FACULDADE ESTÁCIO DE SÁ

CURSO: DESENVOLVIMENTO FULL STACK

3º SEMESTRE – MATRÍCULA 202302595341

Repositório GitHub - alaimalmeida/CadastroBD (github.com)

ALAIM ALMEIDA DE OLIVEIRA

BackEnd sem banco não tem

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

Salvador – BA

2024

1. Introdução

Nesse projeto, com base em todo o material didático desenvolvido pela instituição para que pudéssemos ter experiência cabível, ficamos capazes de desenvolver um aplicativo cadastral em Java, capaz de gerenciar informações de pessoas físicas e jurídicas, utilizando persistência em um banco de dados relacional SQL Server. Para isso, foi adotada a utilização do padrão DAO (Data Access Object) para o manuseio dos dados e do middleware JDBC para a comunicação com o banco de dados.

2. Configuração do Netbeans e banco de dados

No primeiro momento, é criado um pacote chamado CadastroBD na aba de Projects e em Libraries, clicamos com o botão direito do mouse e logo em seguida clicamos no Add JAR/Folder, para inserir o arquivo **mssql-jdbc-12.2.0.jre11.jar** que vai ser utilizado na conexão ao banco de dados.

Na aba Services do NetBrans, é preciso liberar o acesso da divisão do Banco de Dados, clicando com o botão direito do mouse em Drivers e escolha Novo Driver. Na janela que abrirá, clica na opção ADD(Adicionar), escolhendo o arquivo jar utilizado no passo anterior e finalizar com Ok.

Logo em seguida, poderemos ver, caso esteja tudo nos conformes, o nome da conexão criada e clicando com o botão direito na mesma e clicando novamente em Connect Using para poder fazer a configuração dos campos database, user e password de acordo com as configurações do banco de dados já criados no SQL Server Management Studio.

No campo JBDBC URL, deve ser utilizada a expressão:

jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;

Em seguida, clica em Testar conexão, estando tudo correto, só finalizar.

3. Estrutura

3.1. Criação de pacotes

A estrutura do aplicativo se da na criação de alguns pacotes e suas classes para compor o mesmo, sendo eles:

- Foram criados os pacotes “cadastro.model”, “cadastro.model.util” e “cadastrobd.model”.
- No pacote “cadastrobd.model”, foram criadas as classes de modelo Pessoa, “PessoaFisica” e “PessoaJuridica”, representando entidades do sistema.

- No pacote “cadastro.model”, foram criadas as classes “PessoaFisicaDAO”, “PessoaJuridicaDAO”, e “CadastroBDTeste” responsáveis por realizar operações de CRUD no banco de dados.
- No pacote “cadastro.model.util”, foram criadas as classes “ConectorBD” e “SequenceManager”, responsáveis por estabelecer conexão com o banco de dados e gerenciar sequências, respectivamente.

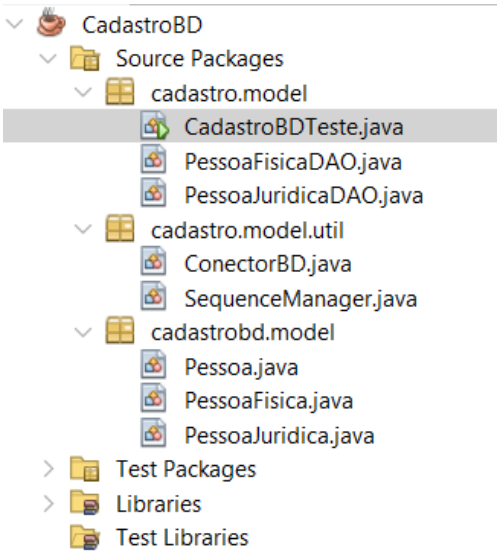

Imagem 01 – Estrutura do aplicativo

4. Mapeamento Objeto-Relacional

Nas classes de modelo “**Pessoa**”(campos de **Id**, **nome**, **logradouro**, **cidade**, **estado**, **telefone** e **email**), “**PessoaFisica**”(com os campos **idPessoaFisica**, **idPessoa** e **cpf**) e “**PessoaJuridica**”(com os campos **idPessoaJuridica**, **idPessoa** e **cnpj**) foram mapeadas para tabela no banco de dados SQL Server.

```java
/*
 * 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 cadastrobd.model;

/**
 *
 * @author Alaim
 */


// Classe Pessoa
public class Pessoa {
    int id;
    String nome;
    String logradouro;
    String cidade;
    String estado;
    String telefone;
    String email;

    // Construtor padrão
    public Pessoa() {
    }

    // Construtor completo
    public Pessoa(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
        this.id = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }
```

Imagem 02 – Classe Pessoa

```java
    //Getters e setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }
```

Imagem 03 – Classe Pessoa

```java

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    // Método para exibir os dados no console
    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);
    }
}
```

Imagem 04 – Classe Pessoa

```java
/*
 * 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 cadastrobd.model;

/**
 *
 * @author Alaim
 */

// Classe PessoaFisica
public class PessoaFisica extends Pessoa {
    private String cpf;

    // Construtor padrão
    public PessoaFisica() {
    }

    // Construtor completo
    public PessoaFisica(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cpf) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cpf = cpf;
    }

    //Getters e setters
    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
```

Imagem 05 – Classe PessoaFisica

```java
    // Método para exibir os dados no console (reescrita)
    @Override
    public String toString() {
        return "Id: " + id + "\n" +
               "Nome: " + nome + "\n" +
               "Logradouro: " + logradouro + "\n" +
               "Cidade: " + cidade + "\n" +
               "Estado: " + estado + "\n" +
               "Telefone: " + telefone + "\n" +
               "E-mail: " + email + "\n" +
               "CPF: " + cpf + "\n";
    }
}
```

Imagem 06 – Classe PessoaFisica

```java
/*
 * 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 cadastrobd.model;

/**
 *
 * @author Alaim
 */

// Classe PessoaJuridica
public class PessoaJuridica extends Pessoa {
    private String cnpj;

    // Construtor padrão
    public PessoaJuridica() {
    }

    // Construtor completo
    public PessoaJuridica(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cnpj) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cnpj = cnpj;
    }

    //Getters e setters
    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

```

Imagem 07 – Classe PessoaJuridica

```
35
36          // Método para exibir os dados no console (reescrita)
37          @Override
            public String toString() {
39              return "Id: " + id + "\n" +
40                     "Nome: " + nome + "\n" +
41                     "Logradouro: " + logradouro + "\n" +
42                     "Cidade: " + cidade + "\n" +
43                     "Estado: " + estado + "\n" +
44                     "Telefone: " + telefone + "\n" +
45                     "E-mail: " + email + "\n" +
46                     "CNPJ: " + cnpj + "\n";
47          }
48      }
```

Imagem 08 – Classe PessoaJuridica

5. Criação do aplicativo cadastral

O aplicativo cadastral permite ao usuário realizar diversas operações, como inclusão, alteração, exclusão, busca por ID e exibição de todas as pessoas cadastradas.

O menu principal do aplicativo é exibido no console, onde o usuário pode escolher a operação desejada digitando o número correspondente. Cada operação do menu é associada a um caso em um switch, onde a lógica específica é executada, fazendo uso dos métodos fornecidos pelas classes DAO.

Os métodos das classes DAO foram implementados de forma a realizar operações de manipulação de dados nas tabelas correspondentes.

Segue abaixo as classes para conexão com seus métodos:

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  package cadastro.model;
6
7  /**
8   *
9   * @author Alaim
10  */
11
12
13 import cadastro.model.util.ConectorBD;
14 import cadastrobd.model.PessoaFisica;
15 import java.sql.Connection;
16 import java.sql.PreparedStatement;
17 import java.sql.ResultSet;
18 import java.sql.SQLException;
19 import java.util.ArrayList;
20 import java.util.List;
21
22 // Classe PessoaFisicaDAO
23 public class PessoaFisicaDAO {
24     public PessoaFisica getPessoa(int id) {
25         PessoaFisica pessoa = null;
26         Connection conn = null;
27         PreparedStatement stmt = null;
28         ResultSet rs = null;
29
30         try {
31             conn = ConectorBD.getConnection();
32             stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaFisica ON pessoas.idPessoa = pessoaFisica.idPessoa WHERE pessoas.idPessoa = ?");
33             stmt.setInt(1, id);
34             rs = stmt.executeQuery();
35
```

Imagem 09 – Classe PessoaFisicaDao

```
35
36              if (rs.next()) {
37                  pessoa = new PessoaFisica(
38                          rs.getInt("idPessoa"),
39                          rs.getString("nome"),
40                          rs.getString("logradouro"),
41                          rs.getString("cidade"),
42                          rs.getString("estado"),
43                          rs.getString("telefone"),
44                          rs.getString("email"),
45                          rs.getString("cpf")
46                  );
47              }
48          } catch (SQLException e) {
49          } finally {
50              ConectorBD.close(rs);
51              ConectorBD.close(stmt);
52              ConectorBD.close(conn);
53          }
54
55          return pessoa;
56      }
57
58      public List<PessoaFisica> getPessoas() {
59          List<PessoaFisica> pessoas = new ArrayList<>();
60          Connection conn = null;
61          PreparedStatement stmt = null;
62          ResultSet rs = null;
63
64          try {
65              conn = ConectorBD.getConnection();
66              stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaFisica ON pessoas.idPessoa = pessoaFisica.idPessoa");
67              rs = stmt.executeQuery();
68
69              while (rs.next()) {
70                  PessoaFisica pessoa = new PessoaFisica(
```

Imagem 10 – Classe PessoaFisicaDao

```
71                          rs.getInt("idPessoa"),
72                          rs.getString("nome"),
73                          rs.getString("logradouro"),
74                          rs.getString("cidade"),
75                          rs.getString("estado"),
76                          rs.getString("telefone"),
77                          rs.getString("email"),
78                          rs.getString("cpf")
79                  );
80                  pessoas.add(pessoa);
81              }
82          } catch (SQLException e) {
83          } finally {
84              ConectorBD.close(rs);
85              ConectorBD.close(stmt);
86              ConectorBD.close(conn);
87          }
88
89          return pessoas;
90      }
91
92      public void incluir(PessoaFisica pessoa) {
93          Connection conn = null;
94          PreparedStatement stmt = null;
95
96          try {
97              conn = ConectorBD.getConnection();
98              stmt = conn.prepareStatement("INSERT INTO pessoas (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)",
99                      PreparedStatement.RETURN_GENERATED_KEYS);
100             stmt.setString(1, pessoa.getNome());
101             stmt.setString(2, pessoa.getLogradouro());
102             stmt.setString(3, pessoa.getCidade());
103             stmt.setString(4, pessoa.getEstado());
104             stmt.setString(5, pessoa.getTelefone());
105             stmt.setString(6, pessoa.getEmail());
```

Imagem 11 – Classe PessoaFisicaDao

```
106             stmt.executeUpdate();
107
108             ResultSet rs = stmt.getGeneratedKeys();
109             if (rs.next()) {
110                 int idPessoa = rs.getInt(1);
111                 stmt = conn.prepareStatement("INSERT INTO pessoaFisica (idPessoa, cpf) VALUES (?, ?)");
112                 stmt.setInt(1, idPessoa);
113                 stmt.setString(2, pessoa.getCpf());
114                 stmt.executeUpdate();
115                 pessoa.setId(idPessoa);
116             }
117         } catch (SQLException e) {
118         } finally {
119             ConectorBD.close(stmt);
120             ConectorBD.close(conn);
121         }
122     }
123
124     public void alterar(PessoaFisica pessoa) {
125         Connection conn = null;
126         PreparedStatement stmt = null;
127
128         try {
129             conn = ConectorBD.getConnection();
130             stmt = conn.prepareStatement("UPDATE pessoas SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?");
131             stmt.setString(1, pessoa.getNome());
132             stmt.setString(2, pessoa.getLogradouro());
133             stmt.setString(3, pessoa.getCidade());
134             stmt.setString(4, pessoa.getEstado());
135             stmt.setString(5, pessoa.getTelefone());
136             stmt.setString(6, pessoa.getEmail());
137             stmt.setInt(7, pessoa.getId());
138             stmt.executeUpdate();
139
```

Imagem 12 – Classe PessoaFisicaDao

```
                stmt = conn.prepareStatement("UPDATE pessoaFisica SET cpf=? WHERE idPessoa=?");
                stmt.setString(1, pessoa.getCpf());
                stmt.setInt(2, pessoa.getId());
                stmt.executeUpdate();
            } catch (SQLException e) {
            } finally {
                ConectorBD.close(stmt);
                ConectorBD.close(conn);
            }
        }

        public void excluir(int id) {
            Connection conn = null;
            PreparedStatement stmt = null;

            try {
                conn = ConectorBD.getConnection();
                stmt = conn.prepareStatement("DELETE FROM pessoaFisica WHERE idPessoa=?");
                stmt.setInt(1, id);
                stmt.executeUpdate();

                stmt = conn.prepareStatement("DELETE FROM pessoas WHERE idPessoa=?");
                stmt.setInt(1, id);
                stmt.executeUpdate();
            } catch (SQLException e) {
            } finally {
                ConectorBD.close(stmt);
                ConectorBD.close(conn);
            }
        }
    }
```

Imagem 13 – Classe PessoaFisicaDao

```
/*
 * 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 cadastro.model;

/**
 *
 * @author Alaim
 */

import cadastro.model.util.ConectorBD;
import cadastrobd.model.PessoaJuridica;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

// Classe PessoaJuridicaDAO
public class PessoaJuridicaDAO {
    public PessoaJuridica getPessoa(int id) {
        PessoaJuridica pessoa = null;
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaJuridica ON pessoas.idPessoa = pessoaJuridica.idPessoa WHERE pessoas.idPessoa = ?");
            stmt.setInt(1, id);
            rs = stmt.executeQuery();

            if (rs.next()) {
                pessoa = new PessoaJuridica(
```

Imagem 14 – Classe PessoaJuridicaDao

```
                        rs.getInt("idPessoa"),
                        rs.getString("nome"),
                        rs.getString("logradouro"),
                        rs.getString("cidade"),
                        rs.getString("estado"),
                        rs.getString("telefone"),
                        rs.getString("email"),
                        rs.getString("cnpj")
                    );
                }
            } catch (SQLException e) {
            } finally {
                ConectorBD.close(rs);
                ConectorBD.close(stmt);
                ConectorBD.close(conn);
            }

            return pessoa;
        }

        public List<PessoaJuridica> getPessoas() {
            List<PessoaJuridica> pessoas = new ArrayList<>();
            Connection conn = null;
            PreparedStatement stmt = null;
            ResultSet rs = null;

            try {
                conn = ConectorBD.getConnection();
                stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaJuridica ON pessoas.idPessoa = pessoaJuridica.idPessoa");
                rs = stmt.executeQuery();

                while (rs.next()) {
                    PessoaJuridica pessoa = new PessoaJuridica(
                            rs.getInt("idPessoa"),
                            rs.getString("nome"),
                            rs.getString("logradouro"),
```

Imagem 15 – Classe PessoaJuridicaDao

```
                        rs.getString("logradouro"),
                        rs.getString("cidade"),
                        rs.getString("estado"),
                        rs.getString("telefone"),
                        rs.getString("email"),
                        rs.getString("cnpj")
                    );
                    pessoas.add(pessoa);
                }
            } catch (SQLException e) {
            } finally {
                ConectorBD.close(rs);
                ConectorBD.close(stmt);
                ConectorBD.close(conn);
            }

            return pessoas;
        }

        public void incluir(PessoaJuridica pessoa) {
            Connection conn = null;
            PreparedStatement stmt = null;

            try {
                conn = ConectorBD.getConnection();
                stmt = conn.prepareStatement("INSERT INTO pessoas (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)",
                        PreparedStatement.RETURN_GENERATED_KEYS);
                stmt.setString(1, pessoa.getNome());
                stmt.setString(2, pessoa.getLogradouro());
                stmt.setString(3, pessoa.getCidade());
                stmt.setString(4, pessoa.getEstado());
                stmt.setString(5, pessoa.getTelefone());
                stmt.setString(6, pessoa.getEmail());
                stmt.executeUpdate();

                ResultSet rs = stmt.getGeneratedKeys();
```

Imagem 16 – Classe PessoaJuridicaDao

```
                ResultSet rs = stmt.getGeneratedKeys();
                if (rs.next()) {
                    int idPessoa = rs.getInt(1);
                    stmt = conn.prepareStatement("INSERT INTO pessoaJuridica (idPessoa, cnpj) VALUES (?, ?)");
                    stmt.setInt(1, idPessoa);
                    stmt.setString(2, pessoa.getCnpj());
                    stmt.executeUpdate();
                    pessoa.setId(idPessoa);
                }
            } catch (SQLException e) {
            } finally {
                ConectorBD.close(stmt);
                ConectorBD.close(conn);
            }
        }

        public void alterar(PessoaJuridica pessoa) {
            Connection conn = null;
            PreparedStatement stmt = null;

            try {
                conn = ConectorBD.getConnection();
                stmt = conn.prepareStatement("UPDATE pessoas SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?");
                stmt.setString(1, pessoa.getNome());
                stmt.setString(2, pessoa.getLogradouro());
                stmt.setString(3, pessoa.getCidade());
                stmt.setString(4, pessoa.getEstado());
                stmt.setString(5, pessoa.getTelefone());
                stmt.setString(6, pessoa.getEmail());
                stmt.setInt(7, pessoa.getId());
                stmt.executeUpdate();

                stmt = conn.prepareStatement("UPDATE pessoaJuridica SET cnpj=? WHERE idPessoa=?");
                stmt.setString(1, pessoa.getCnpj());
                stmt.setInt(2, pessoa.getId());
                stmt.executeUpdate();
```

Imagem 17 – Classe PessoaJuridicaDao

```
            stmt.executeUpdate();
        } catch (SQLException e) {
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }
    }

    public void excluir(int id) {
        Connection conn = null;
        PreparedStatement stmt = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = conn.prepareStatement("DELETE FROM pessoaJuridica WHERE idPessoa=?");
            stmt.setInt(1, id);
            stmt.executeUpdate();

            stmt = conn.prepareStatement("DELETE FROM pessoas WHERE idPessoa=?");
            stmt.setInt(1, id);
            stmt.executeUpdate();
        } catch (SQLException e) {
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }
    }
}
```

Imagem 18 – Classe PessoaJuridicaDao

```
/*
 * 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 cadastro.model.util;

/**
 *
 * @author Alaim
 */

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

// Classe ConectorBD
public class ConectorBD {
    private static final String URL = "jdbc:sqlserver://localhost\\LFHMMEL:1433;databaseName=Loja;encrypt=true;trustServerCertificate=true;";
    private static final String USER = "loja";
    private static final String PASSWORD = "loja";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    public static PreparedStatement getPrepared(String sql) throws SQLException {
        return getConnection().prepareStatement(sql);
    }

    public static ResultSet getSelect(String sql) throws SQLException {
        return getPrepared(sql).executeQuery();
    }

}
```

Imagem 19 – Classe ConectorBD

```
    public static void close(Statement stmt) {
        if (stmt != null) {
            try {
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public static void close(ResultSet rs) {
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public static void close(Connection conn) {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Imagem 20 – Classe ConectorBD

```
/*
 * 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 cadastro.model.util;
/**
 *
 * @author Alaim
 */
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Classe SequenceManager
public class SequenceManager {
    public static int getValue(String sequenceName) throws SQLException {
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        int nextValue = -1;
        try {
            conn = ConectorBD.getConnection();
            stmt = conn.prepareStatement("SELECT NEXTVAL(?)");
            stmt.setString(1, sequenceName);
            rs = stmt.executeQuery();
            if (rs.next()) {
                nextValue = rs.getInt(1);
            }
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }
        return nextValue;
    }
}
```

Imagem 21 – Classe SequenceManager

E por fim, a classe principal que por ela, faz exibir o menu no console para poder cadastrar, alterar e excluir todas as pessoas físicas e jurídicas

```java
/*
 * 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 cadastro.model;

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;

import java.util.List;
import java.util.Scanner;

// Classe CadastroBDTeste
public class CadastroBDTeste {
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

            int opcao;
            do {
                System.out.println("============================");
                System.out.println("1 - Incluir Pessoa");
                System.out.println("2 - Alterar Pessoa");
                System.out.println("3 - Excluir Pessoa");
                System.out.println("4 - Buscar pelo ID");
                System.out.println("5 - Exibir todos");
                System.out.println("0 - Finalizar Programa");
                System.out.println("============================");
                opcao = scanner.nextInt();

                switch (opcao) {
                    case 1 -> incluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    case 2 -> alterar(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    case 3 -> excluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    case 4 -> obter(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    case 5 -> obterTodos(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
```

Imagem 22 – Classe CadastroBDTeste

```java
                    case 5 -> obterTodos(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    case 0 -> System.out.println("Programa finalizado.");
                    default -> System.out.println("Opcao invalida.");
                }
            } while (opcao != 0);
        }
    }

    private static void incluir(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO)
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipo = scanner.next();

        try {
            switch (tipo.toUpperCase()) {
                case "F" -> {
                    System.out.println("Cadastro de Pessoa Fisica:");
                    PessoaFisica pessoaFisica = lerDadosPessoaFisica(scanner);
                    pessoaFisicaDAO.incluir(pessoaFisica);
                    System.out.println("Pessoa fisica incluida com sucesso...");
                    System.out.println("============================");
                }
                case "J" -> {
                    System.out.println("Cadastro de Pessoa Juridica:");
                    PessoaJuridica pessoaJuridica = lerDadosPessoaJuridica(scanner);
                    pessoaJuridicaDAO.incluir(pessoaJuridica);
                    System.out.println("Pessoa juridica incluida com sucesso...");
                    System.out.println("============================");
                }
                default -> System.out.println("Tipo de pessoa invalido.");
            }
        } catch (Exception e) {
            System.out.println("Ocorreu um erro ao tentar incluir a pessoa: " + e.getMessage());
        }
    }

```

Imagem 23 – Classe CadastroBDTeste

```
72      private static PessoaFisica lerDadosPessoaFisica(Scanner scanner) {
73          PessoaFisica pessoaFisica = new PessoaFisica();
74          scanner.nextLine();
75          System.out.print("Nome: ");
76          pessoaFisica.setNome(scanner.nextLine());
77          System.out.print("Logradouro: ");
78          pessoaFisica.setLogradouro(scanner.nextLine());
79          System.out.print("Cidade: ");
80          pessoaFisica.setCidade(scanner.nextLine());
81          System.out.print("Estado: ");
82          pessoaFisica.setEstado(scanner.nextLine());
83          System.out.print("Telefone: ");
84          pessoaFisica.setTelefone(scanner.nextLine());
85          System.out.print("Email: ");
86          pessoaFisica.setEmail(scanner.nextLine());
87          System.out.print("CPF: ");
88          pessoaFisica.setCpf(scanner.nextLine());
89          return pessoaFisica;
90      }
91
92      private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner) {
93          PessoaJuridica pessoaJuridica = new PessoaJuridica();
94          scanner.nextLine();
95          System.out.print("Nome: ");
96          pessoaJuridica.setNome(scanner.nextLine());
97          System.out.print("Logradouro: ");
98          pessoaJuridica.setLogradouro(scanner.nextLine());
99          System.out.print("Cidade: ");
100         pessoaJuridica.setCidade(scanner.nextLine());
101         System.out.print("Estado: ");
102         pessoaJuridica.setEstado(scanner.nextLine());
103         System.out.print("Telefone: ");
104         pessoaJuridica.setTelefone(scanner.nextLine());
105         System.out.print("Email: ");
106         pessoaJuridica.setEmail(scanner.nextLine());
107         System.out.print("CNPJ: ");
```

Imagem 24 – Classe CadastroBDTeste

```
107         System.out.print("CNPJ: ");
108         pessoaJuridica.setCnpj(scanner.nextLine());
109         return pessoaJuridica;
110     }
111
112     private static void alterar(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
113         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
114         String tipo = scanner.next();
115
116         System.out.print("Digite o ID da pessoa: ");
117         int id = scanner.nextInt();
118
119         try {
120             switch (tipo.toUpperCase()) {
121                 case "F" -> {
122                     PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
123                     if (pessoaFisica != null) {
124                         System.out.println("Dados atuais da Pessoa Fisica:");
125                         System.out.println(pessoaFisica);
126
127                         PessoaFisica novosDados = lerDadosPessoaFisica(scanner);
128                         novosDados.setId(id);
129                         pessoaFisicaDAO.alterar(novosDados);
130                         System.out.println("Pessoa fisica alterada com sucesso.");
131                     } else {
132                         System.out.println("Pessoa fisica nao encontrada.");
133                     }
134                 }
135                 case "J" -> {
136                     PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
137                     if (pessoaJuridica != null) {
138                         System.out.println("Dados atuais da Pessoa Juridica:");
139                         System.out.println(pessoaJuridica);
140
141                         PessoaJuridica novosDados = lerDadosPessoaJuridica(scanner);
142                         novosDados.setId(id);
```

Imagem 25 – Classe CadastroBDTeste

```
                    novosDados.setId(id);
                    pessoaJuridicaDAO.alterar(novosDados);
                    System.out.println("Pessoa juridica alterada com sucesso.");
                } else {
                    System.out.println("Pessoa juridica nao encontrada.");
                }
            }
            default -> System.out.println("Tipo de pessoa invalido.");
        }
    } catch (Exception e) {
        System.out.println("Ocorreu um erro ao tentar alterar a pessoa: " + e.getMessage());
    }
}

private static void excluir(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipo = scanner.next();

    System.out.print("Digite o ID da pessoa: ");
    int id = scanner.nextInt();

    try {
        switch (tipo.toUpperCase()) {
            case "F" -> {
                pessoaFisicaDAO.excluir(id);
                System.out.println("Pessoa fisica excluida com sucesso.");
            }
            case "J" -> {
                pessoaJuridicaDAO.excluir(id);
                System.out.println("Pessoa juridica excluida com sucesso.");
            }
            default -> System.out.println("Tipo de pessoa invalido.");
        }
    } catch (Exception e) {
        System.out.println("Ocorreu um erro ao tentar excluir a pessoa: " + e.getMessage());
    }
}
```

Imagem 26 – Classe CadastroBDTeste

```
private static void obter(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipo = scanner.next();

    System.out.print("Digite o ID da pessoa: ");
    int id = scanner.nextInt();

    try {
        switch (tipo.toUpperCase()) {
            case "F" -> {
                PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
                if (pessoaFisica != null) {
                    System.out.println("Pessoa fisica encontrada:");
                    System.out.println(pessoaFisica);
                } else {
                    System.out.println("Pessoa fisica nao encontrada.");
                }
            }
            case "J" -> {
                PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
                if (pessoaJuridica != null) {
                    System.out.println("Pessoa juridica encontrada:");
                    System.out.println(pessoaJuridica);
                } else {
                    System.out.println("Pessoa juridica nao encontrada.");
                }
            }
            default -> System.out.println("Tipo de pessoa invalido.");
        }
    } catch (Exception e) {
        System.out.println("Ocorreu um erro ao tentar buscar a pessoa: " + e.getMessage());
    }
}
```

Imagem 27 – Classe CadastroBDTeste

```
214     private static void obterTodos(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
215         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
216         String tipo = scanner.next();
217 
218         try {
219             switch (tipo.toUpperCase()) {
220                 case "F" -> {
221                     List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
222                     System.out.println("Exibindo dados de Pessoa Fisica...");
223                     System.out.println("==========================");
224                     for (PessoaFisica pf : pessoasFisicas) {
225                         System.out.println("Id: " + pf.getId());
226                         System.out.println("Nome: " + pf.getNome());
227                         System.out.println("Logradouro: " + pf.getLogradouro());
228                         System.out.println("Cidade: " + pf.getCidade());
229                         System.out.println("Estado: " + pf.getEstado());
230                         System.out.println("Telefone: " + pf.getTelefone());
231                         System.out.println("E-mail: " + pf.getEmail());
232                         System.out.println("CPF: " + pf.getCpf());
233                     }
234                 }
235                 case "J" -> {
236                     List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
237                     System.out.println("Exibindo dados de Pessoa Juridica...");
238                     System.out.println("==========================");
239                     for (PessoaJuridica pj : pessoasJuridicas) {
240                         System.out.println("Id: " + pj.getId());
241                         System.out.println("Nome: " + pj.getNome());
242                         System.out.println("Logradouro: " + pj.getLogradouro());
243                         System.out.println("Cidade: " + pj.getCidade());
244                         System.out.println("Estado: " + pj.getEstado());
245                         System.out.println("Telefone: " + pj.getTelefone());
246                         System.out.println("E-mail: " + pj.getEmail());
247                         System.out.println("CNPJ: " + pj.getCnpj());
248                     }
249                 }
250                 default -> System.out.println("Tipo de pessoa invalido.");
```

Imagem 28 – Classe CadastroBDTeste

```
249                     }
250                     default -> System.out.println("Tipo de pessoa invalido.");
251                 }
252             } catch (Exception e) {
253                 System.out.println("Ocorreu um erro ao tentar exibir as pessoas: " + e.getMessage());
254             }
255         }
256     }
```

Imagem 29 – Classe CadastroBDTeste

6. Considerações Finais:

Com isso, se finaliza toda a estrutura do Netbeans, com todas as suas classes com seus métodos e suas respectivas heranças e todas as suas conexões com o banco.

O projeto foi desenvolvido com sucesso, seguindo as boas práticas de programação e utilizando os recursos fornecidos pelo middleware JDBC e pelo padrão DAO.

A implementação do aplicativo cadastral oferece uma interface simples e intuitiva para o usuário, permitindo o gerenciamento eficiente de informações de pessoas físicas e jurídicas.

O uso do SQL Server como banco de dados proporciona robustez, segurança e escalabilidade ao sistema.