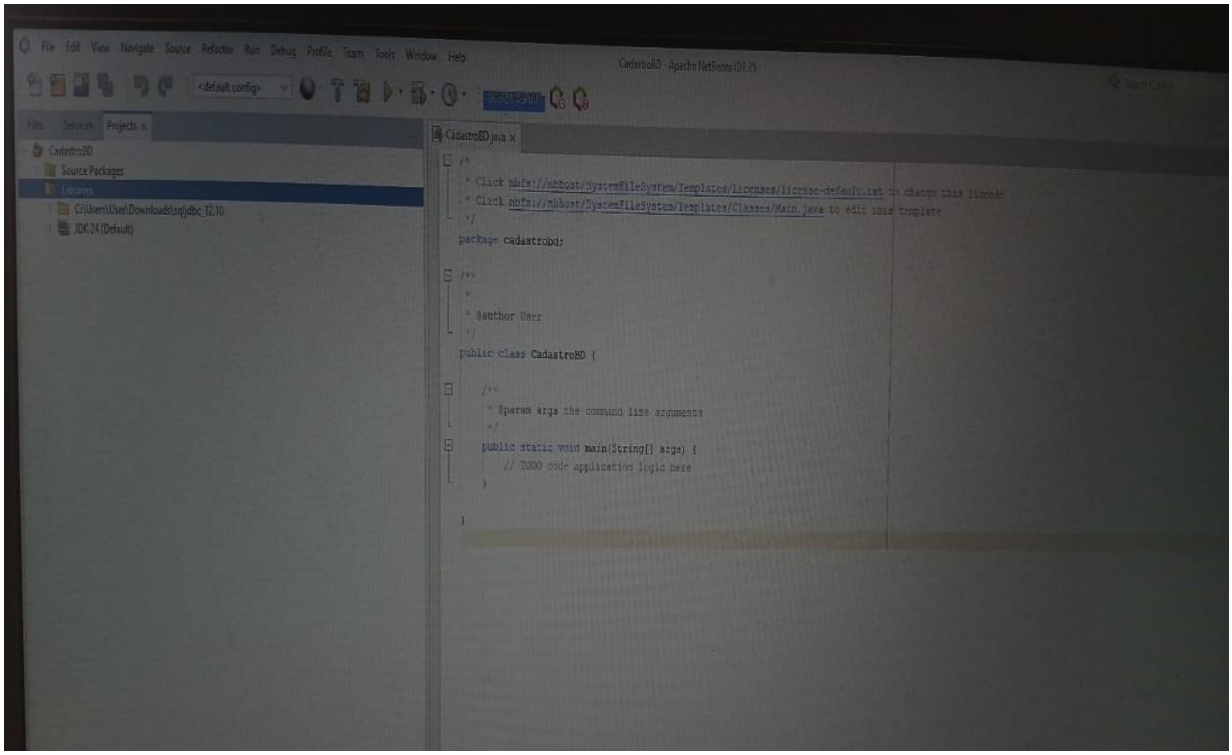
Nome: Emily Duarte Watthier

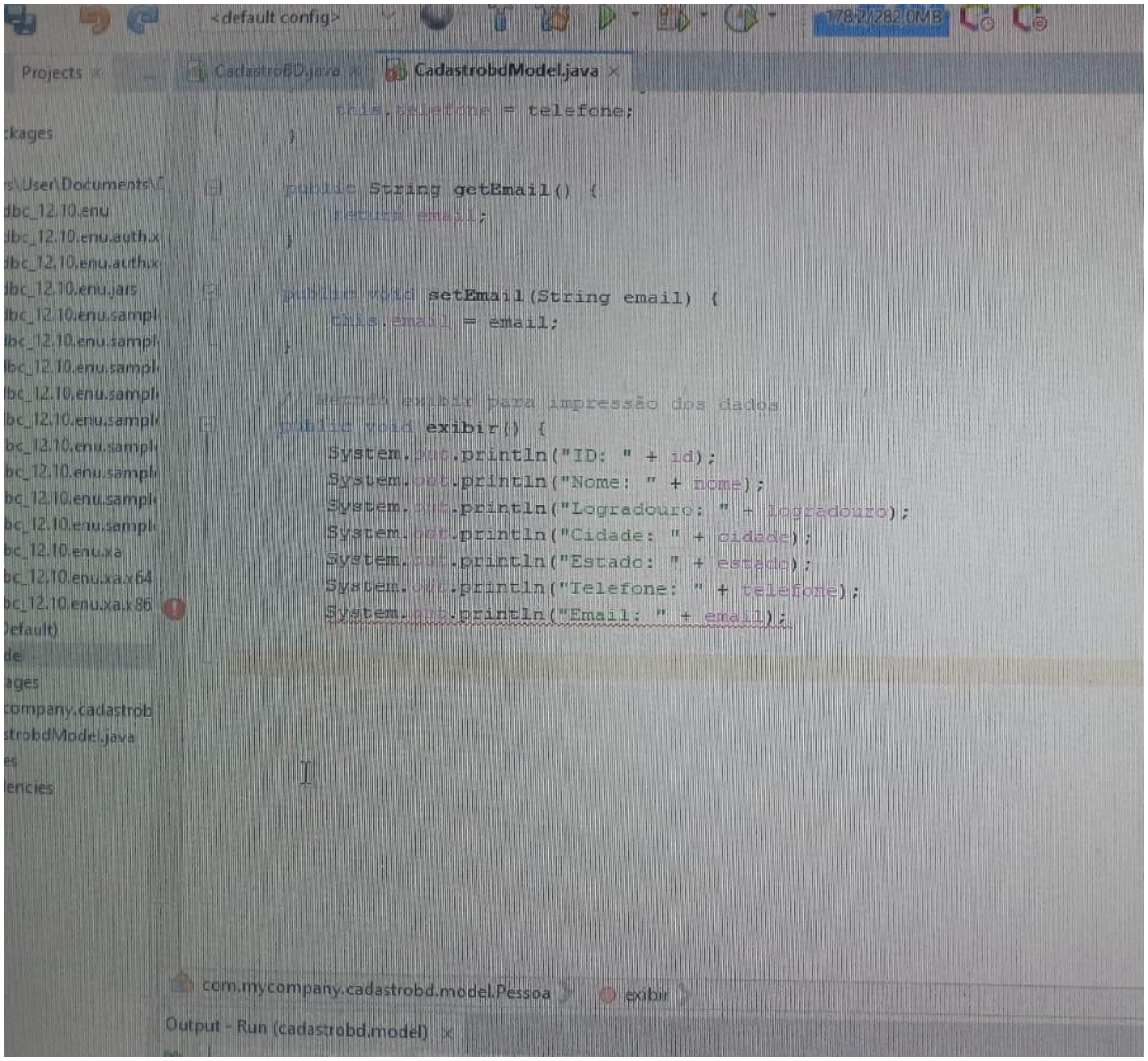
Repositório do GitHub: https://github.com/Watthier09/RPG0016---BackEnd-sem-banco-n-o-tem/blob/main/CadastroBDTeste.java

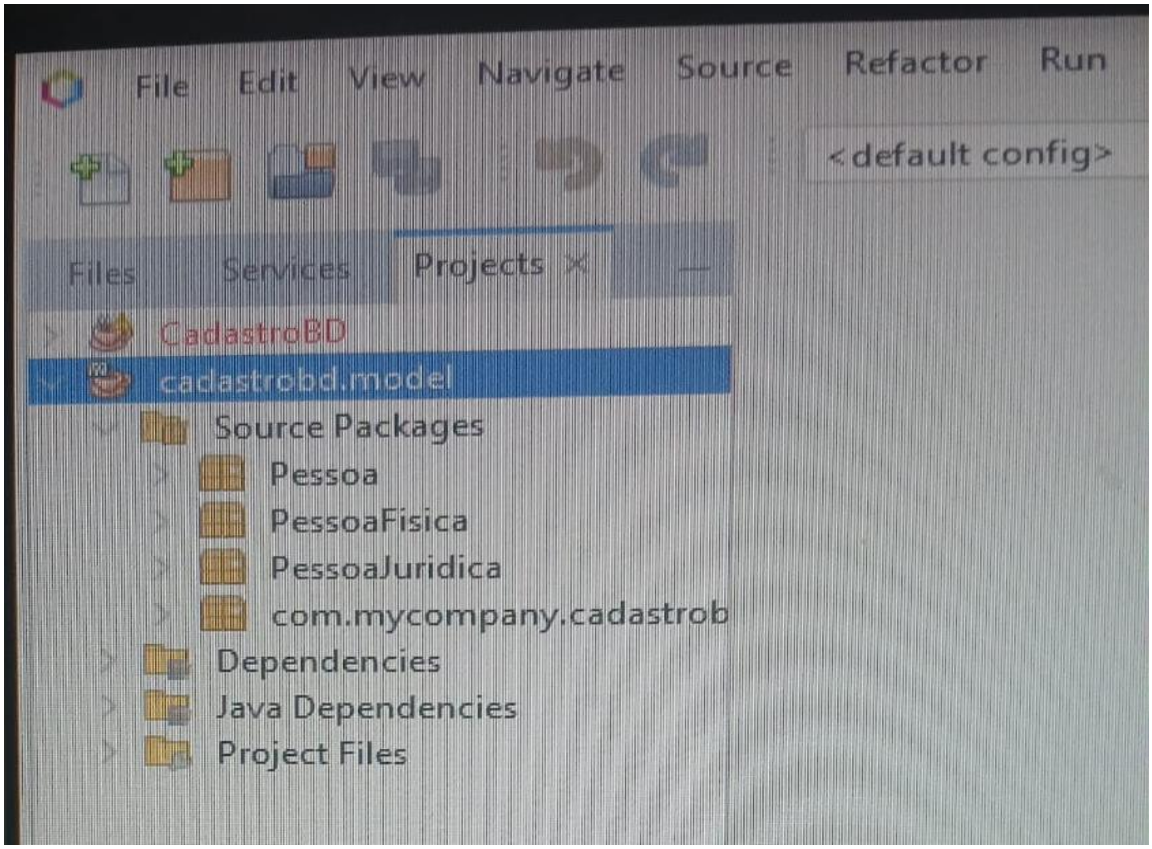
1. **Título da prática:**
   RPG0016 - BackEnd sem banco não tem
   Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

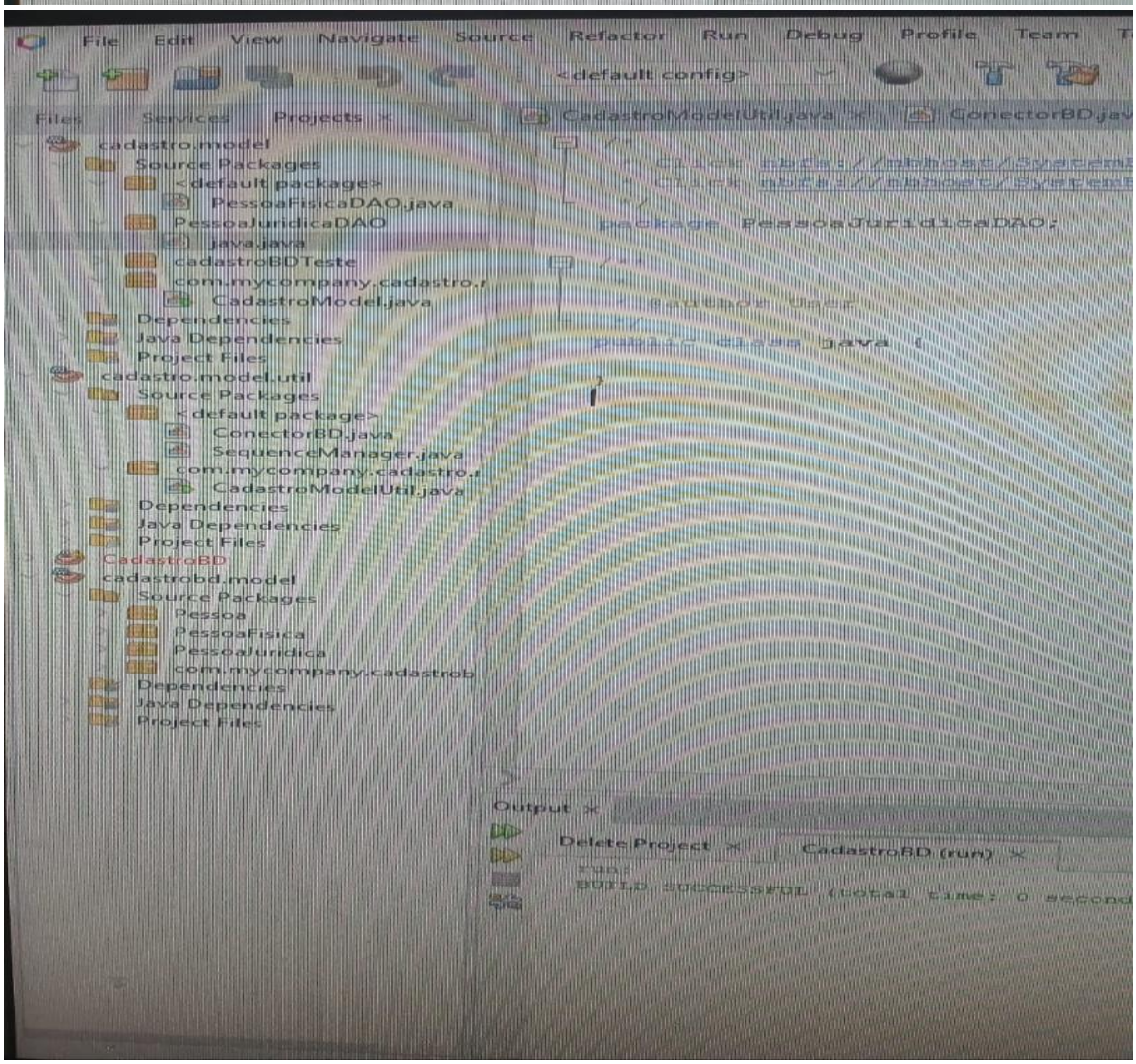
2. **Objetivo da prática:**
   a. Implementar persistência com base no middleware JDBC.
   b. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
   c. Implementar o mapeamento objeto-relacional em sistemas Java.
   d. Criar sistemas cadastrais com persistência em banco relacional.
   e. No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

3. **Códigos:**


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
CadastroBD - Apache NetBeans IDE 25
Projects
CadastroBD
Source Packages
Libraries
JDK 24 (Default)
CadastroBD.java
package cadastrobd;
/**
* @author User
*/
public class CadastroBD {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// TODO code application logic here
}
}


<default config>
Projects
CadastroBD.java
CadastrobdModel.java
this.telefone = telefone;
}
public String getEmail() {
return email;
}
public void setEmail(String email) {
this.email = email;
}
// Método exibir para impressão dos dados
public void exibir() {
System.out.println("ID: " + id);
System.out.println("Nome: " + nome);
System.out.println("Logradouro: " + logradouro);
System.out.println("Cidade: " + cidade);
System.out.println("Estado: " + estado);
System.out.println("Telefone: " + telefone);
System.out.println("Email: " + email);
com.mycompany.cadastrobd.model.Pessoa
exibir
Output - Run (cadastrobd.model)


File Edit View Navigate Source Refactor Run
<default config>
Files Services Projects
CadastroBD
cadastrobd.model
Source Packages
Pessoa
PessoaFisica
PessoaJuridica
com.mycompany.cadastrob
Dependencies
Java Dependencies
Project Files


File Edit View Navigate Source Refactor Run Debug Profile Team
<default config>
Files Services Projects
cadastro.model
Source Packages
<default package>
PessoaFisicaDAO.java
PessoaJuridicaDAO
java.java
cadastroBDTeste
com.mycompany.cadastro.r
CadastroModel.java
Dependencies
Java Dependencies
Project Files
cadastro.modelUtil
Source Packages
<default package>
ConectorBD.java
SequenceManager.java
com.mycompany.cadastro.r
CadastroModelUtil.java
Dependencies
Java Dependencies
Project Files
CadastroBD
cadastrobd.model
Source Packages
Pessoa
PessoaFisica
PessoaJuridica
com.mycompany.cadastrob
Dependencies
Java Dependencies
Project Files
CadastroModelUtil.java
ConectorBD.java
Output
Delete Project
CadastroBD (run)

/*

- Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
- Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

*/ package cadastroBDTeste;

/** *

- @author User

*/ public class CadastroBDTeste {

} package cadastro.model;

import java.util.List; public class CadastroBDTeste { public static void main(String[] args) {

```
    // Testando operações com PessoaFisica
    System.out.println("### TESTE - PESSOA FÍSICA ###");

    // Instanciando e persistindo uma nova Pessoa Física
    PessoaFisica pessoaFisica = new PessoaFisica("João Silva", "Rua A", "Cidade A", "Estado A",
"999999999", "joao.silva@email.com", "12345678900");
    PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
    pessoaFisicaDAO.incluir(pessoaFisica);
    System.out.println("Pessoa Física incluída com sucesso!");

    // Alterando os dados da Pessoa Física
    pessoaFisica.setNome("João Silva Alterado");
    pessoaFisica.setTelefone("888888888");
    pessoaFisicaDAO.alterar(pessoaFisica);
    System.out.println("Pessoa Física alterada com sucesso!");

    // Consultando todas as pessoas físicas
    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
    System.out.println("### Lista de Pessoas Físicas: ###");
    for (PessoaFisica pf : pessoasFisicas) {
        System.out.println(pf);
    }

    // Excluindo a pessoa física
    pessoaFisicaDAO.excluir(pessoaFisica.getId());
    System.out.println("Pessoa Física excluída com sucesso!");



    // Testando operações com PessoaJuridica
    System.out.println("\n### TESTE - PESSOA JURÍDICA ###");

    // Instanciando e persistindo uma nova Pessoa Jurídica
    PessoaJuridica pessoaJuridica = new PessoaJuridica("Empresa ABC", "Av. B", "Cidade B", "Estado B",
"777777777", "empresa.abc@email.com", "12345678000199");
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
    pessoaJuridicaDAO.incluir(pessoaJuridica);
    System.out.println("Pessoa Jurídica incluída com sucesso!");

    // Alterando os dados da Pessoa Jurídica
    pessoaJuridica.setNome("Empresa ABC Alterada");
    pessoaJuridica.setTelefone("666666666");
    pessoaJuridicaDAO.alterar(pessoaJuridica);
    System.out.println("Pessoa Jurídica alterada com sucesso!");

    // Consultando todas as pessoas jurídicas
```

```
    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
    System.out.println("### Lista de Pessoas Jurídicas: ###");
    for (PessoaJuridica pj : pessoasJuridicas) {
        System.out.println(pj);
    }

    // Excluindo a pessoa jurídica
    pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
    System.out.println("Pessoa Jurídica excluída com sucesso!");
}
```

} package cadastro.model;

import java.util.List;

public class CadastroBDTeste {

```
public static void main(String[] args) {
    // Testando operações com PessoaFisica
    System.out.println("### TESTE - PESSOA FÍSICA ###");

    // Instanciando e persistindo uma nova Pessoa Física
    PessoaFisica pessoaFisica = new PessoaFisica("João Silva", "Rua A", "Cidade A", "Estado A",
"999999999", "joao.silva@email.com", "12345678900");
    PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
    pessoaFisicaDAO.incluir(pessoaFisica);
    System.out.println("Pessoa Física incluída com sucesso!");

    // Alterando os dados da Pessoa Física
    pessoaFisica.setNome("João Silva Alterado");
    pessoaFisica.setTelefone("888888888");
    pessoaFisicaDAO.alterar(pessoaFisica);
    System.out.println("Pessoa Física alterada com sucesso!");

    // Consultando todas as pessoas físicas
    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
    System.out.println("### Lista de Pessoas Físicas: ###");
    for (PessoaFisica pf : pessoasFisicas) {
        System.out.println(pf);
    }

    // Excluindo a pessoa física
    pessoaFisicaDAO.excluir(pessoaFisica.getId());
    System.out.println("Pessoa Física excluída com sucesso!");



    // Testando operações com PessoaJuridica
    System.out.println("\n### TESTE - PESSOA JURÍDICA ###");

    // Instanciando e persistindo uma nova Pessoa Jurídica
    PessoaJuridica pessoaJuridica = new PessoaJuridica("Empresa ABC", "Av. B", "Cidade B", "Estado B",
"777777777", "empresa.abc@email.com", "12345678000199");
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
    pessoaJuridicaDAO.incluir(pessoaJuridica);
    System.out.println("Pessoa Jurídica incluída com sucesso!");

    // Alterando os dados da Pessoa Jurídica
    pessoaJuridica.setNome("Empresa ABC Alterada");
```

```
    pessoaJuridica.setTelefone("666666666");
    pessoaJuridicaDAO.alterar(pessoaJuridica);
    System.out.println("Pessoa Jurídica alterada com sucesso!");

    // Consultando todas as pessoas jurídicas
    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
    System.out.println("### Lista de Pessoas Jurídicas: ###");
    for (PessoaJuridica pj : pessoasJuridicas) {
        System.out.println(pj);
    }

    // Excluindo a pessoa jurídica
    pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
    System.out.println("Pessoa Jurídica excluída com sucesso!");
}


}
```

Os demais códigos e arquivos estão no repositório do GitHub, que está listado no começo desse documento e anexado a entrega da atividade também.

5. Análise e conclusão:

a. Importância dos componentes de middleware, como o JDBC?

Componentes de middleware, como o JDBC (Java Database Connectivity), desempenham um papel crucial na arquitetura de aplicações que interagem com bancos de dados. Eles atuam como uma camada de abstração, isolando a aplicação da complexidade específica de cada sistema de gerenciamento de banco de dados (SGBD). Isso traz diversas vantagens, como:

Portabilidade, Reutilização de código,Facilidade de manutenção, Segurança.

b. Diferença entre Statement e PreparedStatement.

Tanto Statement quanto PreparedStatement são interfaces do JDBC usadas para executar consultas SQL, mas possuem diferenças importantes:

Statement: É usado para executar consultas SQL simples, que são compiladas e executadas a cada vez que a consulta é enviada ao banco de

dados. Isso é ineficiente para consultas que são executadas repetidamente com diferentes valores, pois a compilação é feita a cada execução.

PreparedStatement: Permite preparar uma consulta SQL antecipadamente, com marcadores de parâmetros (?). Os valores dos parâmetros são fornecidos posteriormente. A consulta é compilada apenas uma vez, e a execução é muito mais rápida, especialmente para consultas com muitos parâmetros ou que são executadas repetidamente.

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

O padrão DAO encapsula a lógica de acesso a dados em uma camada separada da lógica de negócio da aplicação. Isso melhora a manutenibilidade porque:

Modularidade: O código que interage com o banco de dados é separado do código que executa a lógica de negócio, facilitando a compreensão e a manutenção de cada parte.

Testes: É mais fácil testar a camada DAO separadamente, sem a necessidade de um banco de dados real para testes unitários (mocks).

Reutilização: A camada DAO pode ser reutilizada em diferentes partes da aplicação ou em outros projetos.Abstração: O código da aplicação não precisa se preocupar com os detalhes de implementação do acesso a dados, apenas com a interface do DAO.

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

Em um modelo de banco de dados estritamente relacional, a herança não é diretamente suportada como em linguagens de programação orientadas a objetos. A herança é simulada usando técnicas como:

Tabelas com coluna de tipo: Uma tabela principal contém atributos comuns a todas as subclasses, e uma coluna indica o tipo específico da

subclasse. Atributos específicos de cada subclasse são armazenados em tabelas separadas, relacionadas à tabela principal.

Herança por inclusão (Table per Hierarchy): Todos os atributos de todas as classes são armazenados em uma única tabela. Colunas adicionais representam os atributos específicos de cada subclasse, podendo conter valores NULL para atributos não aplicáveis a uma determinada instância.

Herança por tabela separada (Table per Concrete Class): Cada classe e subclasse possui sua própria tabela. As tabelas são relacionadas através de chaves estrangeiras, representando a relação de herança.