BackEnd sem banco não tem

João Pedro da Silva Zanirati Nunes – 202209081405

Polo Azenha
Nível 3: Back-end Sem Banco Não Tem – 9003 – Mundo 3

Objetivo da Prática

O objetivo do projeto é desenvolver um sistema de cadastro que permita ao usuário armazenar, gerenciar e interagir com informações do banco de dados de pessoas físicas e jurídicas por meio de uma interface de texto.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

```
package cadastrobd.model;

public class Pessoa {

    private int idPessoa;

    private String nome;

    private String cidade;

    private String endereco;

    private String uf;

    private int telefone;

    private String tipoPessoa;

     public Pessoa(int idPessoa, String nome, String cidade, String endereco, String uf, int telefone, String tipoPessoa) {

        this.idPessoa = idPessoa;

        this.nome = nome;

        this.cidade = cidade;

        this.endereco = endereco;
```

```
    this.uf = uf;
    this.telefone = telefone;
    this.tipoPessoa = tipoPessoa;
}
public int getIdPessoa(){
    return idPessoa;
}
public String getNome(){
    return nome;
}
public void setNome(String nome){
    this.nome = nome;
}
public String getCidade(){
    return cidade;
}
public void setCidade(String cidade){
    this.cidade = cidade;
}
public String getEndereco(){
    return endereco;
}

public void setEndereco(String endereco){
    this.endereco = endereco;
```

```
}
public String getUf(){
    return uf;
}
public void setUf(String uf){
    this.uf = uf;
}
public int getTelefone() {
    return telefone;
}
public void setTelefone(int telefone) {
    this.telefone = telefone;
}
public String getTipoPessoa() {
    return tipoPessoa;
}
public void setTipoPessoa(String tipoPessoa) {
    this.tipoPessoa = tipoPessoa;
}
public void exibir() {
    System.out.println("ID: " + idPessoa);
    System.out.println("Nome: " + nome);
    System.out.println("Cidade: " + cidade);
    System.out.println("Endereço: " + endereco);
    System.out.println("Estado: " + uf);
```

```
        System.out.println("Telefone: " + telefone);
        System.out.println("Tipo: " + tipoPessoa);
    }
}
```

```
package cadastrobd.model;
import java.util.Date;
public class PessoaFisica extends Pessoa {
    private int idFisica;
    private long cpf;
    private Date dt_nasc;
        public PessoaFisica(int idPessoa, String nome, String cidade, String endereco, String uf, int telefone, String tipoPessoa, int idFisica, long cpf, Date dt_nasc) {
        super(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa);
        this.idFisica = idFisica;
        this.cpf = cpf;
        this.dt_nasc = dt_nasc;
    }
    public int getIdFisica() {
        return idFisica;
    }
    public void setIdFisica(int idFisica) {
        this.idFisica = idFisica;
    }
    public long getCpf() {
```

```
        return cpf;
    }
    public void setCpf(long cpf) {
        this.cpf = cpf;
    }
    public java.sql.Date getDtNasc() {
        return new java.sql.Date(dt_nasc.getTime()) ;
    }
    public void setDtNasc(Date dt_nasc) {
        this.dt_nasc = dt_nasc;
    }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("ID Pessoa Fisica: " + idFisica);
        System.out.println("CPF: " + cpf);
        System.out.println("Data de Nascimento: " + dt_nasc);
    }
}
```

```
package cadastrobd.model;
public class PessoaJuridica extends Pessoa {
    private int idJuridica;
    private long cnpj;
    private String razaoSocial;
```

```
    public PessoaJuridica(int idPessoa, String nome, String cidade, String endereco, String uf, int telefone, String tipoPessoa, int idJuridica, long cnpj, String razaoSocial) {
        super(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa);
        this.idJuridica = idJuridica;
        this.cnpj = cnpj;
        this.razaoSocial = razaoSocial;
    }
    public int getIdJuridica() {
        return idJuridica;
    }
    public void setIdJuridica(int idJuridica) {
        this.idJuridica = idJuridica;
    }
    public long getCnpj() {
        return cnpj;
    }
    public void setCnpj(long cnpj) {
        this.cnpj = cnpj;
    }
    public String getRazaoSocial() {
        return razaoSocial;
    }
    public void setRazaoSocial(String razaoSocial) {
        this.razaoSocial = razaoSocial;
```

```
    }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("Id Pessoa Juridica: " + idJuridica);
        System.out.println("CNPJ: " + cnpj);
        System.out.println("Razão Social: " + razaoSocial);
    }
}
package cadastrobd.model;
import cadastro.model.util.ConectorBD;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
public class PessoaFisicaDAO {
    public static PessoaFisica getPessoa(int idPessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultSet = null;

        try {
```

```
            connection = ConectorBD.getConnection();
                statement = ConectorBD.getPrepared("SELECT * FROM pessoa_Fisica WHERE idPessoa = ?");
            statement.setInt(1, idPessoa);
            resultSet = statement.executeQuery();

            if (resultSet.next()) {

                String nome = resultSet.getString("nome");
                String cidade = resultSet.getString("cidade");
                String endereco = resultSet.getString("endereco");
                String uf = resultSet.getString("uf");
                int telefone = resultSet.getInt("telefone");
                String tipoPessoa = resultSet.getString("tipoPessoa");
                int idFisica = resultSet.getInt("idFisica");
                long cpf = resultSet.getLong("cpf");
                Date dt_nasc = resultSet.getDate("dt_nasc");

                return new PessoaFisica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idFisica, cpf, dt_nasc);
            }
            return null;
        } catch (SQLException e) {
            return null;
        } finally {
            ConectorBD.close(resultSet);
```

```
        ConectorBD.close(statement);
        ConectorBD.close(connection);
    }
}

public List<PessoaFisica> getPessoas() {
    List<PessoaFisica> pessoas = new ArrayList<>();
    Connection connection = null;
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    try {
        connection = ConectorBD.getConnection();
        statement = ConectorBD.getPrepared("SELECT * FROM pessoa_Fisica");
        resultSet = statement.executeQuery();

        while (resultSet.next()) {
            int idPessoa = resultSet.getInt("idPessoa");
            String nome = resultSet.getString("nome");
            String cidade = resultSet.getString("cidade");
            String endereco = resultSet.getString("endereco");
            String uf = resultSet.getString("uf");
            int telefone = resultSet.getInt("telefone");
            String tipoPessoa = resultSet.getString("tipoPessoa");
            int idFisica = resultSet.getInt("idFisica");
```

```
            long cpf = resultSet.getLong("cpf");
            Date dt_nasc = resultSet.getDate("dt_nasc");

                PessoaFisica pessoa = new PessoaFisica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idFisica, cpf, dt_nasc);
            pessoas.add(pessoa);
          }
          return pessoas;
        } catch (SQLException e) {
          return pessoas;
        } finally {
          ConectorBD.close(resultSet);
          ConectorBD.close(statement);
          ConectorBD.close(connection);
        }
    }

    public void incluir(PessoaFisica pessoa) {
        Connection connection = null;
        PreparedStatement pessoaStatement = null;
        PreparedStatement fisicaStatement = null;

        try {
          connection = ConectorBD.getConnection();
          connection.setAutoCommit(false);
```

```
pessoaStatement = ConectorBD.getPrepared("INSERT INTO pessoa (nome, cidade, endereco, uf, telefone, tipoPessoa) VALUES (?, ?, ?, ?, ?, ?)");
pessoaStatement.setString(1, pessoa.getNome());
pessoaStatement.setString(2, pessoa.getCidade());
pessoaStatement.setString(3, pessoa.getEndereco());
pessoaStatement.setString(4, pessoa.getUf());
pessoaStatement.setInt(5, pessoa.getTelefone());
pessoaStatement.setString(6, pessoa.getTipoPessoa());
pessoaStatement.executeUpdate();

ResultSet generatedKeys = pessoaStatement.getGeneratedKeys();
int idFisica = 0;
if (generatedKeys.next()) {
   idFisica = generatedKeys.getInt(1);
}

fisicaStatement = connection.prepareStatement("INSERT INTO pessoa_Fisica (cpf, dt_Nasc, idPessoa) VALUES (?, ?, ?)");
fisicaStatement.setLong(1, pessoa.getCpf());
fisicaStatement.setDate(2, pessoa.getDtNasc());
fisicaStatement.setInt(3, idFisica);
fisicaStatement.executeUpdate();

connection.commit();
System.out.println("Adicionado com sucesso!");
```

```
        } catch (SQLException e) {
            System.out.println("Erro ao adicionar!" +e);
            try {
                if (connection != null) {
                    connection.rollback();
                }
            } catch (SQLException rollbackEx) {
            }
        } finally {
            ConectorBD.close(pessoaStatement);
            ConectorBD.close(fisicaStatement);
            ConectorBD.close(connection);
        }
    }
    public void alterar(PessoaFisica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        try {
            connection = ConectorBD.getConnection();
            connection.setAutoCommit(false);

             statement = ConectorBD.getPrepared("UPDATE pessoa SET nome=?, cidade=?, endereco=?,
uf=?, telefone=?, tipoPessoa=? WHERE idPessoa=?");
            statement.setString(1, pessoa.getNome());
            statement.setString(2, pessoa.getCidade());
```

```
            statement.setString(3, pessoa.getEndereco());
            statement.setString(4, pessoa.getUf());
            statement.setInt(5, pessoa.getTelefone());
            statement.setString(6, pessoa.getTipoPessoa());
            statement.setInt(7, pessoa.getIdPessoa());
            statement.executeUpdate();


                    statement = ConectorBD.getPrepared("UPDATE pessoa_Fisica SET cpf=?, dt_nasc=?
WHERE idFisica=?");
            statement.setLong(1, pessoa.getCpf());
            statement.setDate(2, pessoa.getDtNasc());
            statement.setInt(3, pessoa.getIdFisica());
            statement.executeUpdate();


            connection.commit();
            System.out.println("Atualizado com sucesso!");
        } catch (SQLException e) {
            System.out.println("Erro ao atualizar!" +e);
            try {
                if (connection != null) {
                    connection.rollback();
                }
            } catch (SQLException rollbackEx) {
            }
        } finally {
```

```
        ConectorBD.close(statement);
        ConectorBD.close(connection);
    }
}

public void excluir(PessoaFisica pessoa) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = ConectorBD.getConnection();
        connection.setAutoCommit(false);

        statement = ConectorBD.getPrepared("DELETE FROM pessoa_Fisica WHERE idFisica=?");
        statement.setInt(1, pessoa.getIdFisica());
        statement.executeUpdate();

        statement = ConectorBD.getPrepared("DELETE FROM pessoa WHERE idPessoa=?");
        statement.setInt(1, pessoa.getIdPessoa());
        statement.executeUpdate();

        connection.commit();
        System.out.println("Removido com sucesso!");
    } catch (SQLException e) {
        System.out.println("Erro ao Remover!" +e);
```

```
            try {
                if (connection != null) {
                    connection.rollback();
                }
            } catch (SQLException rollbackEx) {
            }
        } finally {
            ConectorBD.close(statement);
            ConectorBD.close(connection);
        }
    }
}
package cadastrobd.model;
import cadastro.model.util.ConectorBD;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class PessoaJuridicaDAO {
    public static PessoaJuridica getPessoa(int idPessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultSet = null;
```

```
        try {
            connection = ConectorBD.getConnection();
             statement = ConectorBD.getPrepared("SELECT * FROM pessoa_Juridica WHERE idJuridica = ?");
            statement.setInt(1, idPessoa);
            resultSet = statement.executeQuery();

            if (resultSet.next()) {

                String nome = resultSet.getString("nome");
                String cidade = resultSet.getString("cidade");
                String endereco = resultSet.getString("endereco");
                String uf = resultSet.getString("uf");
                int telefone = resultSet.getInt("telefone");
                String tipoPessoa = resultSet.getString("tipoPessoa");
                int idJuridica = resultSet.getInt("idJuridica");
                long cnpj = resultSet.getLong("cnpj");
                String razaoSocial = resultSet.getString("razaoSocial");

                    return new PessoaJuridica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idJuridica, cnpj, razaoSocial);
            }
            return null;
        } catch (SQLException e) {
            return null;
```

```
    } finally {
        ConectorBD.close(resultSet);
        ConectorBD.close(statement);
        ConectorBD.close(connection);
    }
}

public List<PessoaJuridica> getPessoas() {
    List<PessoaJuridica> pessoas = new ArrayList<>();
    Connection connection = null;
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    try {
        connection = ConectorBD.getConnection();
        statement = ConectorBD.getPrepared("SELECT * FROM pessoa_Juridica");
        resultSet = statement.executeQuery();

        while (resultSet.next()) {
            int idPessoa = resultSet.getInt("idPessoa");
            String nome = resultSet.getString("nome");
            String cidade = resultSet.getString("cidade");
            String endereco = resultSet.getString("endereco");
            String uf = resultSet.getString("uf");
            int telefone = resultSet.getInt("telefone");
```

```
                String tipoPessoa = resultSet.getString("tipoPessoa");
                int idJuridica = resultSet.getInt("idJuridica");
                long cnpj = resultSet.getLong("cnpj");
                String razaoSocial = resultSet.getString("razaoSocial");

                PessoaJuridica pessoa = new PessoaJuridica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idJuridica, cnpj, razaoSocial);
                pessoas.add(pessoa);
            }
            return pessoas;
        } catch (SQLException e) {
            return pessoas;
        } finally {
            ConectorBD.close(resultSet);
            ConectorBD.close(statement);
            ConectorBD.close(connection);
        }
    }

    public void incluir(PessoaJuridica pessoa) {
        Connection connection = null;
        PreparedStatement pessoaStatement = null;
        PreparedStatement juridicaStatement = null;
        try {
            connection = ConectorBD.getConnection();
```

```
connection.setAutoCommit(false);


 pessoaStatement = ConectorBD.getPrepared("INSERT INTO pessoa (nome, cidade, endereco, uf, telefone, tipoPessoa) VALUES (?, ?, ?, ?, ?, ?)");
pessoaStatement.setString(1, pessoa.getNome());
pessoaStatement.setString(2, pessoa.getCidade());
pessoaStatement.setString(3, pessoa.getEndereco());
pessoaStatement.setString(4, pessoa.getUf());
pessoaStatement.setInt(5, pessoa.getTelefone());
pessoaStatement.setString(6, pessoa.getTipoPessoa());
pessoaStatement.executeUpdate();


ResultSet generatedKeys = pessoaStatement.getGeneratedKeys();
int idJuridica = -1;
if (generatedKeys.next()) {
   idJuridica = generatedKeys.getInt(1);
}


        juridicaStatement = ConectorBD.getPrepared("INSERT INTO pessoa_Juridica (cnpj, razaoSocial, idJuridica) VALUES (?, ?, ?)");
juridicaStatement.setLong(2, pessoa.getCnpj());
juridicaStatement.setString(3, pessoa.getRazaoSocial());
juridicaStatement.setInt(1, idJuridica);
juridicaStatement.executeUpdate();
connection.commit();
System.out.println("Adicionado com sucesso!");
```

```
    } catch (SQLException e) {
        System.out.println("Erro ao adicionar!" +e);
        try {
            if (connection != null) {
                connection.rollback();
            }
        } catch (SQLException rollbackEx) {

        }
    } finally {
        ConectorBD.close(pessoaStatement);
        ConectorBD.close(juridicaStatement);
        ConectorBD.close(connection);
    }
}

public void alterar(PessoaJuridica pessoa) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = ConectorBD.getConnection();
        connection.setAutoCommit(false);
```

```
            statement = ConectorBD.getPrepared("UPDATE pessoa SET nome=?, cidade=?, endereco=?, uf=?, telefone=?, tipoPessoa=? WHERE id=?");
            statement.setString(1, pessoa.getNome());
            statement.setString(2, pessoa.getCidade());
            statement.setString(3, pessoa.getEndereco());
            statement.setString(4, pessoa.getUf());
            statement.setInt(5, pessoa.getTelefone());
            statement.setString(6, pessoa.getTipoPessoa());
            statement.setInt(7, pessoa.getIdPessoa());
            statement.executeUpdate();


            statement = ConectorBD.getPrepared("UPDATE pessoa_Juridica SET cnpj=?, razaoSocial=? WHERE id=?");
            statement.setLong(1, pessoa.getCnpj());
            statement.setString(6, pessoa.getRazaoSocial());
            statement.setInt(2, pessoa.getIdJuridica());
            statement.executeUpdate();
            connection.commit();
            System.out.println("Atualizado com sucesso!");


        } catch (SQLException e) {
            System.out.println("Erro ao atualizar!" +e);
            try {
                if (connection != null) {
                    connection.rollback();
                }
```

```
            } catch (SQLException rollbackEx) {
            }
        } finally {
            ConectorBD.close(statement);
            ConectorBD.close(connection);
        }
    }

 public void excluir(PessoaJuridica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;

        try {
            connection = ConectorBD.getConnection();
            connection.setAutoCommit(false);

                    statement = ConectorBD.getPrepared("DELETE FROM pessoa_Juridica WHERE
idJuridica=?");
            statement.setInt(1, pessoa.getIdJuridica());
            statement.executeUpdate();

            statement = ConectorBD.getPrepared("DELETE FROM pessoa WHERE idPessoa=?");
            statement.setInt(1, pessoa.getIdPessoa());
            statement.executeUpdate();
```

```
            connection.commit();
            System.out.println("Removido com sucesso!");
        } catch (SQLException e) {
            System.out.println("Erro ao Remover!" +e);
            try {
                if (connection != null) {
                    connection.rollback();
                }
            } catch (SQLException rollbackEx) {
            }
        } finally {
            ConectorBD.close(statement);
            ConectorBD.close(connection);
        }
    }
}
package cadastro.model.util;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class SequenceManager {
    public static int getValue(String sequenceName) {
        Connection connection = null;
        PreparedStatement statement = null;
```

```
        ResultSet resultSet = null;
        try {
            connection = ConectorBD.getConnection();
            statement = ConectorBD.getPrepared("SELECT NEXT VALUE FOR " + sequenceName);
            resultSet = statement.executeQuery();
            if (resultSet.next()) {
                return resultSet.getInt(1);
            }
            return -1;
        } catch (SQLException e) {
            System.out.println(e);
            return -1;
        } finally {
            ConectorBD.close(resultSet);
            ConectorBD.close(statement);
            ConectorBD.close(connection);
        }
    }
}
package cadastrobd;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridicaDAO;
public class CadastroBDTeste {
```

```
public static void main(String[] args) {
   PessoaFisica pessoaFisica = new PessoaFisica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idFisica, cpf, dt_nasc);
pessoaFisica.setNome("João da Silva");
pessoaFisica.setCidade("Cidade A");
pessoaFisica.setEndereco("Endereço A");
pessoaFisica.setUf("UF A");
pessoaFisica.setTelefone(123456789);
pessoaFisica.setTipoPessoa("PF");
pessoaFisica.setIdFisica(90);
pessoaFisica.setCpf(1234567890);
pessoaFisica.setDtNasc("05/12/1995");
PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

pessoaFisicaDAO.incluir(pessoaFisica);

pessoaFisica.setNome("Novo Nome Pessoa Física");
pessoaFisicaDAO.alterar(pessoaFisica);

System.out.println("Pessoas Físicas no banco de dados:");
for (PessoaFisica pf : pessoaFisicaDAO.getPessoas()) {
   System.out.println(pf);
}

pessoaFisicaDAO.excluir(pessoaFisica);
```

```
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        pessoaJuridica.setRazaoSocial("Empresa ABC");
        pessoaJuridica.setCnpj(1234567890);
        pessoaJuridica.setCidade("Cidade B");
        pessoaJuridica.setEndereco("Endereço B");
        pessoaJuridica.setUf("UF B");
        pessoaJuridica.setTelefone(987654321);
        pessoaJuridica.setTipoPessoa("PJ");

        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
        pessoaJuridicaDAO.incluir(pessoaJuridica);

        pessoaJuridica.setNome("Novo Nome Pessoa Jurídica");
        pessoaJuridicaDAO.alterar(pessoaJuridica);

        System.out.println("Pessoas Jurídicas no banco de dados:");
        for (PessoaJuridica pj : pessoaJuridicaDAO.getPessoas()) {
            System.out.println(pj);
        }
        pessoaJuridicaDAO.excluir(pessoaJuridica);
    }
}
```

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

JDBC tem a capacidade de simplificar a interação com bancos de dados, garantir portabilidade, segurança e desempenho, tornando mais fácil o desenvolvimento de aplicativos e sistemas de software robustos e escaláveis.

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

O PreparedStatement é mais seguro contra injeção de SQL, mais eficiente em termos de desempenho e oferece melhor legibilidade e tratamento de tipos de dados que o Statement.

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

Separando as operações de acesso a dados, promovendo a reutilização do código, facilitando testes e isolar mudanças na camada de acesso a dados. Ele torna o software mais organizado, documentado e flexível, facilitando a manutenção futura e evolução do sistema.

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

Podemos utilizar três abordagens: "Tabela Única", onde todas as classes compartilham uma tabela com uma coluna indicando o tipo; "Tabela por Subclasse", onde cada classe tem sua própria tabela e a classe pai contém atributos comuns; e "Tabela por Agregação", onde cada classe tem sua própria tabela e todas elas duplicam os atributos da classe pai. A escolha depende das necessidades do sistema e das características das classes, e cada abordagem tem vantagens e desvantagens.

2º Procedimento | Alimentando a Base

```
package cadastrobd;


import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.Connection;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
```

```
import cadastrobd.model.PessoaJuridicaDAO;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

/**
 *
 * @author JPZanirati
 */

public class CadastroBD {
   public static void main(String[] args) {
       Connection connection = null;

       try {
           connection = ConectorBD.getConnection();
           Scanner scanner = new Scanner(System.in);
           SequenceManager sequence = new SequenceManager();

           while (true) {
               System.out.println("Opções:");
               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 - Sair");

System.out.println("Selecione uma opção:");


int opcao;

opcao = scanner.nextInt();


if (opcao == 0) {

   break;

}


switch (opcao) {

   case 1 -> {

      System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");


      int tipo = scanner.nextInt();

      int idPessoa = sequence.getValue();


      if (tipo == 1) {

         int idFisica = sequence.getValue();


         scanner.nextLine();

         System.out.print("Nome: ");

         String nome = scanner.nextLine();
```

```
System.out.print("Cidade: ");
String cidade = scanner.nextLine();
System.out.print("Endereço: ");
String endereco = scanner.nextLine();
System.out.print("UF: ");
String uf = scanner.nextLine();
System.out.print("Telefone: ");
int telefone = scanner.nextInt();
scanner.nextLine();
System.out.print("CPF: ");
long cpf = scanner.nextLong();
scanner.nextLine();
System.out.print("Tipo Pessoa: ");
String tipoPessoa = scanner.nextLine();
System.out.print("Data de nascimento: ");
String date = scanner.nextLine();

SimpleDateFormat formatter = new SimpleDateFormat(date);
Date dt_nasc = formatter.parse(date);

 PessoaFisica pessoaFisica = new PessoaFisica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idFisica, cpf, dt_nasc);

PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
pessoaFisicaDAO.incluir(pessoaFisica);
```

```
            scanner.close();


        } else if (tipo == 2) {
            int idJuridica = sequence.getValue();


            scanner.nextLine();
            System.out.print("Nome: ");
            String nome = scanner.nextLine();
            System.out.print("Cidade: ");
            String cidade = scanner.nextLine();
            System.out.print("Endereço: ");
            String endereco = scanner.nextLine();
            System.out.print("UF: ");
            String uf = scanner.nextLine();
            System.out.print("Telefone: ");
            int telefone = scanner.nextInt();
            System.out.print("Tipo Pessoa: ");
            String tipoPessoa = scanner.nextLine();
            System.out.print("Cnpj: ");
            int cnpj = scanner.nextInt();
            System.out.print("Razão Social: ");
            String razaoSocial = scanner.nextLine();
```

PessoaJuridica pessoaJuridica = new PessoaJuridica(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idJuridica, cnpj, razaoSocial);

```
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
        pessoaJuridicaDAO.incluir(pessoaJuridica);
        scanner.close();
    }
}

case 2 -> {
    System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
    int tipo = scanner.nextInt();

    if (tipo == 1) {

        scanner.nextLine();
        System.out.print("ID: ");
        int idPessoa = scanner.nextInt();
        PessoaFisica pessoaFisica = PessoaFisicaDAO.getPessoa(idPessoa);
        pessoaFisica.exibir();

        scanner.nextLine();
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Cidade: ");
        String cidade = scanner.nextLine();
        System.out.print("Endereço: ");
```

```
                String endereco = scanner.nextLine();
                System.out.print("UF: ");
                String uf = scanner.nextLine();
                System.out.print("Telefone: ");
                int telefone = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Tipo Pessoa: ");
                String tipoPessoa = scanner.nextLine();
                System.out.print("CPF: ");
                long cpf = scanner.nextLong();
                scanner.nextLine();
                System.out.print("Data de nascimento: ");
                String date = scanner.nextLine();
                SimpleDateFormat formatter = new SimpleDateFormat(date);
                Date dt_nasc = formatter.parse(date);
                    PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idFisica, cpf, dt_nasc);
                pessoaFisicaDAO.alterar(pessoaFisica);

                scanner.close();
                // Lide com exceções apropriadamente
            } else if (tipo == 2) {
                scanner.nextLine();
                System.out.print("ID: ");
                int idPessoa = scanner.nextInt();
```

```
PessoaJuridica pessoaJuridica = PessoaJuridicaDAO.getPessoa(idPessoa);
pessoaJuridica.exibir();


scanner.nextLine();
System.out.print("Nome: ");
String nome = scanner.nextLine();
System.out.print("Cidade: ");
String cidade = scanner.nextLine();
System.out.print("Endereço: ");
String endereco = scanner.nextLine();
System.out.print("UF: ");
String uf = scanner.nextLine();
System.out.print("Telefone: ");
int telefone = scanner.nextInt();
scanner.nextLine();
System.out.print("Tipo Pessoa: ");
String tipoPessoa = scanner.nextLine();
System.out.print("CNPJ: ");
long cnpj = scanner.nextLong();
scanner.nextLine();
System.out.print("Razão Social: ");
String razaoSocial = scanner.nextLine();
```

```
                PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(idPessoa, nome, cidade, endereco, uf, telefone, tipoPessoa, idJuridica, cnpj, razaoSocial);
                pessoaJuridicaDAO.alterar(pessoaJuridica);
            }
        }

        case 3 -> {
            System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
            int tipoExclusao = scanner.nextInt();

            if (tipoExclusao == 1) {

                scanner.nextLine();
                System.out.print("ID que seja excluir: ");
                int idPessoa = scanner.nextInt();

                PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
                PessoaFisica pessoaFisica = PessoaFisicaDAO.getPessoa(idPessoa);
                pessoaFisica.exibir();

                System.out.print("Tem certeza que deseja excluir? (S para Sim, N para Não): ");
                String confirm = scanner.next();

                if (confirm.equalsIgnoreCase("S")) {
```

```
            pessoaFisicaDAO.excluir(pessoaFisica);


            System.out.println("Pessoa Física excluída com sucesso.");
        } else {
        System.out.println("Exclusão cancelada.");
        }

    } else if (tipoExclusao == 2) {


        scanner.nextLine();
        System.out.print("ID que seja excluir: ");
        int idPessoa = scanner.nextInt();
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
        PessoaJuridica pessoaJuridica = PessoaJuridicaDAO.getPessoa(idPessoa);
        pessoaJuridica.exibir();
        System.out.print("Tem certeza que deseja excluir? (S para Sim, N para Não): ");
        String confirm = scanner.next();
        if (confirm.equalsIgnoreCase("S")) {
            pessoaJuridicaDAO.excluir(pessoaJuridica);
            System.out.println("Pessoa Física excluída com sucesso.");
        } else {
        System.out.println("Exclusão cancelada.");
        }
    }
}
```

```
case 4 -> {
    System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
    int tipoObter = scanner.nextInt();
    if (tipoObter == 1) {
      scanner.nextLine();
        System.out.print("ID que seja exibir: ");
        int idPessoa = scanner.nextInt();
        PessoaFisica pessoaFisica = PessoaFisicaDAO.getPessoa(idPessoa);
        pessoaFisica.exibir();
    } else if (tipoObter == 2) {
        scanner.nextLine();
        System.out.print("ID que seja exibir: ");
        int idPessoa = scanner.nextInt();
        PessoaJuridica pessoaJuridica = PessoaJuridicaDAO.getPessoa(idPessoa);
        pessoaJuridica.exibir();
    }
}
case 5 -> {
    System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
    int tipoObterTodos = scanner.nextInt();
    if (tipoObterTodos == 1) {
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        pessoaFisicaDAO.getPessoas();

    } else if (tipoObterTodos == 2) {
```

```
                    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
                    pessoaJuridicaDAO.getPessoas();
                }
            }
            default -> System.out.println("Opção inválida.");
        }
        ConectorBD.close(connection);
        scanner.close();
    }
    } catch (ParseException e) {
        System.out.println(e);
    }
  }
}
```

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

A persistência em arquivo armazena dados em formatos específicos de arquivo, é menos eficiente para consultas e requer implementações personalizadas para garantir integridade. E a persistência no banco de dados utiliza tabelas relacionais com SQL, oferecendo consultas eficientes, consistência de dados, escalabilidade, segurança, backup automatizado e gerenciamento de concorrência, tornando mais adequado para aplicações complexas e de grande escala.

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

Permitindo a expressão direta de operações em coleções, reduzindo a necessidade de loops tradicionais e variáveis temporárias, resultando em código mais limpo e eficaz.

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

Métodos acionados diretamente pelo método main em Java precisam ser marcados como static porque o main é estático e não cria instâncias da classe que o contém. Portanto, métodos estáticos podem ser chamados diretamente, enquanto métodos não estáticos pertencem a instâncias e requerem a criação de um objeto.

Conclusão

Durante esta criação, utilizei persistência de dados em bancos de dados juntamente ao java. Concentrando na criação de um código Java eficiente, boas práticas de programação, interação com bancos de dados e utilização de recursos modernos, como operadores lambda. O objetivo foi criar uma conexão entre o banco e a linguagem java para facil inserção de informações, a compreender e aplicar esses conceitos no projeto.