



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

ConectorBD

```
/*  
 * 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 cadastro.model.util;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.Statement;  
import javax.swing.JOptionPane;
```

```

/**
 *
 * @author Pedro
 */

public class ConectorBD {

    Connection conn = null;

    //Metodo para conectar java con SQLServer

    public Connection getConnection() throws Exception{
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn =
DriverManager.getConnection("jdbc:sqlserver://localhost\\MSSQLSERVER2019E:1433;
databaseName=Loja;encrypt=true;trustServerCertificate=true",
        "loja", "loja");
        return conn;
    }

    public void closeConnection()throws Exception{
        getConnection().close();
        //JOptionPane.showMessageDialog(null, "Conexao finalizada");
    }

    public PreparedStatement getPrepared(String sql) throws Exception {
        PreparedStatement ps = getConnection().prepareStatement(sql);
        return ps;
    }

    public void closeStatement(String sql)throws Exception{
        getPrepared(sql).close();
        //JOptionPane.showMessageDialog(null, "Statement finalizado");
    }

    public ResultSet getSelect(PreparedStatement ps) throws Exception {
        ResultSet rs = ps.executeQuery();
    }

```

```

        //ResultSet rs = getConnection().createStatement().executeQuery("");
        return rs;
    }

    public void closeResult(PreparedStatement ps)throws Exception{
        getSelect(ps).close();
        //JOptionPane.showMessageDialog(null, "ResultSet finalizado");
    }

}

```

PessoaJuridicaDAO

```

/*
 * 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 cadastro.model;

import cadastro.model.util.ConectorBD;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import cadastrobd.model.PessoaJuridica;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Pedro
 */
public class PessoaJuridicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaJuridica getPessoa(int id)throws Exception {
        PessoaJuridica pessoa = null;
        String sql = "select *\n" +
            "from pessoa, pessoa_juridica\n" +
            "where pessoa.idpessoa = "+ id + "AND " +

```

```

        "pessoa.idpessoa = pessoa_juridica.idpessoa;";
PreparedStatement ps = connection.getPrepared(sql);
ResultSet resultado = ps.executeQuery();
while(resultado.next()){
    pessoa = new PessoaJuridica(resultado.getInt("idpessoa"),
        resultado.getString("nome"),
        resultado.getString("logradouro"),
        resultado.getString("cidade"),
        resultado.getString("estado"),
        resultado.getString("telefone"),
        resultado.getString("email"),
        resultado.getString("cnpj"));
    connection.closeConnection();
    //connection.closeResult(ps);
    connection.closeStatement(sql);
} return pessoa;
}

```

```

public List<PessoaJuridica> getPessoas() throws Exception{
    List<PessoaJuridica> lista = new ArrayList<>();
    String sql = "select *\n" +
        "from pessoa, pessoa_juridica\n" +
        "where pessoa.idpessoa = pessoa_juridica.idpessoa;";
    PreparedStatement ps = connection.getPrepared(sql);
    ResultSet resultado = ps.executeQuery();
    while(resultado.next()){
        //System.out.println(resultado.getString(5));
        lista.add(new PessoaJuridica(resultado.getInt("idpessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cnpj")));
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sql);
    } return lista;
}

```

```

public void incluir(PessoaJuridica pessoajuridica) throws Exception{
    String sqljuridica = "insert into pessoa_juridica (idpessoa, cnpj) values (?,?)";
}

```

```

String sqlpessoa = "insert into pessoa (idpessoa,nome,logradouro, cidade,"
    + "estado, telefone, email ) values (?, ?, ?, ?, ?, ?, ?)";
PreparedStatement ps = connection.getPreparedStatement(sqljuridica);
PreparedStatement ps1 = connection.getPreparedStatement(sqlpessoa);
//ResultSet resultado = ps.executeQuery();
ps.setInt(1, pessoajuridica.getId());
ps.setString(2, pessoajuridica.getCnpj());
ps1.setInt(1, pessoajuridica.getId());
ps1.setString(2, pessoajuridica.getNome());
ps1.setString(3, pessoajuridica.getLogradouro());
ps1.setString(4, pessoajuridica.getCidade());
ps1.setString(5, pessoajuridica.getEstado());
ps1.setString(6, pessoajuridica.getTelefone());
ps1.setString(7, pessoajuridica.getEmail());
ps1.execute();
ps.execute();
connection.closeConnection();
//connection.closeResult(ps);
connection.closeStatement(sqljuridica);
}

```

```

public void alterar(int id, String cnpj, String nome, String logradouro,
String cidade, String estado,String telefone, String email)throws Exception{
PessoaJuridica pessoa = getPessoa(id);
String sqljuridica = "UPDATE pessoa_juridica SET cnpj=? where idpessoa = "+id;
String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?,"
    + "estado=?, telefone=?, email=? WHERE idpessoa= "+id;
PreparedStatement ps = connection.getPreparedStatement(sqljuridica);
PreparedStatement ps1 = connection.getPreparedStatement(sqlpessoa);
if(cnpj.equals("")){
ps.setString(1, pessoa.getCnpj());
} else{
ps.setString(1, cnpj);
}
}

```

```

if(nome.equals("")){
ps1.setString(1, pessoa.getNome());
} else{
ps1.setString(1, nome);
}
}

```

```

if(logradouro.equals("")){
ps1.setString(2, pessoa.getLogradouro());
} else{
}

```

```
ps1.setString(2, logradouro);  
}
```

```
if(cidade.equals("")){  
ps1.setString(3, pessoa.getCidade());  
} else{  
ps1.setString(3, cidade);  
}
```

```
if(estado.equals("")){  
ps1.setString(4, pessoa.getEstado());  
} else{  
ps1.setString(4, estado);  
}
```

```
if(telefone.equals("")){  
ps1.setString(5, pessoa.getTelefone());  
} else{  
ps1.setString(5, telefone);  
}
```

```
if(email.equals("")){  
ps1.setString(6, pessoa.getEmail());  
} else{  
ps1.setString(6, email);  
}
```

```
ps.execute();  
ps1.execute();  
connection.closeConnection();  
//connection.closeResult(ps);  
connection.closeStatement(sqljuridica);  
}
```

```
public void excluir(int id)throws Exception{  
String sqljuridica = "DELETE FROM pessoa_juridica WHERE idpessoa="+id;  
String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa="+id;  
PreparedStatement ps = connection.getPrepared(sqljuridica);  
PreparedStatement ps1 = connection.getPrepared(sqlpessoa);  
ps.execute();  
ps1.execute();  
connection.closeConnection();  
//connection.closeResult(ps);  
connection.closeStatement(sqljuridica);  
}
```

```
}
```

PessoaFisicaDAO

```
/*  
 * 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 cadastro.model;  
import cadastrobd.model.PessoaFisica;  
import java.util.ArrayList;  
import java.util.List;  
import cadastro.model.util.ConectorBD;  
import com.sun.jdi.connect.spi.Connection;  
import java.sql.ResultSet;  
import java.sql.PreparedStatement;
```

```
/**  
 *  
 * @author Pedro  
 */
```

```
public class PessoaFisicaDAO {  
  
    public ConectorBD connection = new ConectorBD();  
  
    public PessoaFisica getPessoa(int id)throws Exception {  
        PessoaFisica pessoa = null;  
        String sql = "select *\n" +  
            "from pessoa, pessoa_fisica\n" +  
            "where pessoa.idpessoa = "+ id + "AND " +  
            "pessoa.idpessoa = pessoa_fisica.idpessoa;";  
        PreparedStatement ps = connection.getPrepared(sql);  
        ResultSet resultado = ps.executeQuery();  
        while(resultado.next()){  
            pessoa = new PessoaFisica(resultado.getInt("idpessoa"),  
                resultado.getString("nome"),  
                resultado.getString("logradouro"),  
                resultado.getString("cidade"),
```



```

        resultado.getString("estado"),
        resultado.getString("telefone"),
        resultado.getString("email"),
        resultado.getString("cpf"));
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sql);
    } return pessoa;
}

```

```

public List<PessoaFisica> getPessoas() throws Exception{
    List<PessoaFisica> lista = new ArrayList<>();
    String sql = "select *\n" +
        "from pessoa, pessoa_fisica\n" +
        "where pessoa.idpessoa = pessoa_fisica.idpessoa;";
    PreparedStatement ps = connection.getPrepared(sql);
    ResultSet resultado = ps.executeQuery();
    while(resultado.next()){
        //System.out.println(resultado.getString(5));
        lista.add(new PessoaFisica(resultado.getInt("idpessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cpf")));
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sql);
    } return lista;
}

```

```

public void incluir(PessoaFisica pessoafisica) throws Exception{
    String sqlfisica = "insert into pessoa_fisica (idpessoa, cpf) values (?,?)";
    String sqlpessoa = "insert into pessoa (idpessoa,nome,logradouro, cidade,"
        + "estado, telefone, email ) values (?,?,?,?,?,?,?)";
}

```

```

PreparedStatement ps = connection.getPrepared(sqlfisica);
PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
//ResultSet resultado = ps.executeQuery();
ps.setInt(1, pessoafisica.getId());
ps.setString(2, pessoafisica.getCpf());
ps1.setInt(1, pessoafisica.getId());
ps1.setString(2, pessoafisica.getNome());
ps1.setString(3, pessoafisica.getLogradouro());
ps1.setString(4, pessoafisica.getCidade());
ps1.setString(5, pessoafisica.getEstado());
ps1.setString(6, pessoafisica.getTelefone());
ps1.setString(7, pessoafisica.getEmail());
ps1.execute();
ps.execute();
connection.closeConnection();
//connection.closeResult(ps);
connection.closeStatement(sqlfisica);
}

```

```

public void alterar(int id, String cpf, String nome, String logradouro,
String cidade, String estado,String telefone, String email)throws Exception{
PessoaFisica pessoa = getPessoa(id);
String sqlfisica = "UPDATE pessoa_fisica SET cpf=? where idpessoa = "+id;
String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?,"
+ "estado=?, telefone=?, email=? WHERE idpessoa= "+id;
PreparedStatement ps = connection.getPrepared(sqlfisica);
PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
if(cpf.equals("")){
ps.setString(1, pessoa.getCpf());
} else{
ps.setString(1, cpf);
}

if(nome.equals("")){
ps1.setString(1, pessoa.getNome());
} else{
ps1.setString(1, nome);
}

if(logradouro.equals("")){

```

```
ps1.setString(2, pessoa.getLogradouro());  
} else{  
ps1.setString(2, logradouro);  
}
```

```
if(cidade.equals("")){  
ps1.setString(3, pessoa.getCidade());  
} else{  
ps1.setString(3, cidade);  
}
```

```
if(estado.equals("")){  
ps1.setString(4, pessoa.getEstado());  
} else{  
ps1.setString(4, estado);  
}
```

```
if(telefone.equals("")){  
ps1.setString(5, pessoa.getTelefone());  
} else{  
ps1.setString(5, telefone);  
}  
if(email.equals("")){  
ps1.setString(6, pessoa.getEmail());  
} else{  
ps1.setString(6, email);  
}  
ps.execute();  
ps1.execute();  
connection.closeConnection();  
//connection.closeResult(ps);  
connection.closeStatement(sqlfisica);  
}
```

```
public void excluir(int id)throws Exception{  
String sqlfisica = "DELETE FROM pessoa_fisica WHERE idpessoa="+id;  
String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa="+id;  
PreparedStatement ps = connection.getPrepared(sqlfisica);  
PreparedStatement ps1 = connection.getPrepared(sqlpessoa);  
ps.execute();
```

```

        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqlfisica);
    }
}

```

CadastroBDTeste2

```

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastro.model.util.SequenceManager;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.util.List;
import java.util.Scanner;

/*
 * 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
 */

/**
 *
 * @author Pedro
 */
public class CadastroBDTeste2 {

    public static void main(String[] args)throws Exception {

        Scanner scan = new Scanner(System.in);
        String escolha;

        do {
            System.out.println("=====");
            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 - Finalizar Programa");
System.out.println("=====");

escolha = scan.next();
SequenceManager seq = new SequenceManager();

switch (escolha) {

    // Incluir
    case "1":
        do {
            System.out.println("=====");
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

            escolha = scan.next();
            scan.nextLine();

            switch (escolha.toUpperCase()) {

                case "F":
                    System.out.println("Insira os dados... ");
                    System.out.print("Nome: ");
                    String nome = scan.nextLine();
                    System.out.print("Logradouro: ");
                    String logradouro = scan.nextLine();
                    System.out.print("Cidade: ");
                    String cidade = scan.nextLine();
                    System.out.print("Estado: ");
                    String estado = scan.nextLine();
                    System.out.print("Telefone: ");
                    String telefone = scan.nextLine();
                    System.out.print("Email: ");
                    String email = scan.nextLine();
                    System.out.print("CPF: ");
                    String cpf = scan.nextLine();

```

```

        PessoaFisica pessoaIncluir = new
PessoaFisica(seq.getValue("seq_Pessoa"),nome, logradouro,
        cidade, estado, telefone,email,cpf);
        PessoaFisicaDAO pessoaPF = new PessoaFisicaDAO();
        pessoaPF.incluir(pessoaIncluir);

        System.out.println("Inclusao realizada com sucesso!");
        break;

        case "J":
        System.out.println("Insira os dados... ");
        System.out.print("Nome: ");
        String nomej = scan.nextLine();
        System.out.print("Logradouro: ");
        String logradouroj = scan.nextLine();
        System.out.print("Cidade: ");
        String cidadej = scan.nextLine();
        System.out.print("Estado: ");
        String estadoj = scan.nextLine();
        System.out.print("Telefone: ");
        String telefonej = scan.nextLine();
        System.out.print("Email: ");
        String emailj = scan.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scan.nextLine();

        PessoaJuridica pessoaJIncluir = new
PessoaJuridica(seq.getValue("seq_Pessoa"),nomej,
        logradouroj,cidadej, estadoj, telefonej,emailj,cnpj);
        PessoaJuridicaDAO pessoaPJ = new PessoaJuridicaDAO();
        pessoaPJ.incluir(pessoaJIncluir);

        System.out.println("Inclusao realizada com sucesso!");
        break;

        case "M":
        break;

        default:
        System.out.println("Opcao invalida.");

```

```

        break;
    }
} while (!escolha.equalsIgnoreCase("M"));
break;

// Alterar
case "2":
do {
    System.out.println("=====");
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

    escolha = scan.next();
    scan.nextLine();

    switch (escolha.toUpperCase()) {

        case "F":
            System.out.println("Digite o ID da pessoa: ");
            int idPessoaFisica = scan.nextInt();
            scan.nextLine();

            PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);
            PessoaFisicaDAO pessoaFisicaLocalizadaAlterar = new
PessoaFisicaDAO();
            //PessoaFisica pessoaFisicaLocalizada =
pfRepo.obter(idPessoaFisica);

            if (pessoaFisicaLocalizada != null) {
                pessoaFisicaLocalizada.exibir();

                System.out.println("Nome atual: " +
pessoaFisicaLocalizada.getNome());
                System.out.print("Novo nome: ");
                String novoNome = scan.nextLine();

                System.out.println("Logradouro: " +
pessoaFisicaLocalizada.getLogradouro());
                System.out.print("Novo Logradouro: ");
                String novoLogradouro = scan.nextLine();

```

```

        System.out.println("Cidade: " +
        pessoaFisicaLocalizada.getCidade());
        System.out.print("Nova Cidade: ");
        String novoCidade = scan.nextLine();

        System.out.println("Estado: " +
        pessoaFisicaLocalizada.getEstado());
        System.out.print("Novo Estado: ");
        String novoEstado = scan.nextLine();

        System.out.println("Telefone: " +
        pessoaFisicaLocalizada.getTelefone());
        System.out.print("Novo Telefone: ");
        String novoTelefone = scan.nextLine();

        System.out.println("Email: " + pessoaFisicaLocalizada.getEmail());
        System.out.print("Novo Email: ");
        String novoEmail = scan.nextLine();

        System.out.println("CPF atual: " +
        pessoaFisicaLocalizada.getCpf());
        System.out.print("Novo CPF: ");
        String novoCPF = scan.nextLine();

        pessoaFisicaLocalizadaAlterar.alterar( idPessoaFisica,novoCPF,
        novoNome, novoLogradouro, novoCidade,
        novoEstado, novoTelefone, novoEmail );

        System.out.println("Pessoa alterada com sucesso!");
    } else
        System.out.println("Pessoa nao localizada! ");
        break;

    case "J":
        System.out.println("Digite o ID da pessoa: ");
        int idPessoaJuridica = scan.nextInt();
        scan.nextLine();

```



```

        PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);
        PessoaJuridicaDAO pessoaJuridicaLocalizadaAlterar = new
PessoaJuridicaDAO();

        if (pessoaJuridicaLocalizada != null) {
            pessoaJuridicaLocalizada.exibir();

            System.out.println("Nome atual: " +
pessoaJuridicaLocalizada.getNome());
            System.out.print("Novo nome: ");
            String novoNome = scan.nextLine();

            System.out.println("Logradouro: " +
pessoaJuridicaLocalizada.getLogradouro());
            System.out.print("Novo Logradouro: ");
            String novoLogradouro = scan.nextLine();

            System.out.println("Cidade: " +
pessoaJuridicaLocalizada.getCidade());
            System.out.print("Nova Cidade: ");
            String novoCidade = scan.nextLine();

            System.out.println("Estado: " +
pessoaJuridicaLocalizada.getEstado());
            System.out.print("Novo Estado: ");
            String novoEstado = scan.nextLine();

            System.out.println("Telefone: " +
pessoaJuridicaLocalizada.getTelefone());
            System.out.print("Novo Telefone: ");
            String novoTelefone = scan.nextLine();

            System.out.println("Email: " +
pessoaJuridicaLocalizada.getEmail());
            System.out.print("Novo Email: ");
            String novoEmail = scan.nextLine();

            System.out.println("CNPJ atual: " +
pessoaJuridicaLocalizada.getCnpj());

```

```

        System.out.print("Novo CNPJ: ");
        String novoCNPJ = scan.nextLine();

        pessoaJuridicaLocalizadaAlterar.alterar( idPessoaJuridica,
novoCNPJ, novoNome, novoLogradouro, novoCidade,
        novoEstado, novoTelefone, novoEmail);

        System.out.println("Pessoa alterada com sucesso!");
    } else
        System.out.println("Pessoa nao localizada!");
        break;

    case "M":
        break;

    default:
        System.out.println("Opcao invalida.");
        break;
    }
} while (!escolha.equalsIgnoreCase("M"));
break;

// EXCLUIR
case "3":
do {
    System.out.println("=====");
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

    escolha = scan.next();
    scan.nextLine();

    switch (escolha.toUpperCase()) {

        case "F":
            System.out.println("Digite o ID da pessoa: ");
            int idPessoaFisica = scan.nextInt();

            PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);

```

```

        PessoaFisicaDAO pessoaFisicaLocalizadaExcluir = new
PessoaFisicaDAO();

        if (pessoaFisicaLocalizada != null) {
            pessoaFisicaLocalizada.exibir();
            pessoaFisicaLocalizadaExcluir.excluir(idPessoaFisica);

            System.out.println("Pessoa excluida com sucesso!");
        } else
            System.out.println("Pessoa nao localizada!");
        break;

        case "J":
            System.out.println("Digite o ID da pessoa: ");
            int idPessoaJuridica = scan.nextInt();

            PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);
            PessoaJuridicaDAO pessoaJuridicaLocalizadaExcluir = new
PessoaJuridicaDAO();

            if (pessoaJuridicaLocalizada != null) {
                pessoaJuridicaLocalizada.exibir();
                pessoaJuridicaLocalizadaExcluir.excluir(idPessoaJuridica);

                System.out.println("Pessoa excluida com sucesso!");
            } else
                System.out.println("Pessoa nao localizada!");
            break;

        case "M":
            break;

        default:
            System.out.println("Opcao invalida.");
            break;
    }

} while (!escolha.equalsIgnoreCase("M"));

```

```

break;

// obter pelo Id
case "4":
do {
System.out.println("=====");
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

escolha = scan.next();
scan.nextLine();

switch (escolha.toUpperCase()) {

    case "F":
        System.out.println("Digite o ID da pessoa: ");
        int idPessoaFisica = scan.nextInt();

        PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);

        if (pessoaFisicaLocalizada != null) {
            System.out.println("Pessoa localizada!");
            pessoaFisicaLocalizada.exibir();
        } else
            System.out.println("Pessoa nao localizada!");
        break;

    case "J":
        System.out.println("Digite o ID da pessoa: ");
        int idPessoaJuridica = scan.nextInt();

        PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);

        if (pessoaJuridicaLocalizada != null) {
            System.out.println("Pessoa localizada!");
            pessoaJuridicaLocalizada.exibir();
        } else
            System.out.println("Pessoa nao localizada!");
        break;

```

```

        case "M":
            break;

        default:
            System.out.println("Opcao invalida.");
            break;
    }

} while (!escolha.equalsIgnoreCase("M"));
break;

//obterTodos
case "5":
do {
    System.out.println("=====");
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

    escolha = scan.next();
    scan.nextLine();

    switch (escolha.toUpperCase()) {

        case "F":
            System.out.println("Pessoas fisicas:");
            PessoaFisicaDAO pessoasFisica = new PessoaFisicaDAO();
            List<PessoaFisica> resultado = pessoasFisica.getPessoas();
            for (PessoaFisica pessoaFisica : resultado) {
                pessoaFisica.exibir();
            }
            break;

        case "J":
            System.out.println("Pessoas juridicas:");
            PessoaJuridicaDAO pessoasJuridica = new PessoaJuridicaDAO();
            List<PessoaJuridica> resultado2 = pessoasJuridica.getPessoas();
            for (PessoaJuridica pessoaJuridica : resultado2) {
                pessoaJuridica.exibir();
            }
            break;
    }
}

```

```
        case "M":
            break;

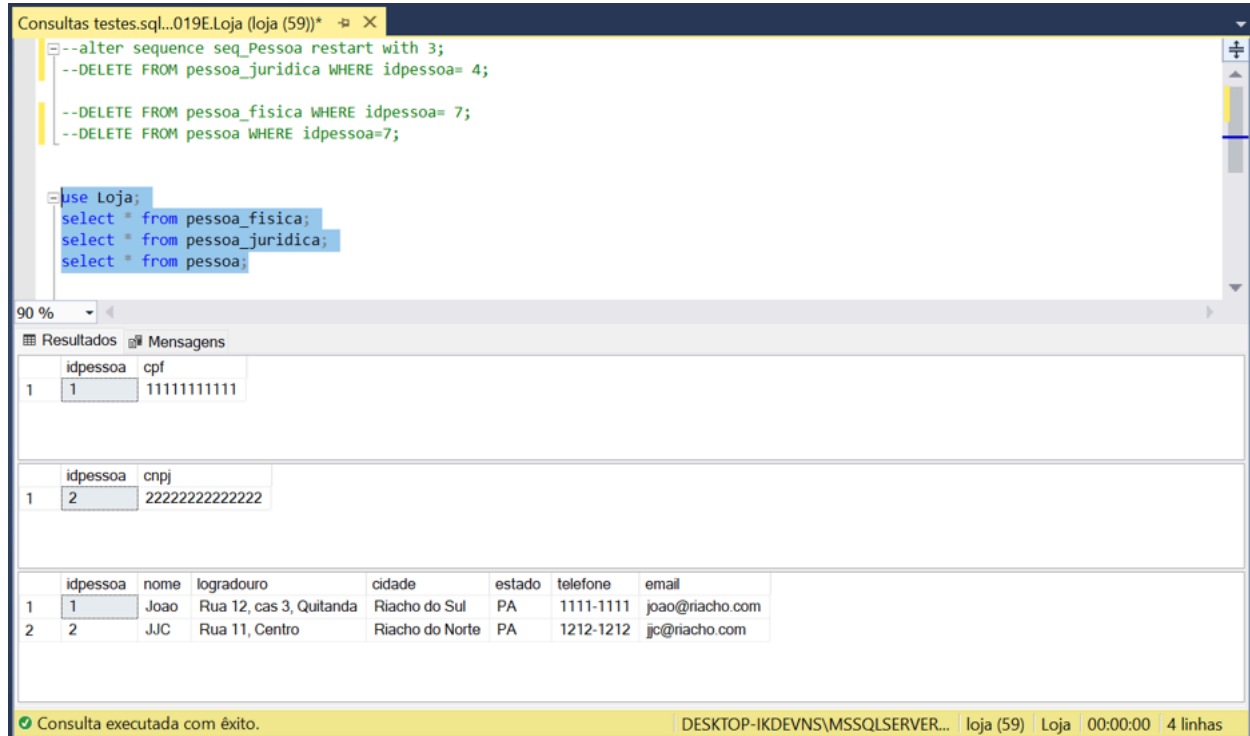
        default:
            System.out.println("Opcao invalida");
            break;
    }

    } while (!escolha.equalsIgnoreCase("M"));
    break;
    case "0":
        System.out.println("Sistema Finalizado com sucesso.");
        break;

    default:
        System.out.println("Opcao invalida");
        break;
    }
    } while (!escolha.equals("0"));
    scan.close();

    }
}
```

Resultados:



The screenshot displays a SQL Server Enterprise Manager window titled "Consultas testes.sql...019E.Loja (loja (59))*". The query window contains the following SQL code:

```
--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 results pane shows three tables of data:

	idpessoa	cpf
1	1	111111111111

	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

A status bar at the bottom indicates: "Consulta executada com êxito." (Query executed successfully). The right side of the status bar shows: "DESKTOP-IKDEVNS\MSSQLSERVER..." | loja (59) | Loja | 00:00:00 | 4 linhas.

Foram testadas as funcionalidades de conexão com o banco de dados e manipulação dos dados usando PreparedStatement. Os testes confirmaram que os dados foram inseridos, consultados, alterados e excluídos corretamente, comprovando o funcionamento da integração entre Java e SQL Server.

Conclusão:

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

A persistência em arquivo é simples e salva dados localmente, mas com menos segurança e controle. Já o banco de dados é mais robusto, permite consultas, múltiplos acessos e maior integridade dos dados.

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

O operador lambda simplifica a impressão ao permitir escrever código mais curto e direto, evitando estruturas repetitivas como loops tradicionais, especialmente com streams e listas.

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

Porque o método main é static, e só pode chamar diretamente outros métodos que também sejam static, ou seja, que não dependem de uma instância (objeto) para serem executados.