Universidade Estácio de Sá

Polo Bangu - Rio de Janeiro

- **Curso:** Desenvolvimento Full-Stack
- **Disciplina:** Back-end Sem Banco Não Tem
- **Semestre:** 3
- **Turma:** 2024.2
- **Aluna:** Clara Martins Azevedo

---

1. Missão Prática - Nível 3

2. Objetivos da Missão 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.

3. Códigos do Projeto:

- Pessoa.java:

```
package cadastrobd.model;

/**
 *
 * @author okidata
 */
public class Pessoa {

    private int id;
    private String nome;
    private String locadouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public Pessoa() {
        id = 0;
        nome = "";
        locadouro = "";
        cidade = "";
        estado = "";
        telefone = "";
        email = "";
    }

    public Pessoa(int id, String nome, String locadouro, String cidade,
            String estado, String telefone, String email) {
        this.id = id;
        this.nome = nome;
        this.locadouro = locadouro;
        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 setId(String nome) {
        this.nome = nome;
    }

    public String getLocadouro() {
        return locadouro;
    }

    public void setLocadouro(String locadouro) {
        this.locadouro = locadouro;
```

```
    }

    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.println("ID: " + id);
        System.out.println("Nome: " + nome);
        System.out.println("Locadouro: " + locadouro);
        System.out.println("Cidade: " + cidade);
        System.out.println("Estado: " + estado);
        System.out.println("Telefone: " + telefone);
        System.out.println("Email: " + email);
    }

}
```

• PessoaFisica.java:

```
package cadastrobd.model;

/**
 *
 * @author okidata
 */
public class PessoaFisica extends Pessoa {

    private String cpf;

    public PessoaFisica() {
        cpf = "";
    }

    public PessoaFisica(int id, String nome, String locadouro, String cidade,
            String estado, String telefone, String email, String cpf) {
        this.cpf = cpf;
```

```
    }

    public String getCpf() {
        return cpf;
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("cpf: " + getCpf());
    }

}
```

• PessoaFisicaDAO.java:

```
package cadastrobd.model;

import cadastro.model.util.ConectorBD;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.sql.SQLException;
import java.sql.Connection;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author okidata
 */
public class PessoaFisicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaFisica getPessoa(int id) throws Exception {

        PessoaFisica Pessoa = null;
        String sql = "SELECT pessoaF.Pessoa_idPessoa, pessoaF.cpf, Pessoa.nome, Pessoa.locadouro,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email "
            + "FROM PessoaFisica pessoaF "
            + "INNER JOIN Pessoa p ON pessoaF.Pessoa_idPessoa = Pessoa.idPessoa "
            + "WHERE pessoaF.Pessoa_idPessoa = ?";
            PreparedStatement stmt = connection.getPrepared(sql);
            ResultSet resultado = stmt.executeQuery();
            while(resultado.next()){
                Pessoa = new PessoaFisica(resultado.getInt("idPessoa"),
                resultado.getString("nome"),
                resultado.getString("locadouro"),
                resultado.getString("cidade"),
                resultado.getString("estado"),
                resultado.getString("telefone"),
                resultado.getString("email"),
                resultado.getString("cpf"));
            } return Pessoa;
    }


    public ArrayList<PessoaFisica> getPessoas() throws SQLException {

        ArrayList<PessoaFisica> list = new ArrayList<>();
```

```
                String sql = "SELECT pessoaF.Pessoa_idPessoa, pessoaF.cpf, Pessoa.nome, Pessoa.locadouro,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email "
                    + "FROM PessoaFisica pessoaF "
                    + "INNER JOIN Pessoa p ON pessoaF.Pessoa_idPessoa = Pessoa.idPessoa";
                try (Connection conn = connection.getConnection(); PreparedStatement stmt =
                   conn.prepareStatement(sql); ResultSet rs = stmt.executeQuery()) {
                   while (rs.next()) {
                      list.add(new PessoaFisica(
                         rs.getInt("Pessoa_idPessoa"),
                         rs.getString("nome"),
                         rs.getString("locadouro"),
                         rs.getString("cidade"),
                         rs.getString("estado"),
                         rs.getString("telefone"),
                         rs.getString("email"),
                         rs.getString("cpf")));
                   }
                } catch (Exception ex) {
                   Logger.getLogger(PessoaFisicaDAO.class.getName()).log(Level.SEVERE, null, ex);
                }
                return list;
             }


             public void Incluir(Pessoa pessoa, Pessoa PessoaFisica) throws Exception {

                String sqlPessoa = "INSERT INTO Pessoa (nome, locadouro, cidade, estado, telefone, email) VALUES (?, ?, ?,
?, ?, ?, 'F')";
                String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoaFisica, cpf) VALUES (?, ?)";
                PreparedStatement p = connection.getPrepared(sqlPessoa);
                PreparedStatement pf = connection.getPrepared(sqlPessoaFisica);
                   p.setString(1, pessoa.getNome());
                   p.setString(2, pessoa.getEmail());
                   p.setString(3, pessoa.getLocadouro());
                   p.setString(4, pessoa.getCidade());
                   p.setString(5, pessoa.getEstado());
                   pf.setInt(6, PessoaFisica.getId());
                   pf.setInt(7, PessoaFisica.getId());
                   p.executeUpdate();
                   pf.executeUpdate();

             }


             public void alterar(Pessoa pessoa, Pessoa PessoaFisica) throws Exception {

                String sqlPessoa = "UPDATE Pessoa SET nome = ?, locadouro = ?, cidade = ?, estado = ?, telefone = ?, email
= ?, WHERE idPessoa = ?";
                String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoaFisica = ?";
                PreparedStatement p = connection.getPrepared(sqlPessoa);
                PreparedStatement pf = connection.getPrepared(sqlPessoaFisica);
                   p.setString(1, pessoa.getNome());
                   p.setString(2, pessoa.getLocadouro());
                   p.setString(3, pessoa.getCidade());
                   p.setString(4, pessoa.getEstado());
                   p.setString(5, pessoa.getTelefone());
                   p.setString(6, pessoa.getEmail());
                   pf.setInt(7, pessoa.getId());
                   p.executeUpdate();
                   pf.executeUpdate();

             }


             public void excluir(int id) throws Exception {

                String sqlpessoa = "DELETE FROM Pessoa WHERE idPessoa="+id;
                String sqlPessoaFisica = "DELETE FROM Pessoa_Fisica WHERE idPessoa="+id;
```

```
        PreparedStatement p = connection.getPrepared(sqlPessoaFisica);
        PreparedStatement pf = connection.getPrepared(sqlpessoa);
            p.execute();
            pf.execute();
    }



}
```

• PessoaJuridica.java:

```
package cadastrobd.model;

/**
 *
 * @author okidata
 */
public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica() {
        cnpj = "";
    }

    public PessoaJuridica(int id, String nome, String locadouro, String cidade,
            String estado, String telefone, String email, String cnpj) {
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return cnpj;
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("cnpj: " + getCnpj());
    }

}
```

• PessoaJuridicaDAO.java:

```
package cadastrobd.model;

import cadastro.model.util.ConectorBD;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.sql.SQLException;
import java.sql.Connection;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
```

```
 * @author okidata
 */
public class PessoaJuridicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaJuridica getPessoa(int id) throws Exception {

        PessoaJuridica Pessoa = null;
        String sql = "SELECT pessoaJ.Pessoa_idPessoa, pessoaJ.cpf, Pessoa.nome, Pessoa.locadouro,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email "
            + "FROM PessoaJuridica pessoaF "
            + "INNER JOIN Pessoa p ON pessoaF.Pessoa_idPessoa = Pessoa.idPessoa "
            + "WHERE pessoaF.Pessoa_idPessoa = ?";
            PreparedStatement stmt = connection.getPrepared(sql);
            ResultSet resultado = stmt.executeQuery();
            while(resultado.next()){
                Pessoa = new PessoaJuridica(resultado.getInt("idPessoa"),
                resultado.getString("nome"),
                resultado.getString("locadouro"),
                resultado.getString("cidade"),
                resultado.getString("estado"),
                resultado.getString("telefone"),
                resultado.getString("email"),
                resultado.getString("cpf"));
            } return Pessoa;
    }


    public ArrayList<PessoaJuridica> getPessoas() throws SQLException {

        ArrayList<PessoaJuridica> list = new ArrayList<>();
        String sql = "SELECT pessoaJ.Pessoa_idPessoa, pessoaJ.cnpj, Pessoa.nome, Pessoa.locadouro,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email "
                + "FROM PessoaJuridica pessoaF "
                + "INNER JOIN Pessoa p ON pessoaJ.Pessoa_idPessoa = Pessoa.idPessoa";
        try (Connection conn = connection.getConnection(); PreparedStatement stmt =
            conn.prepareStatement(sql); ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                list.add(new PessoaJuridica(
                    rs.getInt("Pessoa_idPessoa"),
                    rs.getString("nome"),
                    rs.getString("locadouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email"),
                    rs.getString("cnpj")));
            }
        } catch (Exception ex) {
            Logger.getLogger(PessoaJuridicaDAO.class.getName()).log(Level.SEVERE, null, ex);
        }
        return list;
    }


    public void Incluir(Pessoa pessoa, Pessoa PessoaJuridica) throws Exception {

        String sqlPessoa = "INSERT INTO Pessoa (nome, locadouro, cidade, estado, telefone, email) VALUES (?, ?, ?,
?, ?, ?, ?, 'F')";
        String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoaJuridica, cnpj) VALUES (?, ?)";
        PreparedStatement p = connection.getPrepared(sqlPessoa);
        PreparedStatement pj = connection.getPrepared(sqlPessoaJuridica);
            p.setString(1, pessoa.getNome());
            p.setString(2, pessoa.getEmail());
            p.setString(3, pessoa.getLocadouro());
            p.setString(4, pessoa.getCidade());
            p.setString(5, pessoa.getEstado());
```

```
            pj.setInt(6, PessoaJuridica.getId());
            pj.setInt(7, PessoaJuridica.getId());
            p.executeUpdate();
            pj.executeUpdate();

        }


        public void alterar(Pessoa pessoa, Pessoa PessoaJuridica) throws Exception {

            String sqlPessoa = "UPDATE Pessoa SET nome = ?, locadouro = ?, cidade = ?, estado = ?, telefone = ?, email
= ?, WHERE idPessoa = ?";
            String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cpf = ? WHERE idPessoaJuridica = ?";
            PreparedStatement p = connection.getPrepared(sqlPessoa);
            PreparedStatement pj = connection.getPrepared(sqlPessoaJuridica);
              p.setString(1, pessoa.getNome());
              p.setString(2, pessoa.getLocadouro());
              p.setString(3, pessoa.getCidade());
              p.setString(4, pessoa.getEstado());
              p.setString(5, pessoa.getTelefone());
              p.setString(6, pessoa.getEmail());
              pj.setInt(7, pessoa.getId());
              p.executeUpdate();
              pj.executeUpdate();

        }


        public void excluir(int id) throws Exception {

            String sqlpessoa = "DELETE FROM Pessoa WHERE idPessoa="+id;
            String sqlPessoaJuridica = "DELETE FROM Pessoa_Fisica WHERE idPessoa="+id;
            PreparedStatement p = connection.getPrepared(sqlPessoaJuridica);
            PreparedStatement pf = connection.getPrepared(sqlpessoa);
              p.execute();
              pf.execute();
        }

    }
```

• CadastroBD.java:

```
    package cadastrobd;

    /**
     *
     * @author okidata
     */
    public class CadastroBD {

        /**
         * @param args the command line arguments
         */
        public static void main(String[] args) {
            // TODO code application logic here
            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("=========================");
        }

    }
```

• ConectorBD.java:

```
package cadastro.model.util;

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

/**
 *
 * @author okidata
 */
public class ConectorBD {

    public Connection getConnection() throws Exception{
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Connection conn =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;");
            return conn;
        } catch (SQLException ex) {
            System.out.println("Driver do banco não encontrado.");
        }
            return null;
    }


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


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


    public void closeStatement(String sql) throws Exception{
      getPrepared(sql).close();
    }


    public void closeResult(PreparedStatement stmt)throws Exception{
        getSelect(stmt).close();
    }


    public void closeConnection() throws Exception{
      getConnection().close();
   }

}
```

• SequenceManager.java:

```
package cadastro.model.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
```

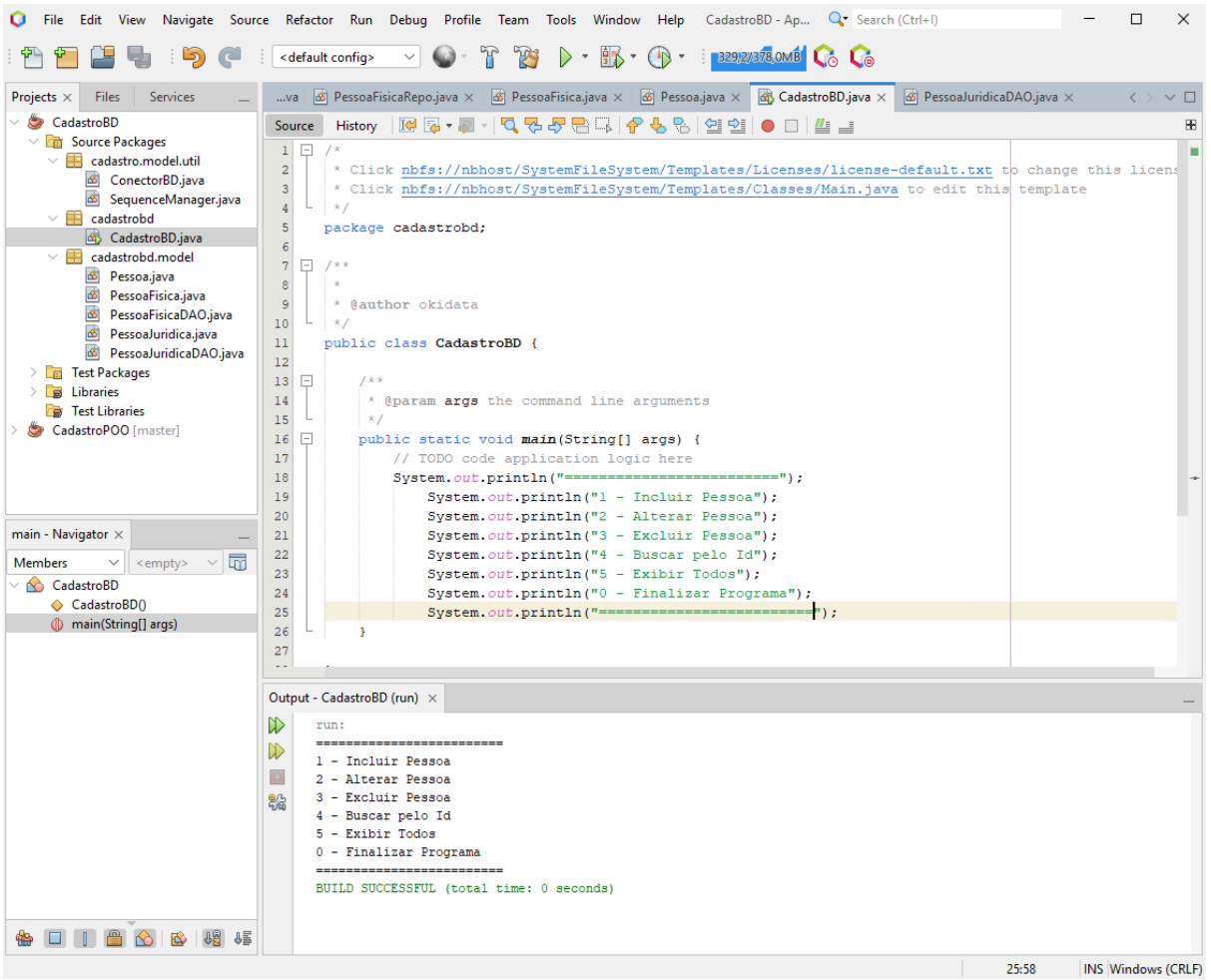
```
import java.sql.ResultSet;

/**
 *
 * @author okidata
 */
public class SequenceManager {

    public int getValue(String sequencia)throws Exception{
        Connection conn =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;");
        String sql = "SELECT NEXT VALUE FOR "+sequencia+" as proximoId";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql);
          rs.getInt("proximoId");
        return 0;
      }

}
```

4. Resultado da execução do código:



5. Análise e Conclusão:

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

Viabiliza a integração de aplicações Java com banco de dados, facilitando esse processo já que abstrai grande parte da complexidade do acesso ao banco de dados.

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

Quando uma consulta chega ao banco de dados ela passa por algumas preparações até ser processada, aí entra a diferença do Statement e do PreparedStatement. Com o uso PreparedStatement a consulta já vem preparada, agilizando o processo, enquanto no Statement não.

c. *Como o padrão DAO melhora a manutenibilidade do software?*

Usando o encapsulamento o padrão DAO isola parte do código, garantindo a segurança do resto da aplicação e consequentemente facilitando a manutenção, já que qualquer alteração na lógica terá um impacto mínimo no resto do código.

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

Normalmente com o uso de tabelas e a relações entre elas através da utilização da chave estrangeira.

1. Missão Prática - Nível 3

2. Objetivos da Missão 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.

3. Códigos do Projeto *(apresentados na primeira parte do relatório).*

4. Resultado da execução do código *(apresentados na primeira parte do relatório).*

5. Análise e Conclusão:

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

   Persistência em arquivo: Mais simples e mais eficiente, porém menos segura.

   Persistência em banco de dados: Estruturada, um pouco mais lenta porém mais segura.

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

Viabilizando formas mais diretas e concisas para operações relativamente simples, reduzindo as linhas de código e consequentemente otimizando o tempo gasto para escreve-los.

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

Um método estático só pode chamar outros métodos estáticos, e como o método main é estático por padrão é necessário a utilização do “static” em métodos adicionados.