



# Estácio

## Mundo 3 - Missão 1

Faculdade Estácio - Polo Centro - Canela - RS

Curso: Desenvolvimento Full Stack

Disciplina: RPG0016 - Back-end Sem Banco Não Tem

Turma: 9001 - Semestre Letivo: 2025.1 - 3º semestre

Integrante: Pedro Henrique Marques Medeiros Pinho

Matrícula: 202402031831

Repositório Git: <https://github.com/PedroPinho23/BackEnd-Sem-Banco-Nao-Tem.git>

# BackEnd sem banco não tem

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

## Objetivos 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, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

## Códigos Utilizados

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrbd.model;
import java.io.Serializable;

/**
 *
 * @author Pedro
 */
public class Pessoa implements Serializable{
    private int id;
    private String nome;
    private String logradouro;
```

```
private String cidade;
private String estado;
private String telefone;
private String email;

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

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

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

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

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

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

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

```

    public int getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public String getEstado() {
        return estado;
    }

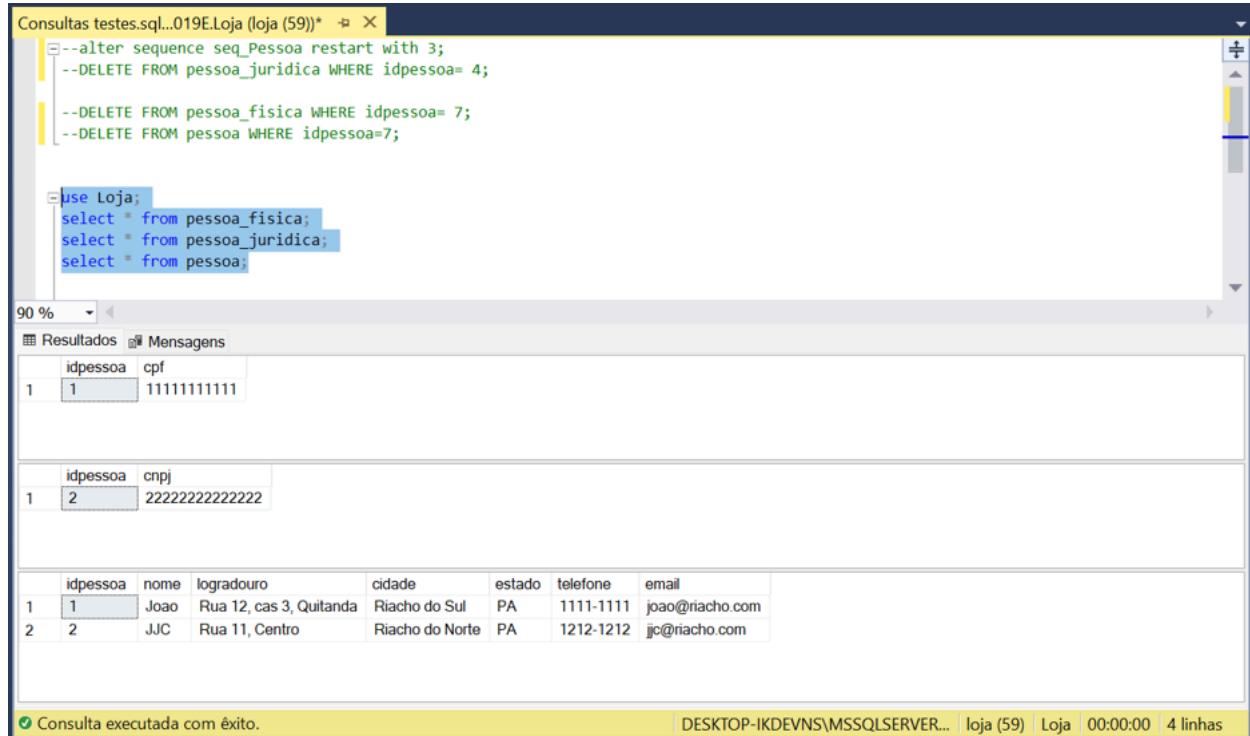
    public String getTelefone() {
        return telefone;
    }

    public String getEmail() {
        return email;
    }

    public void exibir(){
        System.out.print("id: "+this.id + "\n" + "Nome: " + this.nome + "\n" +
            "logradouro: "+this.logradouro+"\n"+"cidade: "+this.cidade+"\n"+
            "estado: "+this.estado+"\n" + "telefone: " + this.telefone + "\n"+ "email: " + this.email );
    }
}

```

## Resultados:



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a query window titled 'Consultas testes.sql...019E.Loja (loja (59))' containing the following SQL commands:

```
--alter sequence seq_Pessoa restart with 3;  
--DELETE FROM pessoa_juridica WHERE idpessoa= 4;  
  
--DELETE FROM pessoa_fisica WHERE idpessoa= 7;  
--DELETE FROM pessoa WHERE idpessoa=7;  
  
use Loja;  
select * from pessoa_fisica;  
select * from pessoa_juridica;  
select * from pessoa;
```

The bottom pane shows the 'Resultados' (Results) tab with a zoom level of 90%. It displays three query results:

- A table with columns 'idpessoa' and 'cpf' containing one row: idpessoa=1, cpf=11111111111.
- A table with columns 'idpessoa' and 'cnpj' containing one row: idpessoa=2, cnpj=22222222222222.
- A table with columns 'idpessoa', 'nome', 'logradouro', 'cidade', 'estado', 'telefone', and 'email' containing two rows of person data.

The status bar at the bottom indicates: 'Consulta executada com êxito.' (Query executed successfully), 'DESKTOP-IKDEVNS\MSSQLSERVER...', 'loja (59)', 'Loja', '00:00:00', and '4 linhas' (4 lines).

	idpessoa	cpf
1	1	11111111111

	idpessoa	cnpj
1	2	22222222222222

	idpessoa	nome	logradouro	cidade	estado	telefone	email
1	1	Joao	Rua 12, cas 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com
2	2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjo@riacho.com

Foram realizados testes diretos no método main com inserção manual de dados de pessoas físicas e jurídicas. Após a execução do programa, foi possível confirmar que os dados foram corretamente salvos no banco de dados e recuperados por meio de comandos SQL no SQL Server, com exibição correta das informações nas tabelas correspondentes.

## Conclusão:

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

O JDBC é importante porque permite que aplicações Java se conectem ao banco de dados e executem comandos SQL de forma simples e padronizada.

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

A diferença é que o PreparedStatement é mais seguro e eficiente que o Statement, pois evita ataques SQL Injection e permite reutilizar comandos com parâmetros.

Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO melhora a manutenibilidade ao separar a lógica de acesso a dados da lógica de negócio, centralizando operações em uma camada dedicada. Isso reduz acoplamento, facilita a modificação de fontes de dados sem impactar outras partes do sistema e promove reuso de código, tornando atualizações e testes mais eficientes

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

A herança em um banco relacional pode ser modelada com tabela única, tabelas por classe concreta ou tabelas por hierarquia. A abordagem mais comum usa chaves estrangeiras para conectar classes pai e filho, garantindo normalização, mas exigindo mais junções em consultas.