



# Estácio

## Universidade Estácio

**Campus:** POLO FSP-RO

**Curso:** Desenvolvimento Full Stack

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

**TURMA:** 2025

**SEMESTRE LETIVO:** 01

**Integrantes:** Eliton Rodrigues de Oliveira

### Relatório Discente de Acompanhamento

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

##### 1. Título da Prática

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

##### 2. Objetivo da Prática

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.

No final do exercício, foi desenvolvido um aplicativo cadastral que utiliza o SQL Server para persistência de dados.

##### 3. Desenvolvimento da Prática

###### 3.1. Criação do Projeto e Configuração

1. Criado o projeto **CadastroBD** no NetBeans como Aplicativo Java Padrão.
2. Adicionado o driver **mssql-jdbc-12.2.0.jre11.jar** para conexão com o banco.
3. Configurada a conexão ao banco **loja** no SQL Server via NetBeans.
4. Teste de conexão bem-sucedido com a seguinte URL:
5. jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;

###### 3.2. Implementação das Classes

1. Criado o pacote **cadastrobd.model** com as classes:
  - **Pessoa**: id, nome, logradouro, cidade, estado, telefone, email.
  - **PessoaFisica** (extends Pessoa): adicionado campo cpf.

- **PessoaJuridica** (extends Pessoa): adicionado campo cnpj.
- 2. Criado o pacote **cadastro.model.util** com as classes:
  - **ConectorBD**: gerencia conexão, PreparedStatement e ResultSet.
  - **SequenceManager**: gera valores sequenciais para chaves primárias.
- 3. Implementado o padrão DAO no pacote **cadastro.model**:
  - **PessoaFisicaDAO** e **PessoaJuridicaDAO** com os métodos:
    - getPessoa(id), getPessoas(), incluir(pessoa), alterar(pessoa), excluir(id).
- 4. Criada a classe de testes **CadastroBDTeste** para operações CRUD:
  - Inserção, alteração, listagem e remoção de pessoas físicas e jurídicas.

#### 4. Resultados

- Todas as operações foram executadas corretamente, com saída no console do NetBeans semelhante ao esperado.
- A persistência dos dados no SQL Server foi confirmada através do Management Studio.

The screenshot shows the NetBeans IDE with several tabs open: SQL 1, Pessoa.java, CadastroBDTeste.java, PessoaFisica.java, PessoaJuridica.java, PessoaFisicaDAO.java, and PessoaJuridicaDAO.java. The **CadastroBDTeste.java** tab is active, displaying the following code:

```

1 import cadastrobd.model.PessoaFisica;
2 import cadastrobd.model.PessoaJuridica;
3
4 public class CadastroBDTeste {
5     public static void main(String[] args) {
6         PessoaFisica pf = new PessoaFisica("Eliton", "Rua c", "Rolim de Moura", "Ro", "3333-2222", "eliton@rodrigues.com", "00808461222");
7         pf.salvar();
8         System.out.println("Pessoa Fisica criada!\n");
9
10        pf.setEmail("eliton@rodrigues.com");
11        pf.atualizar();
12        System.out.println("Pessoa Fisica alterada!\n");
13    }
14 }

```

Below the code editor, the **Output - CadastroBD (run)** window shows the following output:

```

Pessoa Física atualizada com sucesso!
Pessoa Física alterada!

Lista de Pessoas Físicas:
ID: 1029
Nome: Eliton
Endereço: Rua c, Rolim de Moura - Ro
Telefone: 3333-2222
Email: eliton@rodrigues.com
CPF: 00808461222

```

#### 5. Análise e Conclusão

##### 5.1. Importância do Middleware JDBC

O JDBC permite a conexão entre a aplicação Java e o banco de dados relacional, possibilitando a persistência de dados de forma eficiente e segura.

##### 5.2. Diferença entre Statement e PreparedStatement

- Statement: Usado para consultas simples, sem parâmetros dinâmicos.
- PreparedStatement: Mais seguro e eficiente, pois evita SQL Injection e melhora a performance através do reuso de consultas compiladas.

##### 5.3. Benefícios do Padrão DAO

- Organização do código.

- Facilita a manutenção e reutilização.
- Separa a lógica de negócio do acesso ao banco de dados.

#### 5.4. Herança e Modelo Relacional

Quando trabalhamos com um banco de dados relacional, a herança pode ser refletida de diferentes formas:

- **Tabelas separadas** para cada classe (Pessoa, PessoaFisica, PessoaJuridica).
- **Uma única tabela** com um campo discriminador para indicar o tipo de pessoa.

No projeto, utilizamos tabelas separadas para manter a normalização dos dados.

## 2º Procedimento | Alimentando a Base

### Título da Prática

Cadastro de Pessoas com Persistência em Banco de Dados

### Objetivo da Prática

Implementar um sistema em Java para cadastro de pessoas físicas e jurídicas, utilizando persistência de dados em banco de dados SQL Server. O sistema deve permitir as operações de inclusão, alteração, exclusão, consulta individual e listagem de registros.

### Códigos Solicitados

Todos os códigos foram implementados conforme o roteiro da prática, incluindo as classes de modelo, DAO e a classe principal com menu de opções em modo texto.

#### ✓ Main.java

```
package cadastrobd.model;

import java.util.Scanner;

import java.util.List;

public class Main {

    public static void main(String[] args) {

        try (Scanner scanner = new Scanner(System.in)) {

            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

            int opcao;

            do {

                System.out.println("\n=== Menu Principal ===");

                System.out.println("1 - Incluir Pessoa");

                System.out.println("2 - Alterar Pessoa");

                System.out.println("3 - Excluir Pessoa");

                System.out.println("4 - Exibir Pessoa pelo ID");

                System.out.println("5 - Exibir Todas as Pessoas");

                System.out.println("0 - Sair");
```

```
System.out.print("Opção: ");

opcao = scanner.nextInt();

scanner.nextLine();

switch (opcao) {

    case 1 -> {

        System.out.println("Tipo de pessoa: F - Física | J - Jurídica");

        String tipo = scanner.nextLine().toUpperCase();

        switch (tipo) {

            case "F" -> {

                PessoaFisica pf = new PessoaFisica();

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

                pf.setNome(scanner.nextLine());

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

                pf.setLogradouro(scanner.nextLine());

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

                pf.setCidade(scanner.nextLine());

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

                pf.setEstado(scanner.nextLine());

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

                pf.setTelefone(scanner.nextLine());

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

                pf.setEmail(scanner.nextLine());

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

                pf.setCpf(scanner.nextLine());

                pessoaFisicaDAO.incluir(pf);

            }

            case "J" -> {

                PessoaJuridica pj = new PessoaJuridica();

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

                pj.setNome(scanner.nextLine());

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

                pj.setLogradouro(scanner.nextLine());

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

                pj.setCidade(scanner.nextLine());

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

                pj.setEstado(scanner.nextLine());

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

                pj.setTelefone(scanner.nextLine());

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

                pj.setEmail(scanner.nextLine());

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

                pj.setCnpj(scanner.nextLine());

                pessoaJuridicaDAO.incluir(pj);

            }

        }

    }

}
```

```

    }

    default -> System.out.println("Tipo inválido.");
}

}

case 2 -> {

    System.out.println("Tipo de pessoa: F - Física | J - Jurídica");

    String tipo = scanner.nextLine().toUpperCase();

    System.out.print("ID da pessoa a alterar: ");

    int id = scanner.nextInt();

    scanner.nextLine();

    if (tipo.equals("F")) {

        PessoaFisica pf = pessoaFisicaDAO.getPessoa(id);

        if (pf != null) {

            System.out.println("Dados atuais: " + pf);

            System.out.print("Novo CPF: ");

            pf.setCpf(scanner.nextLine());

            pessoaFisicaDAO.alterar(pf);

        } else {

            System.out.println("Pessoa não encontrada.");

        }

    } else if (tipo.equals("J")) {

        PessoaJuridica pj = pessoaJuridicaDAO.getPessoa(id);

        if (pj != null) {

            System.out.println("Dados atuais: " + pj);

            System.out.print("Novo CNPJ: ");

            pj.setCnpj(scanner.nextLine());

            pessoaJuridicaDAO.alterar(pj);

        } else {

            System.out.println("Pessoa não encontrada.");

        }

    } else {

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

    }

}

case 3 -> {

    System.out.println("Tipo de pessoa: F - Física | J - Jurídica");

    String tipo = scanner.nextLine().toUpperCase();


    System.out.print("ID da pessoa a excluir: ");

    int id = scanner.nextInt();

    scanner.nextLine();

    if (tipo.equals("F")) {

        pessoaFisicaDAO.excluir(id);

```

```

    } else if (tipo.equals("J")) {
        pessoaJuridicaDAO.excluir(id);
    } else {
        System.out.println("Tipo inválido.");
    }
}

case 4 -> {
    System.out.println("Tipo de pessoa: F - Física | J - Jurídica");
    String tipo = scanner.nextLine().toUpperCase();
    System.out.print("ID da pessoa a exibir: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    if (tipo.equals("F")) {
        PessoaFisica pf = pessoaFisicaDAO.getPessoa(id);
        System.out.println(pf != null ? pf : "Pessoa não encontrada.");
    } else if (tipo.equals("J")) {
        PessoaJuridica pj = pessoaJuridicaDAO.getPessoa(id);
        System.out.println(pj != null ? pj : "Pessoa não encontrada.");
    } else {
        System.out.println("Tipo inválido.");
    }
}

case 5 -> {
    System.out.println("\n--- Pessoas Físicas ---");
    List<PessoaFisica> listaPF = pessoaFisicaDAO.getPessoas();
    listaPF.forEach(System.out::println);
    System.out.println("\n--- Pessoas Jurídicas ---");
    List<PessoaJuridica> listaPJ = pessoaJuridicaDAO.getPessoas();
    listaPJ.forEach(System.out::println);
}

case 0 -> System.out.println("Saindo do programa...");
default -> System.out.println("Opção inválida.");
}

} while (opcao != 0);
} catch (Exception e) {
    System.out.println("Erro: " + e.getMessage());
}
}
}
}

```

#### Pessoa.java (Classe Abstrata)

```
package cadastrobd.model;
```

```

public abstract class Pessoa {

    protected int id;

    protected String nome;

    protected String logradouro;

    protected String cidade;

    protected String estado;

    protected String telefone;

    protected String 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 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; }


    @Override
    public String toString() {

        return "ID: " + id +

            ", Nome: " + nome +

            ", Logradouro: " + logradouro +

            ", Cidade: " + cidade +

            ", Estado: " + estado +

            ", Telefone: " + telefone +

            ", Email: " + email;

    }

}

```

#### PessoaFisica.java

```

package cadastrabd.model;

public class PessoaFisica extends Pessoa {

    private String cpf;

    public String getCpf() { return cpf; }

```

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

```
@Override
```

```
public String toString() {  
    return super.toString() + ", CPF: " + cpf;  
}  
}
```

#### **PessoaJuridica.java**

```
package cadastrobd.model;
```

```
public class PessoaJuridica extends Pessoa {  
    private String cnpj;  
  
    public String getCnpj() { return cnpj; }  
    public void setCnpj(String cnpj) { this.cnpj = cnpj; }
```

```
@Override
```

```
public String toString() {  
    return super.toString() + ", CNPJ: " + cnpj;  
}  
}
```

#### **ConectoBD.java**

```
package cadastrobd.model.util;
```

```
import java.sql.*;
```

```
public class ConectorBD {
```

```
private static final String URL =  
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;";  
  
private static final String USER = "loja";  
private static final String PASSWORD = "loja";
```

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

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



```

    }

    public static ResultSet getSelect(PreparedStatement stmt) throws SQLException {
        return stmt.executeQuery();
    }

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

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

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

```

#### PessoaFisicaDAO.java

```

package cadastrobd.model;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {
    public void incluir(PessoaFisica pf) throws SQLException {
        String sql = "INSERT INTO PessoaFisica (nome, logradouro, cidade, estado, telefone, email, cpf) VALUES (?, ?, ?, ?, ?, ?, ?)";

        try (Connection conn = Conexao.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, pf.getNome());

```

```

        stmt.setString(2, pf.getLogradouro());
        stmt.setString(3, pf.getCidade());
        stmt.setString(4, pf.getEstado());
        stmt.setString(5, pf.getTelefone());
        stmt.setString(6, pf.getEmail());
        stmt.setString(7, pf.getCpf());
        stmt.executeUpdate();
    }
}

```

```

public void alterar(PessoaFisica pf) throws SQLException {
    String sql = "UPDATE PessoaFisica SET cpf = ? WHERE id = ?";
    try (Connection conn = Conexao.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, pf.getCpf());
        stmt.setInt(2, pf.getId());
        stmt.executeUpdate();
    }
}

```

```

public void excluir(int id) throws SQLException {
    String sql = "DELETE FROM PessoaFisica WHERE id = ?";
    try (Connection conn = Conexao.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
    }
}

```

```

public PessoaFisica getPessoa(int id) throws SQLException {
    String sql = "SELECT * FROM PessoaFisica WHERE id = ?";
    try (Connection conn = Conexao.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            PessoaFisica pf = new PessoaFisica();
            pf.setId(rs.getInt("id"));
            pf.setNome(rs.getString("nome"));
            pf.setLogradouro(rs.getString("logradouro"));
            pf.setCidade(rs.getString("cidade"));
            pf.setEstado(rs.getString("estado"));
            pf.setTelefone(rs.getString("telefone"));

```

```

        pf.setEmail(rs.getString("email"));

        pf.setCpf(rs.getString("cpf"));

        return pf;
    }

    return null;
}

}

public List<PessoaFisica> getPessoas() throws SQLException {

    List<PessoaFisica> lista = new ArrayList<>();

    String sql = "SELECT * FROM PessoaFisica";

    try (Connection conn = Conexao.getConnection();

        Statement stmt = conn.createStatement();

        ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {

            PessoaFisica pf = new PessoaFisica();

            pf.setId(rs.getInt("id"));

            pf.setNome(rs.getString("nome"));

            pf.setLogradouro(rs.getString("logradouro"));

            pf.setCidade(rs.getString("cidade"));

            pf.setEstado(rs.getString("estado"));

            pf.setTelefone(rs.getString("telefone"));

            pf.setEmail(rs.getString("email"));

            pf.setCpf(rs.getString("cpf"));

            lista.add(pf);

        }

    }

    return lista;

}

}

```

## Resultados da Execução

A execução foi realizada com sucesso. Todas as funcionalidades de CRUD foram testadas tanto para Pessoa Física quanto para Pessoa Jurídica. Os dados foram devidamente armazenados, atualizados, recuperados e excluídos do banco de dados SQL Server, validando o correto funcionamento do sistema.

## Pessoa Juridica

=== Menu Principal ===

- 1 - Incluir Pessoa
- 2 - Alterar Pessoa
- 3 - Excluir Pessoa
- 4 - Exibir Pessoa pelo ID
- 5 - Exibir Todas as Pessoas
- 0 - Sair

Opção: 4

Tipo de pessoa: F - Física | J - Jurídica  
j

ID da pessoa a exibir: 6

=====

Id:6

Nome: Tech Solutions LTDA

CNPJ: 99887766554433

Logradouro: Rua das Empresas, 1000

=== Menu Principal ===

- 1 - Incluir Pessoa
- 2 - Alterar Pessoa
- 3 - Excluir Pessoa
- 4 - Exibir Pessoa pelo ID
- 5 - Exibir Todas as Pessoas
- 0 - Sair

Opção: 4

Tipo de pessoa: F - Física | J - Jurídica  
f

ID da pessoa a exibir: 1

=====

Id:1

Nome: João Silva

CPF: 12345678901

Logradouro: Rua A, 123

Exclusão

=== Menu Principal ===

1 - Incluir Pessoa

2 - Alterar Pessoa

3 - Excluir Pessoa

4 - Exibir Pessoa pelo ID

5 - Exibir Todas as Pessoas

0 - Sair

Opção: 3

Tipo de pessoa: F - Física | J - Jurídica  
f

ID da pessoa a excluir: 3

#### Análise e Conclusão:

1. **Diferenças entre persistência em arquivo e em banco de dados:**

- A persistência em arquivos é simples, mas não fornece recursos robustos para consultas, segurança e consistência.
- A persistência em banco de dados é mais adequada para aplicações reais, pois permite manipulação eficiente dos dados, consultas complexas e escalabilidade.

2. **Uso de operador lambda:**

- O uso de expressões lambda no Java 8+ permite simplificar o código, tornando a impressão de listas de objetos mais concisa e legível.

3. **Métodos static no main:**

- O método main é estático e não pertence a uma instância de objeto. Portanto, qualquer método chamado diretamente de main também precisa ser static, a menos que seja invocado a partir de uma instância da classe.

#### Repositório Git

O projeto foi armazenado no GitHub, acessível pelo link: [GitHub - Elitonr65/CadastroBD](#)