UNIVERSIDADE ESTÁCIO DE SÁ

PROGRAMAÇÃO FULL STACK

PÓLO NORTE SHOPPING

CLEYDSON ASSIS COELHO

MISSÃO PRÁTICA 3 MUNDO 3
PROCEDIMENTO 1 e 2

RPG0016 - BackEnd sem banco não tem

RIO DE JANEIRO

2025

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.

MainConsole.java

```
package cadastrobd.model;

import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;

public class MainConsole {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
        boolean running = true;

        while (running) {
            System.out.println("\nMenu:");
            System.out.println("1 - Incluir");
            System.out.println("2 - Alterar");
            System.out.println("3 - Excluir");
            System.out.println("4 - Exibir por ID");
            System.out.println("5 - Exibir todos");
            System.out.println("0 - Sair");
            System.out.print("Opcao: ");


            int opc = Integer.parseInt(sc.nextLine());
            try {
                switch (opc) {
                    case 1:
                        System.out.print("Pessoa Física (F) ou Jurídica (J)? ");
                        String tipoInc = sc.nextLine().trim().toUpperCase();
                        if (tipoInc.equals("F")) {
                            System.out.print("Nome: ");
                            String nomeF = sc.nextLine();
                            System.out.print("Logradouro: ");
                            String logF = sc.nextLine();
                            System.out.print("Telefone: ");
                            String telF = sc.nextLine();
                            System.out.print("CPF: ");
                            String cpf = sc.nextLine();
                            PessoaFisica pf = new PessoaFisica(0, nomeF, logF, null, null, telF, null, cpf);
                            pfDAO.incluir(pf);
                            System.out.println("Incluída PF com ID=" + pf.getId());
                        } else {
                            System.out.print("Nome: ");
                            String nomeJ = sc.nextLine();
```

```
            System.out.print("Logradouro: ");
            String logJ = sc.nextLine();
            System.out.print("Telefone: ");
            String telJ = sc.nextLine();
            System.out.print("CNPJ: ");
            String cnpj = sc.nextLine();
            PessoaJuridica pj = new PessoaJuridica(0, nomeJ, logJ, null, null, telJ, null, cnpj);
            pjDAO.incluir(pj);
            System.out.println("Incluída PJ com ID=" + pj.getId());
        }
        break;

    case 2:
        System.out.print("Pessoa Física (F) ou Jurídica (J)? ");
        String tipoAlt = sc.nextLine().trim().toUpperCase();
        System.out.print("ID: ");
        int idAlt = Integer.parseInt(sc.nextLine());
        if (tipoAlt.equals("F")) {
            PessoaFisica existingF = pfDAO.getPessoa(idAlt);
            if (existingF != null) {
                System.out.print("Novo nome: ");
                existingF.setNome(sc.nextLine());
                System.out.print("Novo logradouro: ");
                existingF.setLogradouro(sc.nextLine());
                System.out.print("Novo telefone: ");
                existingF.setTelefone(sc.nextLine());
                System.out.print("Novo CPF: ");
                existingF.setCpf(sc.nextLine());
                pfDAO.alterar(existingF);
                System.out.println("Alterada PF ID=" + idAlt);
            } else {
                System.out.println("PF não encontrada.");
            }
        } else {
            PessoaJuridica existingJ = pjDAO.getPessoa(idAlt);
            if (existingJ != null) {
                System.out.print("Novo nome: ");
                existingJ.setNome(sc.nextLine());
                System.out.print("Novo logradouro: ");
                existingJ.setLogradouro(sc.nextLine());
                System.out.print("Novo telefone: ");
                existingJ.setTelefone(sc.nextLine());
                System.out.print("Novo CNPJ: ");
                existingJ.setCnpj(sc.nextLine());
                pjDAO.alterar(existingJ);
                System.out.println("Alterada PJ ID=" + idAlt);
            } else {
                System.out.println("PJ não encontrada.");
```

```
            }
        }
        break;

    case 3:
        System.out.print("Pessoa Física (F) ou Jurídica (J)? ");
        String tipoExc = sc.nextLine().trim().toUpperCase();
        System.out.print("ID: ");
        int idExc = Integer.parseInt(sc.nextLine());
        if (tipoExc.equals("F")) {
            pfDAO.excluir(idExc);
            System.out.println("Excluída PF ID=" + idExc);
        } else {
            pjDAO.excluir(idExc);
            System.out.println("Excluída PJ ID=" + idExc);
        }
        break;

    case 4:
        System.out.print("Pessoa Física (F) ou Jurídica (J)? ");
        String tipoExb = sc.nextLine().trim().toUpperCase();
        System.out.print("ID: ");
        int idExb = Integer.parseInt(sc.nextLine());
        if (tipoExb.equals("F")) {
            PessoaFisica pfRes = pfDAO.getPessoa(idExb);
            if (pfRes != null) pfRes.exibir();
            else System.out.println("PF não encontrada.");
        } else {
            PessoaJuridica pjRes = pjDAO.getPessoa(idExb);
            if (pjRes != null) pjRes.exibir();
            else System.out.println("PJ não encontrada.");
        }
        break;

    case 5:
        System.out.print("Listar Todos — PF ou PJ? ");
        String tipoAll = sc.nextLine().trim().toUpperCase();
        if (tipoAll.equals("F")) {
            List<PessoaFisica> allF = pfDAO.getPessoas();
            allF.forEach(Pessoa::exibir);
        } else {
            List<PessoaJuridica> allJ = pjDAO.getPessoas();
            allJ.forEach(Pessoa::exibir);
        }
        break;

    case 0:
        running = false;
```

```
                    break;

                default:
                    System.out.println("Opção inválida.");
            }
        } catch (SQLException e) {
            System.err.println("Erro: " + e.getMessage());
        }
    }

    sc.close();
    System.out.println("Programa encerrado.");
  }
}
```

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 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; }

    public void exibir() {
        System.out.printf("ID: %d, Nome: %s, Endereço: %s, %s-%s, Tel: %s, Email: %s%n",
                id, nome, logradouro, cidade, estado, telefone, email);
    }
}
```

PessoaFisica.java

```
package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    public PessoaFisica() { }

    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.printf("CPF: %s%n", cpf);
    }
}
```

PessoaFisicaDAO.java

```
package cadastrobd.model;

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

public class PessoaFisicaDAO {

   public void incluir(PessoaFisica pf) throws SQLException {
      String sqlPessoa =
         "INSERT INTO Pessoa (id_pessoa, nome, logradouro, telefone) " +
         "VALUES (?, ?, ?, ?)";
      String sqlFisica =
         "INSERT INTO PessoaFisica (id_pessoa, cpf) VALUES (?, ?)";

      int id = SequenceManager.getValue("seq_pessoa");
      pf.setId(id);

      try (Connection conn = ConectorBD.getConnection();
           PreparedStatement ps1 = conn.prepareStatement(sqlPessoa);
           PreparedStatement ps2 = conn.prepareStatement(sqlFisica)) {

         // Pessoa
         ps1.setInt(1, pf.getId());
         ps1.setString(2, pf.getNome());
         ps1.setString(3, pf.getLogradouro());
         ps1.setString(4, pf.getTelefone());
         ps1.executeUpdate();

         // PessoaFisica
         ps2.setInt(1, pf.getId());
         ps2.setString(2, pf.getCpf());
         ps2.executeUpdate();
      }
   }

   public void alterar(PessoaFisica pf) throws SQLException {
      String sqlPessoa =
         "UPDATE Pessoa SET nome=?, logradouro=?, telefone=? " +
         "WHERE id_pessoa=?";
      String sqlFisica =
         "UPDATE PessoaFisica SET cpf=? WHERE id_pessoa=?";

      try (Connection conn = ConectorBD.getConnection();
           PreparedStatement ps1 = conn.prepareStatement(sqlPessoa);
           PreparedStatement ps2 = conn.prepareStatement(sqlFisica)) {
```

```
        // Pessoa
        ps1.setString(1, pf.getNome());
        ps1.setString(2, pf.getLogradouro());
        ps1.setString(3, pf.getTelefone());
        ps1.setInt(4, pf.getId());
        ps1.executeUpdate();

        // PessoaFisica
        ps2.setString(1, pf.getCpf());
        ps2.setInt(2, pf.getId());
        ps2.executeUpdate();
    }
}

public void excluir(int id) throws SQLException {
    String sqlFisica = "DELETE FROM PessoaFisica WHERE id_pessoa=?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE id_pessoa=?";

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement ps1 = conn.prepareStatement(sqlFisica);
         PreparedStatement ps2 = conn.prepareStatement(sqlPessoa)) {

        ps1.setInt(1, id);
        ps1.executeUpdate();

        ps2.setInt(1, id);
        ps2.executeUpdate();
    }
}

public PessoaFisica getPessoa(int id) throws SQLException {
    String sql =
        "SELECT p.id_pessoa, p.nome, p.logradouro, p.telefone, pf.cpf " +
        "FROM Pessoa p " +
        "JOIN PessoaFisica pf ON p.id_pessoa = pf.id_pessoa " +
        "WHERE p.id_pessoa = ?";

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setInt(1, id);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return new PessoaFisica(
                    rs.getInt("id_pessoa"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    null,   // cidade não existe
                    null,   // estado não existe
```

```
                    rs.getString("telefone"),
                    null,   // email não existe
                    rs.getString("cpf")
                );
            }
        }
    }
    return null;
  }

  public List<PessoaFisica> getPessoas() throws SQLException {
    String sql =
      "SELECT p.id_pessoa, p.nome, p.logradouro, p.telefone, pf.cpf " +
      "FROM Pessoa p " +
      "JOIN PessoaFisica pf ON p.id_pessoa = pf.id_pessoa";

    List<PessoaFisica> lista = new ArrayList<>();
    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql);
         ResultSet rs = ps.executeQuery()) {

      while (rs.next()) {
        lista.add(new PessoaFisica(
          rs.getInt("id_pessoa"),
          rs.getString("nome"),
          rs.getString("logradouro"),
          null,
          null,
          rs.getString("telefone"),
          null,
          rs.getString("cpf")
        ));
      }
    }
    return lista;
  }
}
```

PessoaJuridica

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
  private String cnpj;

  public PessoaJuridica() { }

  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.printf("CNPJ: %s%n", cnpj);
    }
}
```

PessoaJuricaDAO.java

```
package cadastrobd.model;

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

public class PessoaJuridicaDAO {

    public void incluir(PessoaJuridica pj) throws SQLException {
        String sqlPessoa =
            "INSERT INTO Pessoa (id_pessoa, nome, logradouro, telefone) " +
            "VALUES (?, ?, ?, ?)";
        String sqlJuridica =
            "INSERT INTO PessoaJuridica (id_pessoa, cnpj) VALUES (?, ?)";

        int id = SequenceManager.getValue("seq_pessoa");
        pj.setId(id);

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement ps1 = conn.prepareStatement(sqlPessoa);
             PreparedStatement ps2 = conn.prepareStatement(sqlJuridica)) {

            // Pessoa
            ps1.setInt(1, pj.getId());
            ps1.setString(2, pj.getNome());
            ps1.setString(3, pj.getLogradouro());
            ps1.setString(4, pj.getTelefone());
            ps1.executeUpdate();

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

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

    public void alterar(PessoaJuridica pj) throws SQLException {
        String sqlPessoa =
            "UPDATE Pessoa SET nome=?, logradouro=?, telefone=? " +
            "WHERE id_pessoa=?";
        String sqlJuridica =
            "UPDATE PessoaJuridica SET cnpj=? WHERE id_pessoa=?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement ps1 = conn.prepareStatement(sqlPessoa);
             PreparedStatement ps2 = conn.prepareStatement(sqlJuridica)) {

            // Pessoa
            ps1.setString(1, pj.getNome());
            ps1.setString(2, pj.getLogradouro());
            ps1.setString(3, pj.getTelefone());
            ps1.setInt(4, pj.getId());
            ps1.executeUpdate();

            // PessoaJuridica
            ps2.setString(1, pj.getCnpj());
            ps2.setInt(2, pj.getId());
            ps2.executeUpdate();
        }
    }

    public void excluir(int id) throws SQLException {
        String sqlJuridica = "DELETE FROM PessoaJuridica WHERE id_pessoa=?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE id_pessoa=?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement ps1 = conn.prepareStatement(sqlJuridica);
             PreparedStatement ps2 = conn.prepareStatement(sqlPessoa)) {

            ps1.setInt(1, id);
            ps1.executeUpdate();

            ps2.setInt(1, id);
            ps2.executeUpdate();
        }
    }

    public PessoaJuridica getPessoa(int id) throws SQLException {
        String sql =
            "SELECT p.id_pessoa, p.nome, p.logradouro, p.telefone, pj.cnpj " +
            "FROM Pessoa p " +
```

```
        "JOIN PessoaJuridica pj ON p.id_pessoa = pj.id_pessoa " +
        "WHERE p.id_pessoa = ?";

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setInt(1, id);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return new PessoaJuridica(
                    rs.getInt("id_pessoa"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    null,
                    null,
                    rs.getString("telefone"),
                    null,
                    rs.getString("cnpj")
                );
            }
        }
    }
    return null;
}

public List<PessoaJuridica> getPessoas() throws SQLException {
    String sql =
        "SELECT p.id_pessoa, p.nome, p.logradouro, p.telefone, pj.cnpj " +
        "FROM Pessoa p " +
        "JOIN PessoaJuridica pj ON p.id_pessoa = pj.id_pessoa";

    List<PessoaJuridica> lista = new ArrayList<>();
    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql);
         ResultSet rs = ps.executeQuery()) {

        while (rs.next()) {
            lista.add(new PessoaJuridica(
                rs.getInt("id_pessoa"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                null,
                null,
                rs.getString("telefone"),
                null,
                rs.getString("cnpj")
            ));
        }
    }
    return lista;
```

```
    }
}
```

TesteConecção.java

```
package cadastrobd.model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class TesteConexao {
    public static void main(String[] args) {
        String url =
"jdbc:sqlserver://localhost:1433;databaseName=Loja;encrypt=true;trustServerCertificate=tr
ue;";
        String user = "loja";
        String pass = "loja";

        try (Connection conn = DriverManager.getConnection(url, user, pass)) {
            System.out.println("OK: conexão estabelecida!");
        } catch (SQLException e) {
            System.err.println("ERRO: " + e.getMessage());
        }
    }
}
```

ConectorBD.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=tru
e;";
    private static final String USER = "loja";
    private static final String PASS = "loja";

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

    public static PreparedStatement getPrepared(Connection conn, String sql) throws
SQLException {
```

```
        return conn.prepareStatement(sql);
    }

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

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

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

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

SequenceManager.java

```
package cadastrobd.model.util;

import java.sql.*;

public class SequenceManager {
    public static int getValue(String sequenceName) throws SQLException {
        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS seq";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql);
             ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return rs.getInt("seq");
            }
        }
        return -1;
    }
}
```

Resultados:

Menu:

1 - Incluir

2 - Alterar

3 - Excluir

4 - Exibir por ID

5 - Exibir todos

0 - Sair

Opcao:

O Menu funcional sem erros incluindo deletando e exibindo valores do banco de dados.