

Aluno: Altair Silva Filho

Matrícula: 202403254409

Unidade: Palhoça – SC

Curso: Desenvolvimento Full Stack

Disciplina: Vamos manter as informações?

Período: 2025.1

## ***Relatório da Missão Prática – Nível 3 – Mundo 3***

### **Back-End Sem Banco Não Tem**

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

##### **Objetivo**

O projeto visa criar um projeto na plataforma Java, usando a IDE NetBeans, com acesso a Banco de Dados SQL através dos componentes do JDBC.

Neste procedimento, o objetivo é criar o mapeamento Objeto-Relacional com as classes para conexão com o banco e instanciação dos demais objetos no padrão DAO.

##### **Códigos do Projeto**

###### **Pacote cadastrobd.model**

*Pessoa.java*

```
package cadastrobd.model;
```

```
public class Pessoa {  
    private int id;  
    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 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 void setCidade(String cidade) {
    this.cidade = cidade;
}

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

    public String getEmail() {
        return email;
    }

    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("E-mail: " + email);
    }
}

```

*PessoaFisica.java*

```

package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    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;
    }

    public String getCpf() {
        return cpf;
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }
}

```

### *PessoaJuridica.java*

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    private String cnpj ;

    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;
    }

    public String getCnpj() {
        return cnpj;
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

### *PessoaFisicaDAO.java*

```
package cadastrobd.model;

import java.sql.*;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {
    public PessoaFisica getPessoa(int id) throws Exception {
        Connection c1 = ConectorBD.getConnection ();
        String sql = "SELECT * FROM Pessoa p JOIN PessoaFisica pf ON p.idPessoa = pf.idPessoaFisica
WHERE p.idPessoa = ?";

        PreparedStatement ps = ConectorBD.getPrepared(c1,sql);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();

        PessoaFisica pf = null;
        while (rs.next()) {
            pf = new PessoaFisica (
                rs.getInt("idPessoa"),
```

```

        rs.getString("nome_razaoSocial"),
        rs.getString("logradouro"),
        rs.getString("cidade"),
        rs.getString("estado"),
        rs.getString("telefone"),
        rs.getString("email"),
        rs.getString("cpf")
    );
}

rs.close();
ps.close();
c1.close();

return pf;
}

public List<PessoaFisica> getPessoas() throws Exception {
    Connection c1 = ConectorBD.getConnection ();
    String sql = "SELECT * FROM Pessoa p JOIN PessoaFisica pf ON p.idPessoa = pf.idPessoaFisica";
    ResultSet rs = ConectorBD.getSelect(c1, sql);

    List<PessoaFisica> lista = new ArrayList<>();
    while (rs.next()) {
        PessoaFisica pf = new PessoaFisica(
            rs.getInt("idPessoa"),
            rs.getString("nome_razaoSocial"),
            rs.getString("logradouro"),
            rs.getString("cidade"),
            rs.getString("estado"),
            rs.getString("telefone"),
            rs.getString("email"),
            rs.getString("cpf")
        );
        lista.add(pf);
    }

    rs.close();
    c1.close();

    return lista;
}

public void incluir(PessoaFisica pf) throws Exception {
    Connection c1 = ConectorBD.getConnection();
    String sql1 = "INSERT INTO Pessoa (idPessoa, nome_razaoSocial, logradouro, "
        + "cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
    PreparedStatement ps1 = ConectorBD.getPrepared(c1, sql1);
    pf.setId(SequenceManager.getValue("Seq_Id_TablePessoa"));

    ps1.setInt(1, pf.getId());
    ps1.setString(2, pf.getNome());

```

```

ps1.setString(3, pf.getLogradouro());
ps1.setString(4, pf.getCidade());
ps1.setString(5, pf.getEstado());
ps1.setString(6, pf.getTelefone());
ps1.setString(7, pf.getEmail());
ps1.executeUpdate();

String sql2 = "INSERT INTO PessoaFisica(idPessoaFisica, cpf) VALUES (?, ?)";
PreparedStatement ps2 = ConectorBD.getPrepared(c1, sql2);
ps2.setInt(1, pf.getId());
ps2.setString(2, pf.getCpf());
ps2.executeUpdate();

ps1.close();
ps2.close();
c1.close();
}

public void alterar (PessoaFisica pf) throws Exception {
    Connection c1 = ConectorBD.getConnection ();

    String sql1 = "UPDATE Pessoa SET nome_razaoSocial = ?, logradouro = ?, cidade = ?,"
        + " estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
    PreparedStatement ps1 = ConectorBD.getPrepared(c1, sql1);
    ps1.setString(1, pf.getNome());
    ps1.setString(2, pf.getLogradouro());
    ps1.setString(3, pf.getCidade());
    ps1.setString(4, pf.getEstado());
    ps1.setString(5, pf.getTelefone());
    ps1.setString(6, pf.getEmail());
    ps1.setInt(7, pf.getId());
    ps1.executeUpdate();

    String sql2 = "UPDATE PessoaFisica SET cpf=? WHERE idPessoaFisica=?";
    PreparedStatement ps2 = ConectorBD.getPrepared(c1, sql2);
    ps2.setString(1, pf.getCpf());
    ps2.setInt(2, pf.getId());
    ps2.executeUpdate();

    ps1.close();
    ps2.close();
    c1.close();
}

public void excluir (int id) throws Exception {
    Connection c1 = ConectorBD.getConnection ();
    String sql1 = "DELETE FROM PessoaFisica WHERE idPessoaFisica=?";
    PreparedStatement ps1 = ConectorBD.getPrepared(c1, sql1);
    ps1.setInt(1, id);
    ps1.executeUpdate();

    String sql2 = "DELETE FROM Pessoa WHERE idPessoa=?";

```

```

        PreparedStatement ps2 = ConectorBD.getPrepared(c1, sql2);
        ps2.setInt(1, id);
        ps2.executeUpdate();

        ps2.close();
        ps1.close();
        c1.close();
    }
}

```

### *PessoaJuridicaDAO.java*

```

package cadastrobd.model;
import java.sql.*;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {
    public PessoaJuridica getPessoa(int id) throws Exception {
        Connection c1 = ConectorBD.getConnection ();
        String sql = "SELECT * FROM Pessoa p JOIN PessoaJuridica pj ON p.idPessoa = pj.idPessoaJuridica WHERE p.idPessoa = ?";

        PreparedStatement ps = ConectorBD.getPrepared(c1,sql);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();

        PessoaJuridica pj = null;
        while (rs.next()) {
            pj = new PessoaJuridica (
                rs.getInt("idPessoa"),
                rs.getString("nome_razaoSocial"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
        }

        rs.close();
        ps.close();
        c1.close();

        return pj;
    }

    public List<PessoaJuridica> getPessoas() throws Exception {
        Connection c1 = ConectorBD.getConnection ();

```

```
String sql = "SELECT * FROM Pessoa p JOIN PessoaJuridica pj ON p.idPessoa =  
pj.idPessoaJuridica";
```

```
ResultSet rs = ConectorBD.getSelect(c1, sql);
```

```
List<PessoaJuridica> lista = new ArrayList<>();
```

```
while (rs.next()) {
```

```
    PessoaJuridica pj = new PessoaJuridica(
```

```
        rs.getInt("idPessoa"),
```

```
        rs.getString("nome_razaoSocial"),
```

```
        rs.getString("logradouro"),
```

```
        rs.getString("cidade"),
```

```
        rs.getString("estado"),
```

```
        rs.getString("telefone"),
```

```
        rs.getString("email"),
```

```
        rs.getString("cnpj")
```

```
    );
```

```
    lista.add(pj);
```

```
}
```

```
rs.close();
```

```
c1.close();
```

```
return lista;
```

```
}
```

```
public void incluir(PessoaJuridica pj) throws Exception {
```

```
    Connection c1 = ConectorBD.getConnection();
```

```
    String sql1 = "INSERT INTO Pessoa (idPessoa, nome_razaoSocial, logradouro, "  
        + "cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
```

```
    PreparedStatement ps1 = ConectorBD.getPrepared(c1, sql1);
```

```
    pj.setId(SequenceManager.getValue("Seq_Id_TablePessoa"));
```

```
    ps1.setInt(1, pj.getId());
```

```
    ps1.setString(2, pj.getNome());
```

```
    ps1.setString(3, pj.getLogradouro());
```

```
    ps1.setString(4, pj.getCidade());
```

```
    ps1.setString(5, pj.getEstado());
```

```
    ps1.setString(6, pj.getTelefone());
```

```
    ps1.setString(7, pj.getEmail());
```

```
    ps1.executeUpdate();
```

```
String sql2 = "INSERT INTO PessoaJuridica(idPessoaJuridica, cnpj) VALUES (?, ?)";
```

```
PreparedStatement ps2 = ConectorBD.getPrepared(c1, sql2);
```

```
ps2.setInt(1, pj.getId());
```

```
ps2.setString(2, pj.getCnpj());
```

```
ps2.executeUpdate();
```

```
ps1.close();
```

```
ps2.close();
```

```
c1.close();
```

```
}
```



```

public void alterar (PessoaJuridica pj) throws Exception {
    Connection c1 = ConectorBD.getConnection ();

    String sql1 = "UPDATE Pessoa SET nome_razaoSocial=?, logradouro=?, cidade=?, estado=?"
        + " telefone=?, email=? WHERE idPessoa=?";
    PreparedStatement ps1 = ConectorBD.getPrepared(c1, sql1);
    ps1.setString(1, pj.getNome());
    ps1.setString(2, pj.getLogradouro());
    ps1.setString(3, pj.getCidade());
    ps1.setString(4, pj.getEstado());
    ps1.setString(5, pj.getTelefone());
    ps1.setString(6, pj.getEmail());
    ps1.setInt(7, pj.getId());
    ps1.executeUpdate();

    String sql2 = "UPDATE PessoaJuridica SET cnpj=? WHERE idPessoaJuridica=?";
    PreparedStatement ps2 = ConectorBD.getPrepared(c1, sql2);
    ps2.setString(1, pj.getCnpj());
    ps2.setInt(2, pj.getId());
    ps2.executeUpdate();

    ps1.close();
    ps2.close();
    c1.close();
}

public void excluir (int id) throws Exception {
    Connection c1 = ConectorBD.getConnection ();
    String sql1 = "DELETE FROM PessoaJuridica WHERE idPessoaJuridica=?";
    PreparedStatement ps1 = ConectorBD.getPrepared(c1, sql1);
    ps1.setInt(1, id);
    ps1.executeUpdate();

    String sql2 = "DELETE FROM Pessoa WHERE idPessoa=?";
    PreparedStatement ps2 = ConectorBD.getPrepared(c1, sql2);
    ps2.setInt(1, id);
    ps2.executeUpdate();

    ps2.close();
    ps1.close();
    c1.close();
}
}

```

## **Pacote cadastro.model.util**

*ConectorBD.java*

//Conexão com o Banco de Dados

package cadastro.model.util;

import java.sql.\*;

```

public class ConectorBD {
    public static Connection getConnection () throws SQLException, ClassNotFoundException {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        return
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;tru
stServerCertificate=true", "loja", "loja");
    }

    public static PreparedStatement getPrepared (Connection c1, String sql) throws SQLException {
        return c1.prepareStatement(sql);
    }

    public static ResultSet getSelect (Connection c1, String sql) throws SQLException {
        Statement st = c1.createStatement();
        return st.executeQuery(sql);
    }

    public void close(Connection c1) throws Exception {
        c1.close();
    }

    public void close(Statement st) throws Exception {
        st.getConnection().close();
    }

    public void close(ResultSet rs) throws Exception {
        rs.close();
    }
}

```

### *SequenceManager.java*

```

package cadastro.model.util;

import java.sql.SQLException;
import java.sql.*;

public class SequenceManager {
    public static int getValue(String sequenceName) throws ClassNotFoundException, SQLException {
        int valor = -1;

        Connection c1 = ConectorBD.getConnection();
        PreparedStatement ps = c1.prepareStatement("SELECT NEXT VALUE FOR " + sequenceName);
        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
            valor = rs.getInt(1);
        }
        rs.close();
        ps.close();
        c1.close();
    }
}

```

```
        return valor;
    }
}
```

## Classe Principal de Testes

*ConectorBDTeste.java*

```
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import java.io.IOException;
import java.util.List;

public class CadastroBDTeste {

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException, Exception {
        //a. Instanciar pessoa física no banco
        PessoaFisica pf = new PessoaFisica(0, "João Riacho", "Rua 11", "Riacho do Sul",
            "PA", "123456789", "joao@riacho.com", "11100022233");
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO ();
        pfDAO.incluir(pf);
        System.out.println("Pessoa Física incluída:");
        pf.exibir();

        // b. Alterar os dados da pessoa física no banco
        pf.setNome("João do Riacho");
        pf.setEmail("joaodoriacho@email.com");
        pfDAO.alterar(pf);
        System.out.println("Pessoa Física alterada:");
        pf.exibir();

        // c. Consultar todas as pessoas físicas
        List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
        System.out.println("Lista de Pessoas Físicas:");
        for (PessoaFisica pessoa : pessoasFisicas) {
            pessoa.exibir();
        }

        // d. Excluir a pessoa física
        pfDAO.excluir(pf.getId());
        System.out.println("Pessoa Física excluída:");

        // e. Instanciar e incluir uma pessoa jurídica
        PessoaJuridica pj = new PessoaJuridica(0, "Empresa JJC ", "Av. 12", "Belém",
            "PA", "987654321", "contato@jjc.com", "12345678000199");
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
```

```

pjDAO.incluir(pj);
System.out.println("Pessoa Jurídica incluída:");
pj.exibir();

// f. Alterar os dados da pessoa jurídica
pj.setNome("Empresa JJA Ltda.");
pj.setEmail("contato@jjaltda.com");
pjDAO.alterar(pj);
System.out.println("Pessoa Jurídica alterada:");
pj.exibir();

// g. Consultar todas as pessoas jurídicas
List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
System.out.println("Lista de Pessoas Jurídicas:");
for (PessoaJuridica pessoa : pessoasJuridicas) {
    pessoa.exibir();
}

// h. Excluir a pessoa jurídica criada
pjDAO.excluir(pj.getId());
System.out.println("Pessoa Jurídica excluída.");
}
}

```

## Resultados da execução dos códigos

As janelas com os resultado da execução dos códigos em Java são mostradas abaixo.

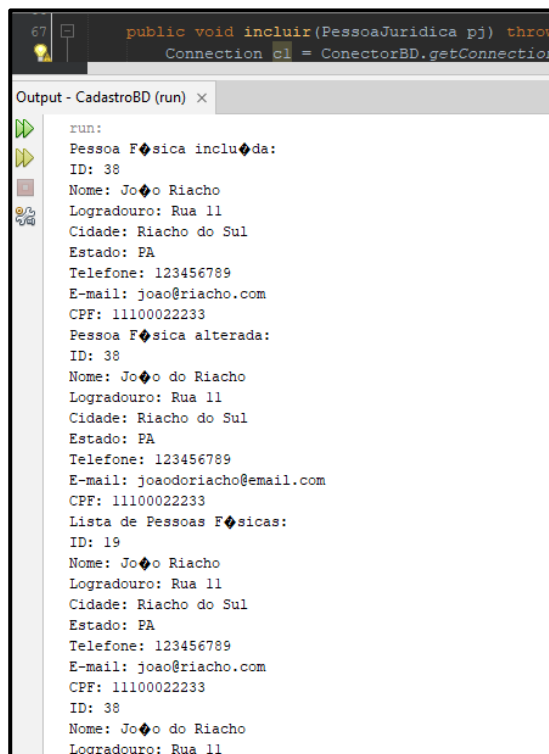


Figura 1 - Janela Parcial do NetBeans com o resultado da execução do código

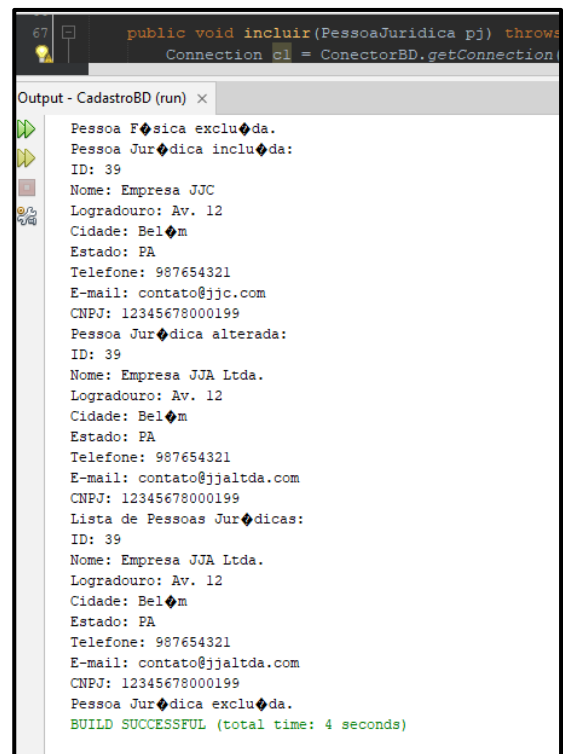


Figura 1 - Janela Parcial do NetBeans com o resultado da execução do código

## Análise e Conclusão

A missão mostrou como se pode implementar as diferentes cardinalidades. No projeto foi feito o uso das cardinalidades 1 x 1 e 1 x N. A cardinalidade 1 x 1 pode ser vista, por exemplo, na relação das tabelas “Pessoa” com “PessoaFisica”. A cardinalidade 1 x N foi utilizada nas relações entre as demais tabelas, como mostra a figura1. Elas foram implementadas através de chaves primárias (PK) e chaves estrangeiras (FK).

O projeto também mostrou como criar métodos para mapear a relação de herança no banco. Isto foi feito nas relações entre a Tabela “Pessoa” com as tabelas especializadas “PessoaFisica” e “PessoaJuridica”, que herdaram características comuns a todas as pessoas, como nome, endereço, telefone e e-mail.

Por fim, podemos ver como o Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados. A possibilidade de interação do programa com a ferramenta de modelagem, torna o desenvolvimento muito mais rápido e prático, minimizando os erros.

Armazenado no Github: <https://github.com/altairsf/MissaoN2M3.git>