Estácio de Sá - Unidade São João de Meriti

Desenvolvimento Full Stack
Aluno: Victor de A. Costa

Missão Prática | Nível 3 | Mundo 3 - 2024.3
BACKEND SEM BANCO NÃO TEM!

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

2º Procedimento | Alimentando a Base

---> CÓDIGOS UTILIZADOS

CadastroBD.java

```
package cadastrobd;
/**
*
* @author victorcosta
*/
import java.util.*;
import cadastro.model.util.ConectorBD;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastro.model.util.SequenceManager;
public class CadastroBD {
// Instanciando ConectorBD e SequenceManager (simulando a inicialização)
private static Scanner scanner = new Scanner(System.in);
private static ConectorBD conectorBD = new ConectorBD();
private static SequenceManager sequenceManager = new SequenceManager();
// Instanciando DAOs
private static PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD,
sequenceManager);
private static PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO(conectorBD, sequenceManager);
public static void main(String[] args) {
int opcao;
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("==================================");
opcao = scanner.nextInt();
switch (opcao) {
case 1:
incluirPessoa();
break;
case 2:
alterarPessoa();
break;
case 3:
excluirPessoa();
break;
case 4:
buscarPeloId();
break;
case 5:
exibirTodos();
break;
case 0:
System.out.println("Saindo...");
break;
default:
System.out.println("Opção inválida!");
```

```
return;
 }
 System.out.println("Insira os dados...");
 System.out.println("Nome:");
 nome = scanner.next();
 scanner.nextLine();
 System.out.println("Logradouro:");
 logradouro = scanner.nextLine();
 System.out.println("Cidade:");
 cidade = scanner.nextLine();
boolean isValidInputEstado = false;
 do {
 System.out.println("Estado:");
 estado = scanner.next();
 scanner.nextLine();
 if (estado.length() > 2) {
 System.out.println("Entrada inválida! Digite uma abreviação de estado com no
 máximo 2 caracteres.");
 } else {
 isValidInputEstado = true;
 }
 } while (!isValidInputEstado);
 System.out.println("Telefone:");
 telefone = scanner.next();
 System.out.println("Email:");
 email = scanner.next();
 boolean isValidInput = false;
 if (tipo.equals("J")) {
 do {
 System.out.println("CNPJ:");
 cnpj = scanner.next();
 if (cnpj.length() > 14) {
 System.out.println("Entrada inválida! Digite um máximo de 14 caracteres.");
 } else {
 isValidInput = true;
 }
 } while (!isValidInput);
 PessoaJuridica pessoaJuridica = new PessoaJuridica(0, nome, logradouro, cidade,
 estado, telefone, email, cnpj);
 pessoaJuridicaDAO.incluir(pessoaJuridica);
 }
 if (tipo.equals("F")) {
 do {
 System.out.println("CPF:");
 cpf = scanner.next();
 if (cpf.length() > 11) {
System.out.println("Entrada inválida! Digite um máximo de 11 caracteres.");
 } else {
 isValidInput = true;
 }
 } while (!isValidInput);
 PessoaFisica pessoaFisica = new PessoaFisica(0, nome, logradouro, cidade, estado,
 telefone, email, cpf);
 pessoaFisicaDAO.incluir(pessoaFisica);
 }
 }
 private static void alterarPessoa() {
 String tipo;
 int id;
 String nome;
 String cpf;
 String cnpj;
 PessoaJuridica pessoaJuridica = null;
 PessoaFisica pessoaFisica = null;
 String logradouro;
 String cidade;
 String estado;
 String telefone;
 String email;
 System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
 tipo = scanner.next().toUpperCase();
 if (!tipo.equals("J") &&!tipo.equals("F")) {
 System.out.println("Opção inválida!");
 return;
 }
```

```
}
 if (tipo.equals("F")) {
 pessoaFisica = pessoaFisicaDAO.getPessoa(id);
 if (pessoaFisica == null) {
 System.out.println("Pessoa não encontrada");
 return;
 }
 }
 System.out.println("Insira os dados...");
 if (tipo.equals("J")) {
 System.out.printf("Nome (%s):", pessoaJuridica.getNome());
 System.out.println("");
 }
 if (tipo.equals("F")) {
 System.out.printf("Nome (%s):", pessoaFisica.getNome());
 System.out.println("");
 }
 nome = scanner.next();
 scanner.nextLine();
 if (tipo.equals("J")) {
 System.out.printf("Logradouro (%s):", pessoaJuridica.getLogradouro());
 System.out.println("");
 }
 if (tipo.equals("F")) {
 System.out.printf("Logradouro (%s):", pessoaFisica.getLogradouro());
 System.out.println("");
 }
 logradouro = scanner.nextLine();
 if (tipo.equals("J")) {
 System.out.printf("Cidade (%s):", pessoaJuridica.getCidade());
 System.out.println("");
 }
 if (tipo.equals("F")) {
 System.out.printf("Cidade (%s):", pessoaFisica.getCidade());
System.out.println("");
 }
 cidade = scanner.nextLine();
 if (tipo.equals("J")) {
 System.out.printf("Estado (%s):", pessoaJuridica.getEstado());
 System.out.println("");
 }
 if (tipo.equals("F")) {
 System.out.printf("Estado (%s):", pessoaFisica.getEstado());
 System.out.println("");
 }
 boolean isValidInputEstado = false;
 do {
 estado = scanner.next();
 scanner.nextLine();
 if (estado.length() > 2) {
 System.out.println("Entrada inválida! Digite uma abreviação de estado com no
 máximo 2 caracteres.");
 } else {
 isValidInputEstado = true;
 }
 } while (!isValidInputEstado);
 if (tipo.equals("J")) {
 System.out.printf("Telefone (%s):", pessoaJuridica.getTelefone());
 }
 if (tipo.equals("F")) {
 System.out.printf("Telefone (%s):", pessoaFisica.getTelefone());
 }
 System.out.println("");
 telefone = scanner.next();
 if (tipo.equals("J")) {
 System.out.printf("Email (%s):", pessoaJuridica.getEmail());
 }
 if (tipo.equals("F")) {
 System.out.printf("Email (%s):", pessoaFisica.getEmail());
}
 System.out.println("");
 email = scanner.next();
 boolean isValidInput = false;
 if (tipo.equals("J")) {
 do {
```

```
System.out.printf("CNPJ (%s):", pessoaJuridica.getCnpj());
System.out.println("");
cnpj = scanner.next();
if (cnpj.length() > 14) {
System.out.println("Entrada inválida! Digite um máximo de 14 caracteres.");
} else {
isValidInput = true;
}
} while (!isValidInput);
PessoaJuridica pessoaJuridicaData = new PessoaJuridica(id, nome, logradouro, cidade,
estado, telefone, email, cnpj);
pessoaJuridicaDAO.alterar(pessoaJuridicaData);
}
if (tipo.equals("F")) {
do {
System.out.printf("CPF (%s):", pessoaFisica.getCpf());
System.out.println("");
cpf = scanner.next();
if (cpf.length() > 11) {
System.out.println("Entrada inválida! Digite um máximo de 11 caracteres.");
} else {
isValidInput = true;
}
} while (!isValidInput);
PessoaFisica pessoaFisicaData = new PessoaFisica(id, nome, logradouro, cidade,
estado, telefone, email, cpf);
pessoaFisicaDAO.alterar(pessoaFisicaData);
}
}
private static void buscarPeloId() {
String tipo;
int id;
System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
tipo = scanner.next().toUpperCase();
if (!tipo.equals("J") &&!tipo.equals("F")) {
System.out.println("Opção inválida!");
return;
}
System.out.println("Digite o id da Pessoa:");
id = scanner.nextInt();
if (tipo.equals("J")) {
PessoaJuridica pessoa = pessoaJuridicaDAO.getPessoa(id);
if (pessoa == null) {
System.out.println("Pessoa não encontrada");
} else {
System.out.println("Pessoa Juridica:");
pessoa.exibir();
}
}
if (tipo.equals("F")) {
PessoaFisica pessoa = pessoaFisicaDAO.getPessoa(id);
if (pessoa == null) {
System.out.println("Pessoa não encontrada");
} else {
System.out.println("Pessoa Fisica:");
pessoa.exibir();
}
}
}
private static void excluirPessoa() {
String tipo;
int id;
System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
tipo = scanner.next().toUpperCase();
if (!tipo.equals("J") &&!tipo.equals("F")) {
System.out.println("Opção inválida!");
return;
}
System.out.println("Digite o id da Pessoa:");
id = scanner.nextInt();
if (tipo.equals("J")) {
PessoaJuridica pessoa = pessoaJuridicaDAO.getPessoa(id);
if (pessoa == null) {
System.out.println("Pessoa não encontrada");
} else {
```

```
pessoaFisicaDAO.excluir(id);
 }
 }
 }
 private static void exibirTodos() {
 System.out.println("Dados de Pessoas Fisicas:");
 List<PessoaFisica> todasPessoasFisicas = pessoaFisicaDAO.getPessoas();
 if (!todasPessoasFisicas.isEmpty()) {
 for (PessoaFisica pf : todasPessoasFisicas) {
 pf.exibir();
 System.out.println("");
 }
 } else {
 System.out.println("Nenhuma Pessoa Fisica");
 }
 System.out.println("------------------------------");
 System.out.println("Dados de Pessoas Juridicas:");
 List<PessoaJuridica> todasPessoasJuridicas = pessoaJuridicaDAO.getPessoas();
 if (!todasPessoasJuridicas.isEmpty()) {
for (PessoaJuridica pf : todasPessoasJuridicas) {
 pf.exibir();
 System.out.println("");
 }
 } else {
 System.out.println("Nenhuma Pessoa Juridica");
 }
 }
 }
 Pessoa.java
 package cadastrobd.model;
 public class Pessoa {
 protected int idPessoa;
 protected String nome;
 protected String logradouro;
 protected String cidade;
 protected String estado;
 protected String telefone;
 protected String email;
 // Construtor padrão
 public Pessoa() {}
 // Construtor completo
 public Pessoa(int idPessoa, String nome, String logradouro, String cidade, String estado,
 String telefone, String email) {
 this.idPessoa = idPessoa;
 this.nome = nome;
 this.logradouro = logradouro;
 this.cidade = cidade;
 this.estado = estado;
 this.telefone = telefone;
 this.email = email;
 }
 public void setId(int id) {
 this.idPessoa = id;
}
 public void setNome(String nome) {
 this.nome = nome;
 }
 // Método para exibir dados
 public void exibir() {
 System.out.println("ID: " + idPessoa);
 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("Email: " + email);
 }
 public int getId() {
 return idPessoa;
 }
 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;
}
}
PessoaFisica.java
package cadastrobd.model;
public class PessoaFisica extends Pessoa {
private String cpf;
// Construtor padrão
public PessoaFisica() {
super();
}
// Construtor completo
public PessoaFisica(int idPessoa, String nome, String logradouro, String cidade, String
estado, String telefone, String email, String cpf) {
super(idPessoa, nome, logradouro, cidade, estado, telefone, email);
this.cpf = cpf;
}
@Override
public void exibir() {
super.exibir();
System.out.println("CPF: " + cpf);
}
public String getCpf() {
return cpf;
}
public void setCpf(String cpf) {
this.cpf = cpf;
}
}
PessoaFisicaDAO.java
package cadastrobd.model;
import java.util.*;
import java.util.List;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Connection;
public class PessoaFisicaDAO {
private ConectorBD conector;
private SequenceManager sequenceManager;
public PessoaFisicaDAO(ConectorBD conector, SequenceManager sequenceManager) {
this.conector = conector;
this.sequenceManager = sequenceManager;
}
// Método para obter uma pessoa física pelo ID
public PessoaFisica getPessoa(int id) {
String sql = "SELECT * FROM Pessoa JOIN PessoaFisica ON Pessoa.idPessoa =
PessoaFisica.idPessoa WHERE Pessoa.idPessoa = ?";
try (
Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql);
) {
pstmt.setInt(1, id);
ResultSet rs = pstmt.executeQuery();
if (rs.next()) {
String nome = rs.getString("nome");
String logradouro = rs.getString("logradouro");
String cidade = rs.getString("cidade");
String estado = rs.getString("estado");
String telefone = rs.getString("telefone");
String email = rs.getString("email");
String cpf = rs.getString("cpf");
```

```
return new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email, cpf);
}
} catch (SQLException e) { // Catch SQLException specifically first
e.printStackTrace();
} catch (Exception e) { // Then catch the more general Exception
e.printStackTrace();
}
return null;
}
// Método para obter todas as pessoas físicas
public List<PessoaFisica> getPessoas() {
List<PessoaFisica> pessoas = new ArrayList<>();
String sql = "SELECT * FROM Pessoa JOIN PessoaFisica ON Pessoa.idPessoa =
PessoaFisica.idPessoa";
try (
Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql);
ResultSet rs = pstmt.executeQuery()
) {
while (rs.next()) {
int id = rs.getInt("idPessoa");
String nome = rs.getString("nome");
String logradouro = rs.getString("logradouro");
String cidade = rs.getString("cidade");
String estado = rs.getString("estado");
String telefone = rs.getString("telefone");
String email = rs.getString("email");
String cpf = rs.getString("cpf");
pessoas.add(new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email,
cpf));
}
} catch (SQLException e) { // Catch SQLException specifically first
e.printStackTrace();
} catch (Exception e) { // Then catch the more general Exception
e.printStackTrace();
}
return pessoas;
}
// Método para incluir uma nova pessoa física
public int incluir(PessoaFisica pessoaFisica) {
String sqlInsertPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
String sqlInsertPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, cpf) VALUES (?,
?)";
}
String sqlMaxIdPessoa = "SELECT MAX(idPessoa) AS MaxId FROM Pessoa";
try {
// Insert Pessoa
try (Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sqlInsertPessoa)) {
pstmt.setString(1, pessoaFisica.getNome());
pstmt.setString(2, pessoaFisica.getLogradouro());
pstmt.setString(3, pessoaFisica.getCidade());
pstmt.setString(4, pessoaFisica.getEstado());
pstmt.setString(5, pessoaFisica.getTelefone());
pstmt.setString(6, pessoaFisica.getEmail());
pstmt.executeUpdate();
PreparedStatement pstmtMaxId = conn.prepareStatement(sqlMaxIdPessoa);
ResultSet rsMaxId = pstmtMaxId.executeQuery();
if (rsMaxId.next()) {
int lastInsertedId = rsMaxId.getInt("MaxId");
// Now use this ID to insert PessoaFisica
try (PreparedStatement pstmt2 = conn.prepareStatement(sqlInsertPessoaFisica)) {
pstmt2.setInt(1, lastInsertedId);
pstmt2.setString(2, pessoaFisica.getCpf());
pstmt2.executeUpdate();
System.out.println("Pessoa Física incluída com sucesso.");
return lastInsertedId;
}
} else {
System.out.println("No se encontró el último ID insertado.");
}
}
} catch (SQLException e) {
e.printStackTrace();
```

```
 }
 return 0;
// Método para alterar os dados de uma pessoa física
 public void alterar(PessoaFisica pessoaFisica) {
 String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?,
 telefone=?, email=? WHERE idPessoa=?";
 String sql2 = "UPDATE PessoaFisica SET cpf=? WHERE idPessoa=?";
 try {
 // Insert Pessoa
 try (Connection conn = ConectorBD.getConnection();
 PreparedStatement pstmt = conn.prepareStatement(sql);
 PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {
 pstmt.setString(1, pessoaFisica.getNome());
 pstmt.setString(2, pessoaFisica.getLogradouro());
 pstmt.setString(3, pessoaFisica.getCidade());
 pstmt.setString(4, pessoaFisica.getEstado());
 pstmt.setString(5, pessoaFisica.getTelefone());
 pstmt.setString(6, pessoaFisica.getEmail());
 pstmt.setInt(7, pessoaFisica.getId());
 pstmt.executeUpdate();
 pstmt2.setString(1, pessoaFisica.getCpf());
 pstmt2.setInt(2, pessoaFisica.getId());
 pstmt2.executeUpdate();
 System.out.println("Dados da pessoa fisica alterados com sucesso.");
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }
 }
 // Método para excluir uma pessoa física
 public void excluir(int id) {
 String sql = "DELETE FROM PessoaFisica WHERE idPessoa=?";
 String sql2 = "DELETE FROM Pessoa WHERE idPessoa=?";
 try (
 Connection conn = ConectorBD.getConnection();
 PreparedStatement pstmt = conn.prepareStatement(sql);
 PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {
 pstmt.setInt(1, id);
pstmt.executeUpdate();
 pstmt2.setInt(1, id);
 pstmt2.executeUpdate();
 System.out.println("Pessoa Física excluída com sucesso.");
 } catch (SQLException e) {
 e.printStackTrace();
 }
 }
 }
 PessoaJuridica.java
 package cadastrobd.model;
 public class PessoaJuridica extends Pessoa {
 private String cnpj;
 // Construtor padrão
 public PessoaJuridica() {
 super();
 }
 // Construtor completo
 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;
 }
 @Override
 public void exibir() {
 super.exibir();
 System.out.println("CNPJ: " + cnpj);
 }
 public String getCnpj() {
 return cnpj;
}
 public void setCnpj(String cnpj) {
 this.cnpj = cnpj;
 }
 }
 PessoaJuridicaDAO.java
 package cadastrobd.model;
```

```
import java.util.*;
import java.util.List;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Connection;
public class PessoaJuridicaDAO {
private ConectorBD conector;
private SequenceManager sequenceManager;
public PessoaJuridicaDAO(ConectorBD conector, SequenceManager sequenceManager) {
this.conector = conector;
this.sequenceManager = sequenceManager;
}
// Método para obter uma pessoa física pelo ID
public PessoaJuridica getPessoa(int id) {
String sql = "SELECT * FROM Pessoa JOIN PessoaJuridica ON Pessoa.idPessoa =
PessoaJuridica.idPessoa WHERE Pessoa.idPessoa = ?";
try (
Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql);
) {
pstmt.setInt(1, id);
ResultSet rs = pstmt.executeQuery();
if (rs.next()) {
String nome = rs.getString("nome");
String logradouro = rs.getString("logradouro");
String cidade = rs.getString("cidade");
String estado = rs.getString("estado");
String telefone = rs.getString("telefone");
String email = rs.getString("email");
String cnpj = rs.getString("cnpj");
return new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone, email,
cnpj);
}
} catch (SQLException e) { // Catch SQLException specifically first
e.printStackTrace();
} catch (Exception e) { // Then catch the more general Exception
e.printStackTrace();
}
return null;
}
// Método para obter todas as pessoas físicas
public List<PessoaJuridica> getPessoas() {
List<PessoaJuridica> pessoas = new ArrayList<>();
String sql = "SELECT * FROM Pessoa JOIN PessoaJuridica ON Pessoa.idPessoa =
PessoaJuridica.idPessoa";
try (
Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql);
ResultSet rs = pstmt.executeQuery()
) {
while (rs.next()) {
int id = rs.getInt("idPessoa");
String nome = rs.getString("nome");
String logradouro = rs.getString("logradouro");
String cidade = rs.getString("cidade");
String estado = rs.getString("estado");
String telefone = rs.getString("telefone");
String email = rs.getString("email");
String cnpj = rs.getString("cnpj");
pessoas.add(new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone,
email, cnpj));
}
} catch (SQLException e) { // Catch SQLException specifically first
e.printStackTrace();
} catch (Exception e) { // Then catch the more general Exception
e.printStackTrace();
}
return pessoas;
}
// Método para incluir uma nova pessoa física
public int incluir(PessoaJuridica pessoaJuridica) {
String sqlInsertPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
```

```
telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
String sqlInsertPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa, cnpj) VALUES
(?, ?)";
String sqlMaxIdPessoa = "SELECT MAX(idPessoa) AS MaxId FROM Pessoa";
try {
// Insert Pessoa
try (Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sqlInsertPessoa)) {
pstmt.setString(1, pessoaJuridica.getNome());
pstmt.setString(2, pessoaJuridica.getLogradouro());
pstmt.setString(3, pessoaJuridica.getCidade());
pstmt.setString(4, pessoaJuridica.getEstado());
pstmt.setString(5, pessoaJuridica.getTelefone());
pstmt.setString(6, pessoaJuridica.getEmail());
pstmt.executeUpdate();
PreparedStatement pstmtMaxId = conn.prepareStatement(sqlMaxIdPessoa);
ResultSet rsMaxId = pstmtMaxId.executeQuery();
if (rsMaxId.next()) {
int lastInsertedId = rsMaxId.getInt("MaxId");
// Now use this ID to insert PessoaJuridica
try (PreparedStatement pstmt2 = conn.prepareStatement(sqlInsertPessoaJuridica))
{
pstmt2.setInt(1, lastInsertedId);
pstmt2.setString(2, pessoaJuridica.getCnpj());
pstmt2.executeUpdate();
System.out.println("Pessoa Jurídica incluída com sucesso.");
return lastInsertedId;
}
} else {
System.out.println("No se encontró el último ID insertado.");
}
}
} catch (SQLException e) {
e.printStackTrace();
// Handle exception, such as logging the error or informing the user
}
return 0;
}
// Método para alterar os dados de uma pessoa física
public void alterar(PessoaJuridica pessoaJuridica) {
String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?,
telefone=?, email=? WHERE idPessoa=?";
String sql2 = "UPDATE PessoaJuridica SET cnpj=? WHERE idPessoa=?";
try {
// Insert Pessoa
try (Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql);
PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {
pstmt.setString(1, pessoaJuridica.getNome());
pstmt.setString(2, pessoaJuridica.getLogradouro());
pstmt.setString(3, pessoaJuridica.getCidade());
pstmt.setString(4, pessoaJuridica.getEstado());
pstmt.setString(5, pessoaJuridica.getTelefone());
pstmt.setString(6, pessoaJuridica.getEmail());
pstmt.setInt(7, pessoaJuridica.getId());
pstmt.executeUpdate();
pstmt2.setString(1, pessoaJuridica.getCnpj());
pstmt2.setInt(2, pessoaJuridica.getId());
pstmt2.executeUpdate();
System.out.println("Dados da pessoa jurídica alterados com sucesso.");
}
} catch (SQLException e) {
e.printStackTrace();
// Handle exception, such as logging the error or informing the user
}
}
// Método para excluir uma pessoa física
public void excluir(int id) {
String sql = "DELETE FROM PessoaJuridica WHERE idPessoa=?";
String sql2 = "DELETE FROM Pessoa WHERE idPessoa=?";
try (
Connection conn = ConectorBD.getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql);
PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {
pstmt.setInt(1, id);
```

```
pstmt.executeUpdate();
 pstmt2.setInt(1, id);
 pstmt2.executeUpdate();
 System.out.println("Pessoa Jurídica excluída com sucesso.");
 } catch (SQLException e) {
 e.printStackTrace();
 }
 }
 }
 ConectorBD.java
 package cadastro.model.util;
 import java.sql.Connection;
 import java.sql.PreparedStatement;
 import java.sql.ResultSet;
 import java.sql.Statement;
 import java.sql.DriverManager;
 import java.sql.SQLException;
 public class ConectorBD {
 // Método para obter uma conexão com o banco de dados
 public static Connection getConnection() throws SQLException {
 String url =
 "jdbc:sqlserver://localhost:1433;databaseName=Loja3;trustServerCertificate=true";
 String usuario = "loja";
 String senha = "loja";
return DriverManager.getConnection(url, usuario, senha);
 }
 // Método para obter um PreparedStatement
 public static PreparedStatement getPrepared(String sql) throws Exception {
 try (Connection connection = getConnection()) {
 return connection.prepareStatement(sql);
 }
 }
 // Método para obter um ResultSet
 public static ResultSet getSelect(String sql) throws Exception {
 try (Connection connection = getConnection(); Statement statement =
 connection.createStatement()) {
 return statement.executeQuery(sql);
 }
 }
 // Métodos sobrecarregados para fechar Statement, ResultSet e Connection
 public static void close(Statement statement) {
 if (statement != null) {
 try {
 statement.close();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 }
 public static void close(ResultSet resultSet) {
 if (resultSet != null) {
 try {
 resultSet.close();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 }
 public static void close(Connection connection) {
 if (connection != null) {
 try {
 connection.close();
 } catch (Exception e) {
e.printStackTrace();
 }
 }
 }
 }
 SequenceManager.java
 package cadastro.model.util;
 import java.sql.Connection;
 import java.sql.PreparedStatement;
 import java.sql.SQLException;
 import java.sql.ResultSet;
```

```
public class SequenceManager {
// Método para obter o próximo valor de uma sequência
public static long getValue(String sequenceName) throws SQLException {
try (Connection connection = ConectorBD.getConnection();
PreparedStatement preparedStatement = connection.prepareStatement("SELECT
nextval('" + sequenceName + "')")) {
try (ResultSet resultSet = preparedStatement.executeQuery()) {
if (resultSet.next()) {
long result = resultSet.getLong(1);
return result;
} else {
throw new SQLException("No result found");
}
}
} catch (SQLException e) {
throw new RuntimeException(e);
}
}
}
```

---> RESULTADO

Output - **CadastroBD (run)**

```
run:
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Insira os dados...
Nome:
Joao
Logradouro:
rua 11, centro
Cidade:
riacho do sul
Estado:
PA
Telefone:
1212-1212
Email:
joao@gmail.com
CPF:
11111111111
Pessoa Fisica incluída com sucesso.
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
5
Dados de Pessoas Fisicas:
ID: 17
Nome: Joao
Logradouro: rua 11, centro
Cidade: riacho do sul
Estado: PA
Telefone: 1212-1212
Email: joao@gmail.com
CPF: 11111111111
```

---> ANÁLISE E CONCLUSÃO

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

Persistência em arquivo envolve salvar dados diretamente em arquivos no sistema de arquivos. Já a persistência em banco de dados armazena dados em um sistema de gerenciamento de banco de dados (SGBD). Isso oferece melhor organização, segurança, integridade referencial e desempenho nas operações de leitura e escrita.

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

Com lambda, é possível passar blocos de código como argumentos para métodos, eliminando a necessidade de classes anônimas e tornando o código mais legível, trazendo um grande avanço na simplicidade e concisão do código.

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

O método static pertence à própria classe, em vez de uma instância da classe. Quando utilizamos um método no main, ele não tem acesso a nenhuma instância de objeto, então precisa ser identificado como estático para ser chamado diretamente pela classe. Assim, o método static pode ser acessado sem criar um objeto, o que é essencial para iniciar a execução do programa.