BackEnd sem banco não tem

Jonathan da Silva Araujo - 202205178111

Campus Nova Iguaçu – RJ
BackEnd sem banco não tem (RPG0016) – 9001 – 2023.2 (3º Semestre)

Objetivo da Prática

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.

Procedimentos Realizados

Classe Pessoa

```java
package model;
import java.io.Serializable;

2 usages  2 inheritors  Jonathan Araujo *
public class Pessoa implements Serializable{
    2 usages
    int id;
    2 usages
    String nome;
    2 usages
    String logradouro;
    2 usages
    String cidade;
    2 usages
    String estado;
    2 usages
    String telefone;
    2 usages
    String email;

    no usages  Jonathan Araujo
    public int getId() {
        return id;
    }

    no usages  Jonathan Araujo
    public void setId(int id) {
        this.id = id;
    }
```

```
21      public String getNome() {
22          return nome;
23      }
24
        no usages  new *
25      public void setNome(String nome) {
26          this.nome = nome;
27      }
28
        no usages  new *
29      public String getLogradouro() {
30          return logradouro;
31      }
        no usages  new *
32      public void setLogradouro(String logradouro) {this.logradouro = logradouro;}
33
        no usages  new *
34      public String getCidade() {return cidade;}
        no usages  new *
35      public void setCidade(String cidade) {this.cidade = cidade;}
36
        no usages  new *
37      public String getEstado() {return estado;}
        no usages  new *
38      public void setEstado(String estado) {this.estado = estado;}
39
        no usages  new *
40      public String getTelefone() {return telefone;}
        no usages  new *
41      public void setTelefone(String telefone) {this.telefone = telefone;}
43          public String getEmail() {return email;}
            no usages  new *
44          public void setEmail(String email) {this.email = email;}
45      }
```

Classe PessoaFisica

```
1   package model;
2   import java.io.Serializable;
3
    no usages  Jonathan Araujo *
4   public class PessoaFisica extends Pessoa implements Serializable{
        2 usages
5       long cpf;
6
        no usages  Jonathan Araujo
7       public long getCpf() { return cpf; }
10
        no usages  Jonathan Araujo
11      public void setCpf(long cpf) { this.cpf = cpf; }
14  }
15
```

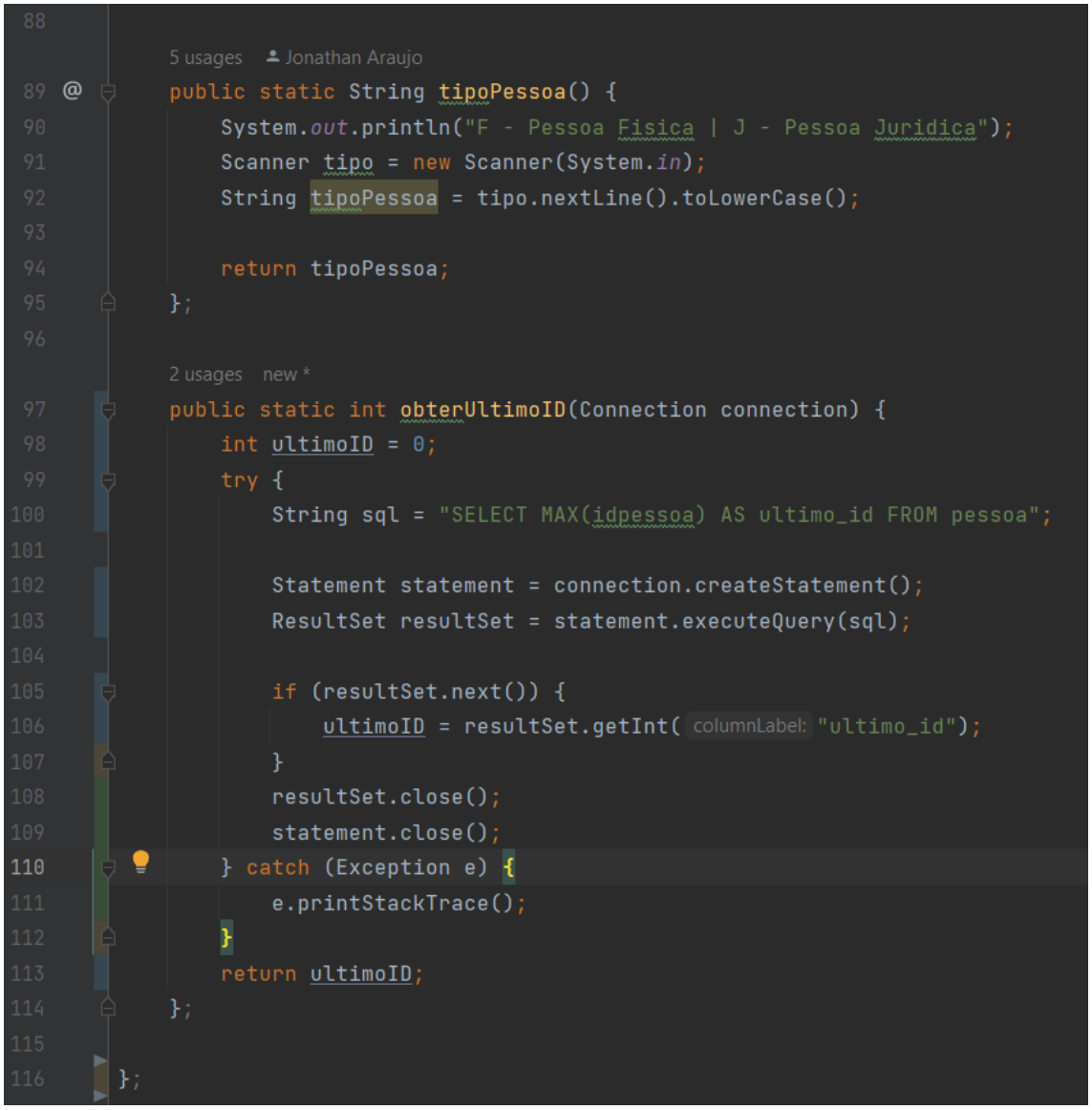
Classe PessoaJuridica

```java
package model;
import java.io.Serializable;

no usages  Jonathan Araujo *
public class PessoaJuridica extends Pessoa implements Serializable{
    2 usages
    long cnpj;

    no usages  Jonathan Araujo
    public long getCnpj() { return cnpj; }

    no usages  Jonathan Araujo
    public void setCnpj(long cnpj) { this.cnpj = cnpj; }
}
```

Classe Metodos

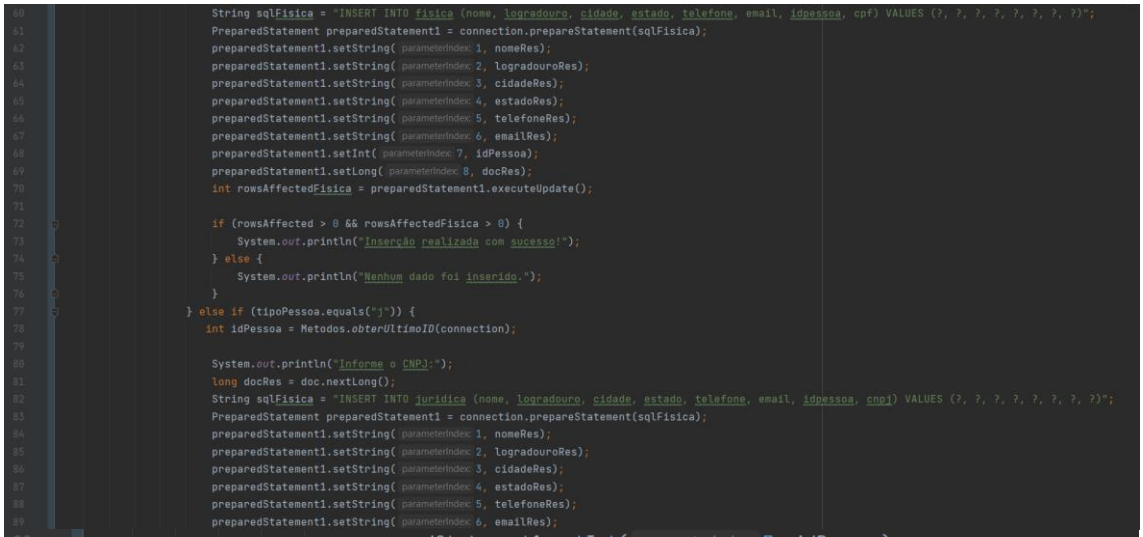
```java
package model;

import java.sql.*;
import java.util.Scanner;

11 usages  Jonathan Araujo *
public class Metodos {
    2 usages  new *
    public static void restauraObjeto(String tipoPessoa, Connection connection) throws SQLException {
        if (tipoPessoa == "pf") {
            try {
                String sql = "SELECT * FROM fisica";
                PreparedStatement preparedStatement = connection.prepareStatement(sql);
                ResultSet resultSet = preparedStatement.executeQuery();
                while (resultSet.next()) {
                    int id = resultSet.getInt( columnLabel: "idpessoa");
                    String nome = resultSet.getString( columnLabel: "nome");
                    String logradouro = resultSet.getString( columnLabel: "logradouro");
                    String cidade = resultSet.getString( columnLabel: "cidade");
                    String estado = resultSet.getString( columnLabel: "estado");
                    String telefone = resultSet.getString( columnLabel: "telefone");
                    String email = resultSet.getString( columnLabel: "email");
                    String cpf = resultSet.getString( columnLabel: "cpf");

                    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);
```

```
                System.out.println("CPF: " + cpf);
                System.out.println("########################################");
            }
            resultSet.close();
            preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    } else if (tipoPessoa == "pj") {
        try {
            String sql = "SELECT * FROM juridica";
            PreparedStatement preparedStatement = connection.prepareStatement(sql);
            ResultSet resultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                int id = resultSet.getInt( columnLabel: "idpessoa");
                String nome = resultSet.getString( columnLabel: "nome");
                String logradouro = resultSet.getString( columnLabel: "logradouro");
                String cidade = resultSet.getString( columnLabel: "cidade");
                String estado = resultSet.getString( columnLabel: "estado");
                String telefone = resultSet.getString( columnLabel: "telefone");
                String email = resultSet.getString( columnLabel: "email");
                String cnpj = resultSet.getString( columnLabel: "cnpj");

                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);
                System.out.println("CNPJ: " + cnpj);
                System.out.println("########################################");
            }

            resultSet.close();
            preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
};

1 usage  Jonathan Araujo *
public static int opcoesTerminal() {
    System.out.println("""
            ================================================
            1 - Incluir Pessoa
            2 - Alterar Pessoa
            3 - Excluir Pessoa
            4 - Buscar pelo Id
            5 - Exibir Todos
            0 - Finalizar Programa
            ================================================
            """);

    Scanner input = new Scanner(System.in);
    int opcao = input.nextInt();
    return opcao;
};
```


5 usages Jonathan Araujo
public static String tipoPessoa() {
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
Scanner tipo = new Scanner(System.in);
String tipoPessoa = tipo.nextLine().toLowerCase();
return tipoPessoa;
};
2 usages new *
public static int obterUltimoID(Connection connection) {
int ultimoID = 0;
try {
String sql = "SELECT MAX(idpessoa) AS ultimo_id FROM pessoa";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(sql);
if (resultSet.next()) {
ultimoID = resultSet.getInt(columnLabel: "ultimo_id");
}
resultSet.close();
statement.close();
} catch (Exception e) {
e.printStackTrace();
}
return ultimoID;
};
};

Classe Main

```java
import model.Metodos;
import java.sql.*;
import java.util.Scanner;

Jonathan Araujo *
public class Main {
    Jonathan Araujo *
    public static void main(String[] args) {
        String jdbcUrl = "jdbc:postgresql://localhost:5432/postgres";
        String username = "loja";
        String password = "loja";

        try {
        Connection connection = DriverManager.getConnection(jdbcUrl, username, password);
        System.out.println("Banco de dados conectado com sucesso!");

        int opcao = -1;

            while (opcao != 0) {
                opcao = Metodos.opcoesTerminal();
                if (opcao == 1) {
                    String tipoPessoa = Metodos.tipoPessoa();

                    Scanner nome = new Scanner(System.in);
                    Scanner logradouro = new Scanner(System.in);
                    Scanner cidade = new Scanner(System.in);
                    Scanner estado = new Scanner(System.in);
                    Scanner telefone = new Scanner(System.in);
                    Scanner email = new Scanner(System.in);
                    Scanner doc = new Scanner(System.in);

                    System.out.println("Informe o nome:");
                    String nomeRes = nome.nextLine();
                    System.out.println("Informe o logradouro:");
                    String logradouroRes = logradouro.nextLine();
                    System.out.println("Informe a cidade:");
                    String cidadeRes = cidade.nextLine();
                    System.out.println("Informe o estado:");
                    String estadoRes = estado.nextLine();
                    System.out.println("Informe o telefone:");
                    String telefoneRes = telefone.nextLine();
                    System.out.println("Informe o email:");
                    String emailRes = email.nextLine();

                    String sql = "INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)";

                    PreparedStatement preparedStatement = connection.prepareStatement(sql);
                    preparedStatement.setString( parameterIndex: 1, nomeRes);
                    preparedStatement.setString( parameterIndex: 2, logradouroRes);
                    preparedStatement.setString( parameterIndex: 3, cidadeRes);
                    preparedStatement.setString( parameterIndex: 4, estadoRes);
                    preparedStatement.setString( parameterIndex: 5, telefoneRes);
                    preparedStatement.setString( parameterIndex: 6, emailRes);
                    int rowsAffected = preparedStatement.executeUpdate();

                    if (tipoPessoa.equals("f")) {

                        int idPessoa = Metodos.obterUltimoID(connection);

                        System.out.println("Informe o CPF:");
                        long docRes = doc.nextLong();
```

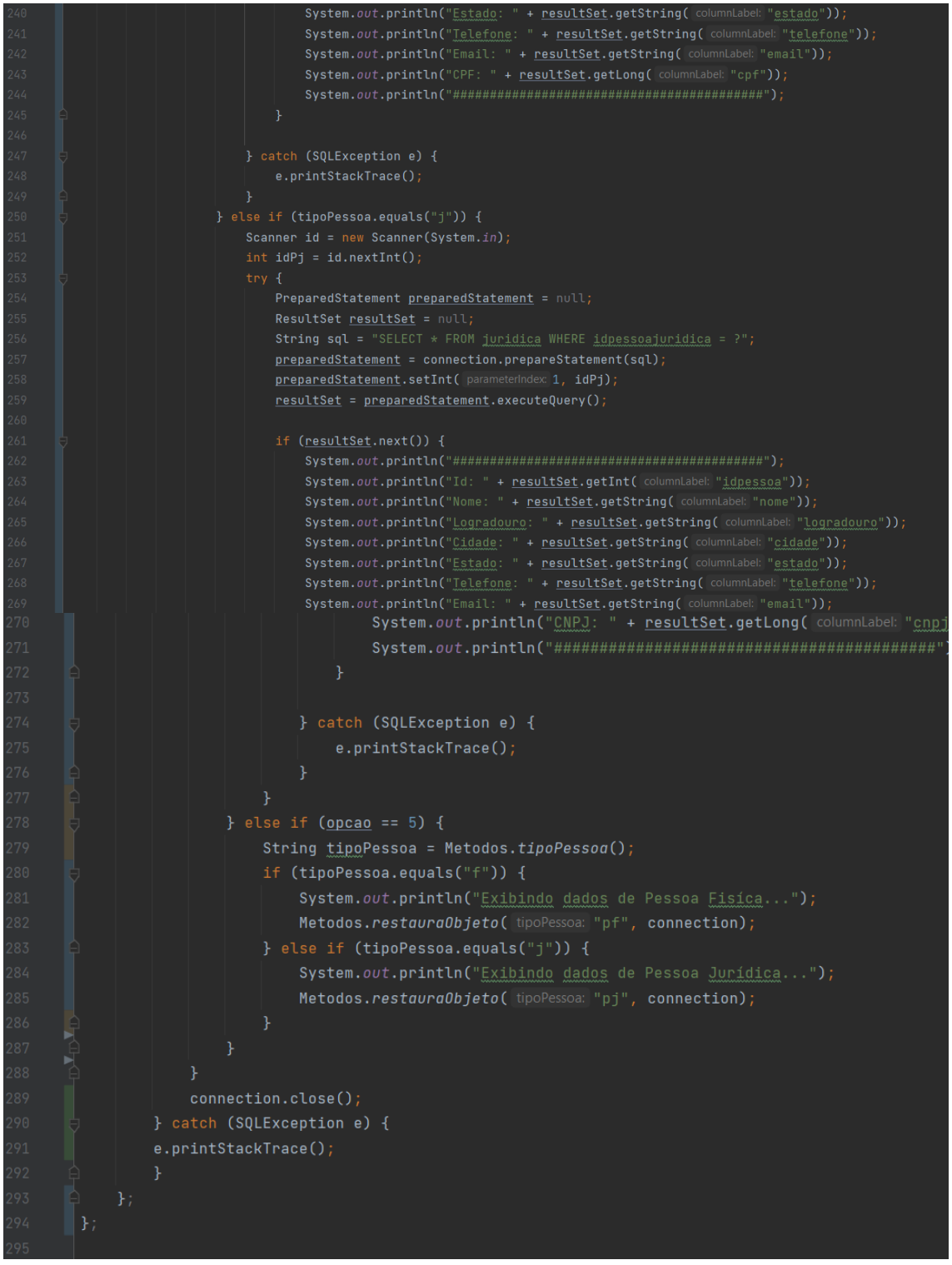


```
                String sqlFisica = "INSERT INTO fisica (nome, logradouro, cidade, estado, telefone, email, idpessoa, cpf) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement preparedStatement1 = connection.prepareStatement(sqlFisica);
                preparedStatement1.setString( parameterIndex: 1, nomeRes);
                preparedStatement1.setString( parameterIndex: 2, logradouroRes);
                preparedStatement1.setString( parameterIndex: 3, cidadeRes);
                preparedStatement1.setString( parameterIndex: 4, estadoRes);
                preparedStatement1.setString( parameterIndex: 5, telefoneRes);
                preparedStatement1.setString( parameterIndex: 6, emailRes);
                preparedStatement1.setInt( parameterIndex: 7, idPessoa);
                preparedStatement1.setLong( parameterIndex: 8, docRes);
                int rowsAffectedFisica = preparedStatement1.executeUpdate();

                if (rowsAffected > 0 && rowsAffectedFisica > 0) {
                    System.out.println("Inserção realizada com sucesso!");
                } else {
                    System.out.println("Nenhum dado foi inserido.");
                }
            } else if (tipoPessoa.equals("j")) {
              int idPessoa = Metodos.obterUltimoID(connection);

                System.out.println("Informe o CNPJ:");
                long docRes = doc.nextLong();
                String sqlFisica = "INSERT INTO juridica (nome, logradouro, cidade, estado, telefone, email, idpessoa, cnpj) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement preparedStatement1 = connection.prepareStatement(sqlFisica);
                preparedStatement1.setString( parameterIndex: 1, nomeRes);
                preparedStatement1.setString( parameterIndex: 2, logradouroRes);
                preparedStatement1.setString( parameterIndex: 3, cidadeRes);
                preparedStatement1.setString( parameterIndex: 4, estadoRes);
                preparedStatement1.setString( parameterIndex: 5, telefoneRes);
                preparedStatement1.setString( parameterIndex: 6, emailRes);
```

```
                    preparedStatement1.setInt( parameterIndex: 7, idPessoa);
                    preparedStatement1.setLong( parameterIndex: 8, docRes);
                    int rowsAffectedJuridica = preparedStatement1.executeUpdate();

                    if (rowsAffected > 0 && rowsAffectedJuridica > 0) {
                        System.out.println("Inserção realizada com sucesso!");
                    } else {
                        System.out.println("Nenhum dado foi inserido.");
                    }
                }
            } else if (opcao == 2) {
                String tipoPessoa = Metodos.tipoPessoa();

                Scanner nome = new Scanner(System.in);
                Scanner logradouro = new Scanner(System.in);
                Scanner cidade = new Scanner(System.in);
                Scanner estado = new Scanner(System.in);
                Scanner telefone = new Scanner(System.in);
                Scanner email = new Scanner(System.in);
                Scanner doc = new Scanner(System.in);
                Scanner id = new Scanner(System.in);

                System.out.println("Insira o ID da pessoa que deseja alterar:");
                int idP = id.nextInt();
                System.out.println("Insira o nome da pessoa que deseja alterar:");
                String novoNome = nome.nextLine();
                System.out.println("Insira o logradouro da pessoa que deseja alterar:");
                String novoLogradouro = logradouro.nextLine();
                System.out.println("Insira a cidade da pessoa que deseja alterar:");
                String novoCidade = cidade.nextLine();
```

```java
                System.out.println("Insira o estado da pessoa que deseja alterar:");
                String novoEstado = estado.nextLine();
                System.out.println("Insira o telefone da pessoa que deseja alterar:");
                String novoTelefone = telefone.nextLine();
                System.out.println("Insira o email da pessoa que deseja alterar:");
                String novoEmail = email.nextLine();

                String sql = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idpessoa = ?";
                PreparedStatement preparedStatement = connection.prepareStatement(sql);
                preparedStatement.setString( parameterIndex: 1, novoNome);
                preparedStatement.setString( parameterIndex: 2, novoLogradouro);
                preparedStatement.setString( parameterIndex: 3, novoCidade);
                preparedStatement.setString( parameterIndex: 4, novoEstado);
                preparedStatement.setString( parameterIndex: 5, novoTelefone);
                preparedStatement.setString( parameterIndex: 6, novoEmail);
                preparedStatement.setInt( parameterIndex: 7, idP);

                int linhasAfetadas = preparedStatement.executeUpdate();

                if (tipoPessoa.equals("f")) {
                    System.out.println("Insira o CPF atualizado:");
                    Long cpf = doc.nextLong();
                    String sqlPf = "UPDATE fisica SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ?, cpf = ? WHERE idpessoa = ?";
                    PreparedStatement preparedStatement1 = connection.prepareStatement(sqlPf);
                    preparedStatement1.setString( parameterIndex: 1, novoNome);
                    preparedStatement1.setString( parameterIndex: 2, novoLogradouro);
                    preparedStatement1.setString( parameterIndex: 3, novoCidade);
                    preparedStatement1.setString( parameterIndex: 4, novoEstado);
                    preparedStatement1.setString( parameterIndex: 5, novoTelefone);
                    preparedStatement1.setString( parameterIndex: 6, novoEmail);
                    preparedStatement1.setLong( parameterIndex: 7, cpf);
                    preparedStatement1.setInt( parameterIndex: 8, idP);

                    int linhasAfetadas1 = preparedStatement1.executeUpdate();
                    if (linhasAfetadas > 0 && linhasAfetadas1 > 0) {
                        System.out.println("Atualização bem-sucedida!");
                    } else {
                        System.out.println("Nenhum registro atualizado. Verifique o ID da pessoa.");
                    }
                } else if (tipoPessoa.equals("j")) {
                    System.out.println("Insira o CNPJ atualizado:");
                    Long cnpj = doc.nextLong();
                    String sqlPf = "UPDATE juridica SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ?, cnpj = ? WHERE idpessoa = ?";
                    PreparedStatement preparedStatement1 = connection.prepareStatement(sqlPf);
                    preparedStatement1.setString( parameterIndex: 1, novoNome);
                    preparedStatement1.setString( parameterIndex: 2, novoLogradouro);
                    preparedStatement1.setString( parameterIndex: 3, novoCidade);
                    preparedStatement1.setString( parameterIndex: 4, novoEstado);
                    preparedStatement1.setString( parameterIndex: 5, novoTelefone);
                    preparedStatement1.setString( parameterIndex: 6, novoEmail);
                    preparedStatement1.setLong( parameterIndex: 7, cnpj);
                    preparedStatement1.setInt( parameterIndex: 8, idP);

                    int linhasAfetadas1 = preparedStatement1.executeUpdate();
                    if (linhasAfetadas > 0 && linhasAfetadas1 > 0) {
                        System.out.println("Atualização bem-sucedida!");
                    } else {
                        System.out.println("Nenhum registro atualizado. Verifique o ID da pessoa.");
                    }
                }
```

```
            } else if (opcao == 3) {
                String tipoPessoa = Metodos.tipoPessoa();

                Scanner id = new Scanner(System.in);
                System.out.println("Digite o ID de quem deseja excluir:");
                int idP = id.nextInt();
                String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa = ?";
                PreparedStatement preparedStatement = connection.prepareStatement(sqlpessoa);
                preparedStatement.setInt( parameterIndex: 1, idP);
                int rowsDeletedPessoa = preparedStatement.executeUpdate();

                if (tipoPessoa.equals("f")) {
                    try {
                        String sql = "DELETE FROM fisica WHERE idpessoa = ?";
                        PreparedStatement preparedStatement1 = connection.prepareStatement(sql);
                        preparedStatement1.setInt( parameterIndex: 1, idP);
                        int rowsDeleted = preparedStatement1.executeUpdate();
                        if (rowsDeleted > 0 && rowsDeletedPessoa > 0) {
                            System.out.println("Registro deletado com sucesso!");
                        } else {
                            System.out.println("Nenhum registro encontrado com o ID fornecido!");
                        }
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                } else if (tipoPessoa.equals("j")) {
                    try {
                        String sql = "DELETE FROM juridica WHERE idpessoajuridica = ?";
                        PreparedStatement preparedStatement1 = connection.prepareStatement(sql);
                        preparedStatement1.setInt( parameterIndex: 1, idP);
                        int rowsDeleted = preparedStatement1.executeUpdate();
                        if (rowsDeleted > 0) {
                            System.out.println("Registro deletado com sucesso!");
                        } else {
                            System.out.println("Nenhum registro encontrado com o ID fornecido!");
                        }
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            } else if (opcao == 4) {
                String tipoPessoa = Metodos.tipoPessoa();
                if (tipoPessoa.equals("f")) {
                    Scanner id = new Scanner(System.in);
                    System.out.println("Digite o ID para realizar a busca:");
                    int idPf = id.nextInt();
                    try {
                        PreparedStatement preparedStatement = null;
                        ResultSet resultSet = null;
                        String sql = "SELECT * FROM fisica WHERE idpessoafisica = ?";
                        preparedStatement = connection.prepareStatement(sql);
                        preparedStatement.setInt( parameterIndex: 1, idPf);
                        resultSet = preparedStatement.executeQuery();

                        if (resultSet.next()) {
                            System.out.println("########################################");
                            System.out.println("Id: " + resultSet.getInt( columnLabel: "idpessoa"));
                            System.out.println("Nome: " + resultSet.getString( columnLabel: "nome"));
                            System.out.println("Logradouro: " + resultSet.getString( columnLabel: "logradouro"));
                            System.out.println("Cidade: " + resultSet.getString( columnLabel: "cidade"));
```



```
                        System.out.println("Estado: " + resultSet.getString( columnLabel: "estado"));
                        System.out.println("Telefone: " + resultSet.getString( columnLabel: "telefone"));
                        System.out.println("Email: " + resultSet.getString( columnLabel: "email"));
                        System.out.println("CPF: " + resultSet.getLong( columnLabel: "cpf"));
                        System.out.println("########################################");
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
            } else if (tipoPessoa.equals("j")) {
                Scanner id = new Scanner(System.in);
                int idPj = id.nextInt();
                try {
                    PreparedStatement preparedStatement = null;
                    ResultSet resultSet = null;
                    String sql = "SELECT * FROM juridica WHERE idpessoajuridica = ?";
                    preparedStatement = connection.prepareStatement(sql);
                    preparedStatement.setInt( parameterIndex: 1, idPj);
                    resultSet = preparedStatement.executeQuery();

                    if (resultSet.next()) {
                        System.out.println("########################################");
                        System.out.println("Id: " + resultSet.getInt( columnLabel: "idpessoa"));
                        System.out.println("Nome: " + resultSet.getString( columnLabel: "nome"));
                        System.out.println("Logradouro: " + resultSet.getString( columnLabel: "logradouro"));
                        System.out.println("Cidade: " + resultSet.getString( columnLabel: "cidade"));
                        System.out.println("Estado: " + resultSet.getString( columnLabel: "estado"));
                        System.out.println("Telefone: " + resultSet.getString( columnLabel: "telefone"));
                        System.out.println("Email: " + resultSet.getString( columnLabel: "email"));
                                System.out.println("CNPJ: " + resultSet.getLong( columnLabel: "cnpj
                                System.out.println("########################################")
                            }

                        } catch (SQLException e) {
                            e.printStackTrace();
                        }
                    }
                } else if (opcao == 5) {
                    String tipoPessoa = Metodos.tipoPessoa();
                    if (tipoPessoa.equals("f")) {
                        System.out.println("Exibindo dados de Pessoa Fisica...");
                        Metodos.restauraObjeto( tipoPessoa: "pf", connection);
                    } else if (tipoPessoa.equals("j")) {
                        System.out.println("Exibindo dados de Pessoa Jurídica...");
                        Metodos.restauraObjeto( tipoPessoa: "pj", connection);
                    }
                }
            }
            connection.close();
        } catch (SQLException e) {
        e.printStackTrace();
        }
    };
};

```

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

Eles desempenham um papel fundamental na integração de sistemas de software, na simplificação do desenvolvimento, na melhoria da portabilidade e

no aumento da segurança e do desempenho dos aplicativos. Eles são essenciais para construir sistemas robustos e escaláveis em um ambiente empresarial moderno.

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

   O `Statement` é adequado para consultas SQL dinâmicas simples, o `PreparedStatement` é a escolha preferida para a maioria dos casos devido à sua segurança aprimorada, desempenho melhorado e legibilidade do código.

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

   O padrão DAO é uma abordagem valiosa para melhorar a manutenibilidade do software, pois promove a separação de responsabilidades, reutilização de código e flexibilidade, tornando mais fácil e seguro modificar e expandir um sistema ao longo do tempo.

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

   Há duas abordagens comuns para refletir herança em um banco de dados estritamente relacional: herança por tabelas separadas e herança por tabela única.

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

   Persistência em arquivo pode ser mais simples para casos simples ou quando a estrutura de dados é relativamente simples, enquanto persistência em banco de dados é mais apropriada para casos que requerem estruturação, desempenho, segurança e recursos avançados de gerenciamento de dados.

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

   O uso simplificou a manipulação e impressão de valores em entidades ao permitir que os desenvolvedores escrevam código mais conciso e expressivo, melhorando a legibilidade e a manutenibilidade do código.

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

   Os métodos static são usados para operações que não dependem de instâncias e são chamados no contexto da classe em si.

Conclusão

Esse projeto tem por finalidade realizar a criação de um aplicativo Java, o qual possa realizar o acesso ao banco de dados através de um terminal, possibilitando realizar as funções de incluir pessoa (cadastrar no bd), alterar pessoa (editar bd), excluir pessoa (deletar elemento do bd), buscar pelo id (retornar dados específicos do bd), exibir todos os dados (retornar dados do bd) e finalizar o programa (encerrar execução do script).