# Missão Prática - Nível 3 Mundo 3

**Campus : POLO COPACABANA**

**Curso : DESENVOLVIMENTO FULL STACK**

**Disciplina : RPG0016 - BackEnd sem banco não tem**

**Turma : 9003**

**Semestre: 2023.3 FLEX**

**Nome : MARCO SERGIO ALBINO VITTORIO BAROZZI**

## 1º Procedimento | Mapeamento Objeto-Relacional e DAO

### Objetivos:

1 - Implementar persistência com base no middleware

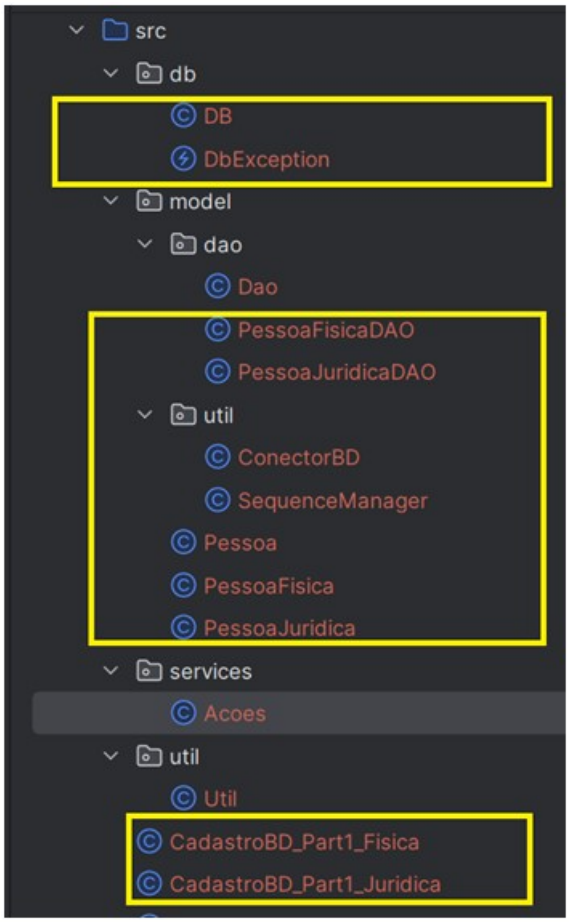
2 - Utilizar o padrão DAO (Data Access Object) no m
dados.

3 - Implementar o mapeamento objeto-relacional em s
Java.

4 - Criar sistemas cadastrais com persistência em b
relacional.

## Códigos desenvolvidos :

*Classes*




*Classe CadastroBD_Part1_Fisica.java*

```
import model.PessoaFisica;
import model.dao.PessoaFisicaDAO;

import java.util.List;

public class CadastroBD_Part1_Fisica {
    public static void main(String[] args) {
       missoPraticaPessoaFisica();

    }

    private static void missoPraticaPessoaFisica(){
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        Boolean ret=false;
```

```
        List<PessoaFisica> lstPessaoFisica ;
         /*
         a. Instanciar uma pessoa física e persistir no banco de dados.
         */


        System.out.println("");
        System.out.println("");

        System.out.println("a. Instanciar uma pessoa física e persistir");

        System.out.println("");

        PessoaFisica pfNw = new PessoaFisica();
        pfNw.setNome("Virgulino da Silva");
        pfNw.setCpf("04392077039");
        pfNw.setEstado("RJ");
        pfNw.setCidade("Rio de Janeiro");
        pfNw.setLogradouro("Av Atlantica em frente a praia, s/n");
        pfNw.setTelefone("(21)4499444");
        pfNw.setEmail("virgulino@gmail.com");

        ret= pessoaFisicaDAO.incluir(pfNw);

        if(ret){
            System.out.println("Pessoa Fisica incluida com sucesso");
            System.out.println("");
            pfNw =pessoaFisicaDAO.getPessoa("Virgulino");
            System.out.println(pfNw.exibir());
        }

        //b. Alterar os dados da pessoa física no banco.
        System.out.println("");
        System.out.println("b. Alterar os dados da pessoa fisica");
        pfNw.setLogradouro("Av Vieira souto , fundos, s/n");
        pfNw.setTelefone("(21)9999884");
        ret= pessoaFisicaDAO.alterar(pfNw);
        System.out.println("");
        if (ret){
            System.out.println("Pessoa fisica alterada com sucesso !");
            System.out.println("");

            PessoaFisica pf1= pessoaFisicaDAO.getPessoa("Virgulino");
            System.out.println("-------");
            System.out.println(pf1.exibir());
            System.out.println("-------");

        }else{
            System.out.println("Pessoa fisica não pode ser alterada");
        }

        /* c.Consultar todas as pessoas físicas do banco de dados */
        System.out.println("");
        System.out.println("-------");
        System.out.println("c. Consultar todas as pessoas físicas");
        lstPessaoFisica = pessoaFisicaDAO.getPessoas();
        for ( PessoaFisica p :lstPessaoFisica) {
```

```
            System.out.println(p.exibir());
        }
        System.out.println("-------");

        //d. Excluir a pessoa física criada anteriormente no banco
        PessoaFisica pf2 =pessoaFisicaDAO.getPessoa(pfNw.getId());
        if(pf2==null){
            System.out.println("Pessoa fisica não encontrada");
        }else{
            ret=pessoaFisicaDAO.excluir(pf2.getId());
            if (ret){
                System.out.println("");
                System.out.println("Pessoa fisica removida com sucesso !");
                lstPessaoFisica =null;
                lstPessaoFisica = pessoaFisicaDAO.getPessoas();
                for ( PessoaFisica p :lstPessaoFisica) {
                    System.out.println("-------");
                    System.out.println(p.exibir());
                }
            }else{
                System.out.println("Pessoa fisica não pode ser removida");
            }
            System.out.println("");

        }


    }

}
```

## *Classe CadastroBD_Part1_Juridica.java*

```
import model.PessoaJuridica;
import model.dao.PessoaJuridicaDAO;

import java.util.List;

public class CadastroBD_Part1_Juridica {
    public static void main(String[] args) {

       missoPraticaPessoaJuridica();
    }


    private static void missoPraticaPessoaJuridica(){
        PessoaJuridicaDAO  pessoaJuridicaDAO  = new PessoaJuridicaDAO ();
        Boolean ret=false;
        List<PessoaJuridica> lstPessoaJuridica ;
         /*
         a. Instanciar uma pessoa juridica e persistir no banco de dados.
```

```
 */


System.out.println("");
System.out.println("");

System.out.println("a. Instanciar uma pessoa juridica e persistir");

System.out.println("");

PessoaJuridica pjNw  = new PessoaJuridica();
pjNw .setNome("Comercio e importação S/A");
pjNw .setCnpj("30002645999194");
pjNw .setEstado("SP");
pjNw .setCidade("São Paulo");
pjNw .setLogradouro("Av Paulista em Manhattan Bank, s/n");
pjNw .setTelefone("(11)2746222");
pjNw .setEmail("contato@imporcom.com");

ret= pessoaJuridicaDAO.incluir(pjNw);

if(ret){
    System.out.println("Pessoa juridica incluida com sucesso");
    System.out.println("");
    pjNw  =pessoaJuridicaDAO .getPessoa("Com");
    System.out.println(pjNw.exibir());
}

//b. Alterar os dados da pessoa juridica no banco.
System.out.println("");
System.out.println("b. Alterar os dados da Pessoa juridica");
pjNw .setLogradouro("Av Rio Branco , centro rj, s/n");
pjNw .setCidade("Rio de Janeiro");
pjNw .setEstado("RJ");
pjNw .setTelefone("(21)2445797");
ret= pessoaJuridicaDAO.alterar(pjNw);
System.out.println("");
if (ret){
    System.out.println("Pessoa juridica alterada com sucesso !");
    System.out.println("");
    PessoaJuridica pj1= pessoaJuridicaDAO.getPessoa("Com");
    System.out.println("-------");
    System.out.println(pj1.exibir());
    System.out.println("-------");
}else{
    System.out.println("Pessoa juridica não pode ser removida");
}

/* c.Consultar todas as pessoas juridicas do banco de dados */
System.out.println("");
System.out.println("c. Consultar todas as pessoas Juridicas");



lstPessoaJuridica = pessoaJuridicaDAO .getPessoas();

for ( PessoaJuridica p :lstPessoaJuridica) {
```

```
            System.out.println("-------");
            System.out.println(p.exibir());
        }

        //d. Excluir a pessoa juridica criada anteriormente no banco
        PessoaJuridica pj2 =pessoaJuridicaDAO .getPessoa(pjNw .getId());
        if(pj2==null){
            System.out.println("Pessoa juridica não encontrada");
        }else{
            ret=pessoaJuridicaDAO .excluir(pj2.getId());
            if (ret){
                System.out.println("");
                System.out.println("Pessoa juridica removida com sucesso !");

                lstPessoaJuridica = pessoaJuridicaDAO .getPessoas();
                for ( PessoaJuridica p :lstPessoaJuridica) {
                    System.out.println("-------");
                    System.out.println(p.exibir());
                }
            }else{
                System.out.println("Pessoa juridica não pode ser removida");
            }
            System.out.println("");

        }


    }
}
```

*Classe Pessao.java*

```
package model;

public class Pessoa {
    private   Integer id =0;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;
    public Pessoa() {

    }

    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;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer 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;
    }

    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;
    }


    public String exibir() {
        return "ID: " + id + "\n" +
                "Nome: " + nome + "\n" +
                "Logradouro: " + logradouro + "\n" +
                "Cidade: " + cidade + "\n" +
                "Estado: " + estado + "\n" +
                "Telefone: " + telefone + "\n" +
                "Email: " + email;
    }

}
```

*Classe PessaoFisica.java*

```
package model.dao;
import model.PessoaFisica;

import model.util.ConectorBD;
import model.util.SequenceManager;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;



public class PessoaFisicaDAO {

    private ConectorBD conectorBD;

    public PessoaFisicaDAO() {
        conectorBD = new ConectorBD(); // conexões aqui
    }

    public PessoaFisica getPessoa(int id) {
        PessoaFisica pessoaFisica = null;
        String sql = "SELECT p.*, pf.cpf FROM PessoaFisica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa WHERE p.id = ?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql)) {
            stmt.setInt(1, id);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    pessoaFisica = new PessoaFisica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                            rs.getString("cidade"), rs.getString("estado"),
```

```
rs.getString("telefone"),
                            rs.getString("email"), rs.getString("cpf"));
                }
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return pessoaFisica;
    }

    public PessoaFisica getPessoa(String nome) {
        PessoaFisica pessoaFisica = null;
        String sql = "SELECT p.*, pf.cpf FROM PessoaFisica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa WHERE p.nome like ?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql)) {
            stmt.setString(1, nome + "%");
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    pessoaFisica = new PessoaFisica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                            rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                            rs.getString("email"), rs.getString("cpf"));
                }
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return pessoaFisica;
    }

    public List<PessoaFisica> getPessoas(){
        List<PessoaFisica> lstPessoaFisica = new ArrayList<>();
        String sql = "SELECT p.*, pf.cpf FROM PessoaFisica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa ORDER BY p.nome";

        try (
                PreparedStatement stmt = conectorBD.getPrepared(sql);
                ResultSet rs = stmt.executeQuery()
        ) {
            while (rs.next()) {
                lstPessoaFisica.add(new PessoaFisica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                        rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                        rs.getString("email"), rs.getString("cpf")));
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
```

```
        conectorBD.closeConnection();
    }

        return lstPessoaFisica;
    }
    SequenceManager sc= new SequenceManager();

    public boolean incluir(PessoaFisica pessoa) {
        boolean result = false;
        String sql = "INSERT INTO Pessoa (id, nome, logradouro, cidade,
estado, telefone, email,tipo) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        String sqlPessoaFisica = "INSERT INTO PessoaFisica (idpessoa, cpf)
VALUES (?, ?)";

        int idNext = sc.getValue("PessoaIdSeq");

        try (PreparedStatement stmt = conectorBD.getPrepared(sql);
             PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaFisica)) {

            stmt.setInt(1, idNext);
            stmt.setString(2, pessoa.getNome());
            stmt.setString(3, pessoa.getLogradouro());
            stmt.setString(4, pessoa.getCidade());
            stmt.setString(5, pessoa.getEstado());
            stmt.setString(6, pessoa.getTelefone());
            stmt.setString(7, pessoa.getEmail());
            stmt.setString(8, "F");
            stmtPf.setInt(1, idNext);
            stmtPf.setString(2, pessoa.getCpf());

            int r1=stmt.executeUpdate();
            int r2=stmtPf.executeUpdate();

            if(r1 > 0 && r2 > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }

    public boolean alterar(PessoaFisica pessoa) {
        boolean result = false;
        String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?,
estado=?, telefone=?, email=? WHERE id=?";
        String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf=? WHERE
idpessoa=?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql);
             PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaFisica)) {
```

```
            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());


            stmtPf.setString(1, pessoa.getCpf());
            stmtPf.setInt(2, pessoa.getId());

            if(stmt.executeUpdate() > 0 && stmtPf.executeUpdate() > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }

    public boolean excluir(int id) {
        boolean result = false;
        String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE idpessoa=?";
        String sql = "DELETE FROM Pessoa WHERE id=?";

        try (PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaFisica);
             PreparedStatement stmt = conectorBD.getPrepared(sql)) {

            stmtPf.setInt(1, id);
            stmt.setInt(1, id);

            // Deletando primeiro da tabela PessoaFisica por causa da
restrição de chave estrangeira.
            if(stmtPf.executeUpdate() > 0 && stmt.executeUpdate() > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }
    private void handleSQLException(SQLException e) {
        // Handle exceptions in a more sophisticated way, maybe logging and
rethrowing
        e.printStackTrace();
    }
```

```
}
```

*Classe PessoaFisicaDao.java*

```
package model.dao;
import model.PessoaFisica;

import model.util.ConectorBD;
import model.util.SequenceManager;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;



public class PessoaFisicaDAO {

    private ConectorBD conectorBD;

    public PessoaFisicaDAO() {
        conectorBD = new ConectorBD(); // conexões aqui
    }

    public PessoaFisica getPessoa(int id) {
        PessoaFisica pessoaFisica = null;
        String sql = "SELECT p.*, pf.cpf FROM PessoaFisica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa WHERE p.id = ?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql)) {
            stmt.setInt(1, id);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    pessoaFisica = new PessoaFisica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                            rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                            rs.getString("email"), rs.getString("cpf"));
                }
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return pessoaFisica;
    }
```

```
    public PessoaFisica getPessoa(String nome) {
        PessoaFisica pessoaFisica = null;
        String sql = "SELECT p.*, pf.cpf FROM PessoaFisica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa WHERE p.nome like ?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql)) {
            stmt.setString(1, nome + "%");
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    pessoaFisica = new PessoaFisica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                            rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                            rs.getString("email"), rs.getString("cpf"));
                }
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return pessoaFisica;
    }

    public List<PessoaFisica> getPessoas(){
        List<PessoaFisica> lstPessoaFisica = new ArrayList<>();
        String sql = "SELECT p.*, pf.cpf FROM PessoaFisica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa ORDER BY p.nome";

        try (
                PreparedStatement stmt = conectorBD.getPrepared(sql);
                ResultSet rs = stmt.executeQuery()
        ) {
            while (rs.next()) {
                lstPessoaFisica.add(new PessoaFisica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                        rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                        rs.getString("email"), rs.getString("cpf")));
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
        conectorBD.closeConnection();
    }

        return lstPessoaFisica;
    }
    SequenceManager sc= new SequenceManager();

    public boolean incluir(PessoaFisica pessoa) {
        boolean result = false;
        String sql = "INSERT INTO Pessoa (id, nome, logradouro, cidade,
estado, telefone, email,tipo) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        String sqlPessoaFisica = "INSERT INTO PessoaFisica (idpessoa, cpf)
VALUES (?, ?)";
```

```
        int idNext = sc.getValue("PessoaIdSeq");

        try (PreparedStatement stmt = conectorBD.getPrepared(sql);
             PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaFisica)) {

            stmt.setInt(1, idNext);
            stmt.setString(2, pessoa.getNome());
            stmt.setString(3, pessoa.getLogradouro());
            stmt.setString(4, pessoa.getCidade());
            stmt.setString(5, pessoa.getEstado());
            stmt.setString(6, pessoa.getTelefone());
            stmt.setString(7, pessoa.getEmail());
            stmt.setString(8, "F");
            stmtPf.setInt(1, idNext);
            stmtPf.setString(2, pessoa.getCpf());

            int r1=stmt.executeUpdate();
            int r2=stmtPf.executeUpdate();

            if(r1 > 0 && r2 > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }

    public boolean alterar(PessoaFisica pessoa) {
        boolean result = false;
        String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?,
estado=?, telefone=?, email=? WHERE id=?";
        String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf=? WHERE
idpessoa=?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql);
             PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaFisica)) {

            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());


            stmtPf.setString(1, pessoa.getCpf());
            stmtPf.setInt(2, pessoa.getId());
```

```
            if(stmt.executeUpdate() > 0 && stmtPf.executeUpdate() > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }

    public boolean excluir(int id) {
        boolean result = false;
        String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE idpessoa=?";
        String sql = "DELETE FROM Pessoa WHERE id=?";

        try (PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaFisica);
             PreparedStatement stmt = conectorBD.getPrepared(sql)) {

            stmtPf.setInt(1, id);
            stmt.setInt(1, id);

            // Deletando primeiro da tabela PessoaFisica por causa da
restrição de chave estrangeira.
            if(stmtPf.executeUpdate() > 0 && stmt.executeUpdate() > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }
    private void handleSQLException(SQLException e) {
        // Handle exceptions in a more sophisticated way, maybe logging and
rethrowing
        e.printStackTrace();
    }



}
```

*Classe PessoaJuridicaDao.java*

```
package model.dao;

import model.Pessoa;
import model.PessoaJuridica;
```

```
import model.util.ConectorBD;
import model.util.SequenceManager;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;


public class PessoaJuridicaDAO {

    private ConectorBD conectorBD;

    public PessoaJuridicaDAO() {
        conectorBD = new ConectorBD(); // considerar usar um pool de conexões
aqui
    }

    public PessoaJuridica getPessoa(int id) {
        PessoaJuridica PessoaJuridica = null;
        String sql = "SELECT p.*, pf.cnpj FROM PessoaJuridica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa WHERE p.id = ?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql)) {
            stmt.setInt(1, id);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    PessoaJuridica = new PessoaJuridica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                            rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                            rs.getString("email"), rs.getString("cnpj"));
                }
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return PessoaJuridica;
    }

    public PessoaJuridica getPessoa(String nome) {
        PessoaJuridica PessoaJuridica = null;
        String sql = "SELECT p.*, pf.cnpj FROM PessoaJuridica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa WHERE p.nome like ?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql)) {
            stmt.setString(1, nome + "%");
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    PessoaJuridica = new PessoaJuridica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                            rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
```

```
                            rs.getString("email"), rs.getString("cnpj"));
                }
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return PessoaJuridica;
    }

    public List<PessoaJuridica> getPessoas(){
        List<PessoaJuridica> lstPessoaJuridica = new ArrayList<>();
        String sql = "SELECT p.*, pf.cnpj FROM PessoaJuridica pf INNER JOIN
Pessoa p ON p.id = pf.idpessoa ORDER BY p.nome";

        try (
                PreparedStatement stmt = conectorBD.getPrepared(sql);
                ResultSet rs = stmt.executeQuery()
        ) {
            while (rs.next()) {
                lstPessoaJuridica.add(new PessoaJuridica(rs.getInt("id"),
rs.getString("nome"), rs.getString("logradouro"),
                        rs.getString("cidade"), rs.getString("estado"),
rs.getString("telefone"),
                        rs.getString("email"), rs.getString("cnpj")));
            }
        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return lstPessoaJuridica;
    }
    SequenceManager sc= new SequenceManager();

    public boolean incluir(PessoaJuridica pessoa) {
        boolean result = false;
        String sql = "INSERT INTO Pessoa (id, nome, logradouro, cidade,
estado, telefone, email,tipo) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idpessoa,
cnpj) VALUES (?, ?)";

        int idNext = sc.getValue("PessoaIdSeq");

        try (PreparedStatement stmt = conectorBD.getPrepared(sql);
             PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaJuridica)) {

            stmt.setInt(1, idNext);
            stmt.setString(2, pessoa.getNome());
            stmt.setString(3, pessoa.getLogradouro());
            stmt.setString(4, pessoa.getCidade());
            stmt.setString(5, pessoa.getEstado());
            stmt.setString(6, pessoa.getTelefone());
```

```
            stmt.setString(7, pessoa.getEmail());
            stmt.setString(8, "F");
            stmtPf.setInt(1, idNext);
            stmtPf.setString(2, pessoa.getCnpj());

            int r1=stmt.executeUpdate();
            int r2=stmtPf.executeUpdate();

            if(r1 > 0 && r2 > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }

    public boolean alterar(PessoaJuridica pessoa) {
        boolean result = false;
        String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?,
estado=?, telefone=?, email=? WHERE id=?";
        String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj=? WHERE
idpessoa=?";

        try (PreparedStatement stmt = conectorBD.getPrepared(sql);
             PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaJuridica)) {

            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());


            stmtPf.setString(1, pessoa.getCnpj());
            stmtPf.setInt(2, pessoa.getId());

            if(stmt.executeUpdate() > 0 && stmtPf.executeUpdate() > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }
```

```
    public boolean excluir(int id) {
        boolean result = false;
        String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE
idpessoa=?";
        String sql = "DELETE FROM Pessoa WHERE id=?";

        try (PreparedStatement stmtPf =
conectorBD.getPrepared(sqlPessoaJuridica);
             PreparedStatement stmt = conectorBD.getPrepared(sql)) {

            stmtPf.setInt(1, id);
            stmt.setInt(1, id);

            // Deletando primeiro da tabela PessoaJuridica por causa da
restrição de chave estrangeira.
            if(stmtPf.executeUpdate() > 0 && stmt.executeUpdate() > 0) {
                result = true;
            }

        } catch (SQLException e) {
            handleSQLException(e);
        } finally {
            conectorBD.closeConnection();
        }

        return result;
    }
    private void handleSQLException(SQLException e) {
        // Handle exceptions in a more sophisticated way, maybe logging and
rethrowing
        e.printStackTrace();
    }

}
```

*Classe DB.java*

```
package db;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.*;
import java.util.Properties;

public class DB {

    private static Connection conn = null;

    public static Connection getConnection() {
        if (conn == null) {
            try {
                Properties props = loadProperties();
                String url = props.getProperty("dburl");
                conn = DriverManager.getConnection(url, props);
```

```
            } catch (SQLException e) {
                throw new DbException(e.getMessage());
            }
        }
        return conn;
    }

    public static void closeConnection() {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                throw new DbException(e.getMessage());
            }
        }
    }

    private static Properties loadProperties() {
        try (FileInputStream fs = new FileInputStream("db.properties")) {
            Properties props = new Properties();
            props.load(fs);
            return props;
        } catch (IOException e) {
            throw new DbException(e.getMessage());
        }
    }

    public static void closeResultSet(ResultSet rs) {
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
                throw new DbException(e.getMessage());
            }
        }
    }

    public static void closeStatement(Statement st) {
        if (st != null) {
            try {
                st.close();
            } catch (SQLException e) {
                throw new DbException(e.getMessage());
            }
        }
    }
}
```

*arquivo db.properties*

```
dburl=jdbc:sqlserver://localhost:1434;databaseName=loja;user=loja;password=lo
ja;trustServerCertificate=true
encrypt=true
```

*Classe DB.Exception.java*

```
package db;

public class DbException extends RuntimeException {
    private static final long serialVersionUID = 1L;

    public DbException(String msg) {
        super(msg);
    }
}
```

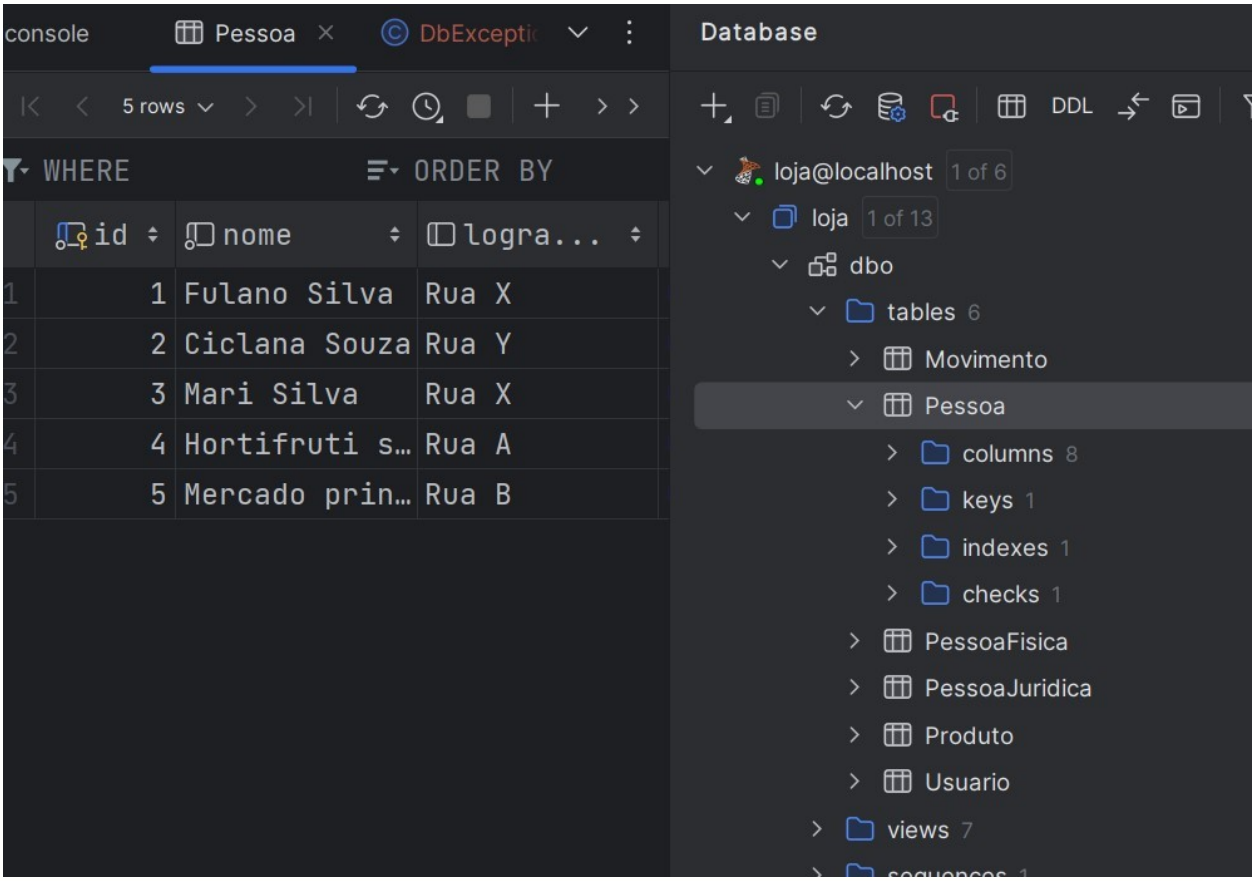
*Classe SequenceManager.java*

```
package model.util;

 ;
import java.sql.ResultSet;
import java.sql.SQLException;

public class SequenceManager {

    private  ConectorBD conectorBD= new ConectorBD(); // considerar usar um
pool de conexões aqui;
    /*
    public static int getValue2(String sequenceName) {

        int nextValue = -1;
        String sql = "SELECT NEXT VALUE FOR " + sequenceName; // Este SQL
dependerá da sintaxe do banco de dados
        try (ResultSet resultSet = conectorBD.getSelect(sql)) {
            if (resultSet.next()) {
                nextValue = resultSet.getInt(1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return nextValue;
    }
    */
    public int getValue(String sequenceName) {
        int nextValue = -1;
        String sql = "SELECT NEXT VALUE FOR " + sequenceName; // Este SQL
dependerá da sintaxe do banco de dados

        try (ResultSet resultSet = conectorBD.getSelect(sql)) {
            if (resultSet.next()) {
                nextValue = resultSet.getInt(1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            conectorBD.closeConnection();
```

```
        }
        return nextValue;
    }

}
```

## Resultados da execução dos códigos

Verificando a conexão ao banco



1 - Instanciar uma pessoa física e persistir no banco de dados.

```
PessoaFisica pfNw = new PessoaFisica();
pfNw.setNome("Virgulino da Silva");
pfNw.setCpf("04392077039");
pfNw.setEstado("RJ");
pfNw.setCidade("Rio de Janeiro");
pfNw.setLogradouro("Av Atlantica em frente a praia, s/n");
pfNw.setTelefone("(21)4499444");
pfNw.setEmail("virgulino@gmail.com");

ret= pessoaFisicaDAO.incluir(pfNw);
```

```
Pessoa Fisica incluida com sucesso

ID: 39
Nome: Virgulino da Silva
Logradouro: Av Atlantica em frente a praia, s/n
Cidade: Rio de Janeiro
Estado: RJ
Telefone: (21)4499444
Email: virgulino@gmail.com
CPF: 04392077039

b. Alterar os dados da pessoa fisica

Pessoa fisica alterada com sucesso !
```

2 - Alterar os dados da pessoa física no banco.
Consultar todas as pessoas físicas do banco de dados e listar no console.

```
//b. Alterar os dados da pessoa física no banco.
System.out.println("");
System.out.println("b. Alterar os dados da pessoa fisica");
pfNw.setLogradouro("Av Vieira souto , fundos, s/n");
pfNw.setTelefone("(21)9999884");
ret= pessoaFisicaDAO.alterar(pfNw);
System.out.println("");
if (ret){
    System.out.println("Pessoa fisica alterada com sucesso !");
```

Lista abaixo

```
-------
c. Consultar todas as pessoas físicas
ID: 2
Nome: Ciclana Souza
Logradouro: Rua Y
Cidade: Cidade Y
Estado: SP
Telefone: (21)4444444
Email: ciclana@example.com
CPF: 66203143049
ID: 1
Nome: Fulano Silva
Logradouro: Rua X
Cidade: Cidade X
Estado: MG
Telefone: (21)3333333
Email: fulano@example.com
CPF: 69418874083
ID: 3
Nome: Mari Silva
Logradouro: Rua X
Cidade: Cidade T
Estado: SP
Telefone: (21)4445675
Email: mari@example.com
CPF: 65241594043
ID: 39
Nome: Virgulino da Silva
Logradouro: Av Vieira souto , fundos, s/n
Cidade: Rio de Janeiro
Estado: RJ
Telefone: (21)9999884
Email: virgulino@gmail.com
CPF: 04392077039
```

3 - Excluir a pessoa física criada anteriormente no banco. Instanciar uma pessoa jurídica e persistir no banco de dados.

```
//d. Excluir a pessoa física criada anteriormente no banco
PessoaFisica pf2 =pessoaFisicaDAO.getPessoa( nome: "Virgulino");
if(pf2==null){
    System.out.println("Pessoa fisica não encontrada");
}else{
    ret=pessoaFisicaDAO.excluir(pf2.getId());
    if (ret){
        System.out.println("");
        System.out.println("Pessoa fisica removida com sucesso !");
```

Lista PFs após exclusão

```
Pessoa fisica removida com sucess
-------
ID: 2
Nome: Ciclana Souza
Logradouro: Rua Y
Cidade: Cidade Y
Estado: SP
Telefone: (21)4444444
Email: ciclana@example.com
CPF: 66203143049
-------
ID: 1
Nome: Fulano Silva
Logradouro: Rua X
Cidade: Cidade X
Estado: MG
Telefone: (21)3333333
Email: fulano@example.com
CPF: 69418874083
-------
ID: 3
Nome: Mari Silva
Logradouro: Rua X
Cidade: Cidade T
Estado: SP
Telefone: (21)4445675
Email: mari@example.com
CPF: 65241594043
```

Instanciar e persistir pj

```
PessoaJuridica pjNw  = new PessoaJuridica();
pjNw .setNome("Comercio e importação S/A");
pjNw .setCnpj("30002645999194");
pjNw .setEstado("SP");
pjNw .setCidade("São Paulo");
pjNw .setLogradouro("Av Paulista em Manhattan Bank, s/n");
pjNw .setTelefone("(11)2746222");
pjNw .setEmail("contato@imporcom.com");

ret= pessoaJuridicaDAO.incluir(pjNw);
```

```
a. Instanciar uma pessoa juridica e persistir

Pessoa juridica incluida com sucesso

ID: 29
Nome: Comercio e importação S/A
Logradouro: Av Paulista em Manhattan Bank, s/n
Cidade: São Paulo
Estado: SP
Telefone: (11)2746222
Email: contato@imporcom.com
CNPJ: 30002645999194
```

4 - Alterar os dados da pessoa jurídica no banco.
Consultar todas as pessoas jurídicas do banco e listar no console.

```
//b. Alterar os dados da pessoa juridica no banco.
System.out.println("");
System.out.println("b. Alterar os dados da Pessoa j
pjNw .setLogradouro("Av Rio Branco , centro rj, s/n
pjNw .setCidade("Rio de Janeiro");
pjNw .setEstado("RJ");
```

Segue a lista

```
Pessoa juridica alterada com sucesso !

-------
ID: 29
Nome: Comercio e importação S/A
Logradouro: Av Rio Branco , centro rj, s/n
Cidade: Rio de Janeiro
Estado: RJ
Telefone: (21)2445797
Email: contato@imporcom.com
CNPJ: 38002645999194
-------

c. Consultar todas as pessoas Juridicas
-------
ID: 29
Nome: Comercio e importação S/A
Logradouro: Av Rio Branco , centro rj, s/n
Cidade: Rio de Janeiro
Estado: RJ
Telefone: (21)2445797
Email: contato@imporcom.com
CNPJ: 38002645999194
-------
ID: 4
Nome: Hortifruti sucesso
Logradouro: Rua A
Cidade: Cidade A
Estado: SP
Telefone: (11)1111111
Email: empresaA@example.com
CNPJ: 30333306000120
-------
ID: 5
Nome: Mercado principal
Logradouro: Rua B
Cidade: Cidade B
Estado: RJ
Telefone: (11)2222222
Email: empresaB@example.com
CNPJ: 33102645000194
```

5-Excluir a pessoa jurídica criada anteriormente no banco.

```java
//d. Excluir a pessoa juridica criada anteriormente no banco
PessoaJuridica pj2 =pessoaJuridicaDAO .getPessoa( nome: "Com");
if(pj2==null){
    System.out.println("Pessoa juridica não encontrada");
}else{
    ret=pessoaJuridicaDAO .excluir(pj2.getId());
    if (ret){
        System.out.println("");
        System.out.println("Pessoa juridica removida com sucesso !");
```

```
Pessoa juridica removida com sucesso !
-------
ID: 4
Nome: Hortifruti sucesso
Logradouro: Rua A
Cidade: Cidade A
Estado: SP
Telefone: (11)1111111
Email: empresaA@example.com
CNPJ: 30333306000120
-------
ID: 5
Nome: Mercado principal
Logradouro: Rua B
Cidade: Cidade B
Estado: RJ
Telefone: (11)2222222
Email: empresaB@example.com
CNPJ: 33102645000194
```

## Análise e Conclusão:

***1 - Qual a importância dos componentes de middleware,***

***como o JDBC?***

r) Eles facilitam a comunicação e o gerenciamento de dados entre sistemas heterogêneos como por exemplo o JDBC que fornece uma interface para conectar aplicações Java a bancos de dados relacionais.

Sua importancia esta em :

***Padronização e abstração*** : trabalham com uma interface padrão

***Interoperabilidade*** : ajuda a diferentes sistemmas a trabalharem juntos

***Flexibilidade*** : devido a padronização e troca é simples

***Produtividade*** : o desenvolvedor apenas incorpora a solução

***Escalabilidade e Performance*** : se o middleware for um servidor de aplicação terá recursos para balanceamento de carga,pooling de conexões e otimização, o que pode melhorar a performance e permitir que aplicações escalem para atender a demandas crescentes.

***2 - Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?***

***A execução*** : no caso do "Statement" a cada chamada para execução, ou seja a query receba novos parametros,esta será recompilda pelo banco . No caso do "PreparedStatement" a query e compilada apenas uma vez mesmo que os parametros sejam alterados em outra execução.

**Proteção ao ataque de SQL injection:** No caso do "Statement" a query e concatenada aos valores de seus parãmetros e por isso esta mais propenso a ataques do tipo "SQL injection". No caso do "*PreparedStatement*" isso não ocorre devido aos valores não serem tratados como parte do sql mais sim como se fossem "parãmetros de uma função"

**Performance e Flexibilidad**e: O "PreparedStatement" é indicado para os casos em  onde o SQL precisa ser gerado dinamicamente e a estrutura da consulta pode mudar frequentemente.

***3 - Como o padrão DAO melhora a manutenibilidade do  software?***

**Separação de Responsabilidades** :  Ao usar o padrão DAO, a lógica de acesso a dados é separada da lógica de negócios.

**Flexibilidade** : Sendo uma camada isolada, no caso de mudança de banco de dados se torna mais facil devido as alterações seram apenas na camada DAO

**Reutilização de Código** :Por se tratar de um acesso encapsulado em objetos, esses podem ser reutilizados em diferentes partes da aplicação ou mesmo em diferentes aplicações

**Testabilidade** :Ao utilizar o acesso a dados em objetos DAO, você pode mais facilmente criar mockups desses objetos para testes unitários, permitindo testar a lógica de negócios sem a necessidade de um banco de dados real

**Isolamento** : Como referido no item flexibilidade,O padrão DAO pode abstrair detalhes específicos da tecnologia usada para o armazenamento de dados fazendo que os metodos disponibilizados no codigo sejam sempre padronizados