CENTRO UNIVERSITÁRIO ESTÁCIO DE SANTA CATARINA

Desenvolvimento Full Stack

Nome do Aluno: Matheus Oliveira de Jesus
Matrícula: 2023.10.10087-8
Disciplina: Back-end Sem Banco Não Tem Turma: RPG0016
Semestre Letivo: 2°/2024

Este relatório tem como objetivo apresentar códigos e resultados referente aos exercícios solicitados em: Missão Prática | Nível 1 | Mundo 3.

Respostas: 1º Procedimento | Mapeamento Objeto-Relacional e DAO

1 – Título da Prática: BackEnd sem banco não tem

2 – Objetivo 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.

3 – Códigos gerados:

1) Pessoa.java

```
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(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;
    }
}
```

2) PessoaFisica.java

```
public class PessoaFisica extends Pessoa {

private String cpf;

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 void imprimirInformacoes() {

    System.out.println("Estado: " + getEstado());

    System.out.println("Telefone: " + getTelefone());

    System.out.println("E-mail: " + getEmail());

    System.out.println("CPF: " + this.cpf);

    System.out.println("Id: " + getId());

    System.out.println("Nome: " + getNome());

    System.out.println("Logradouro: " + getLogradouro());

    System.out.println("Cidade: " + getCidade());

}


public String getCpf() {

    return cpf;

}


public void setCpf(String cpf) {

    this.cpf = cpf;

}

}
```

3) PessoaJuridica.java

```
public class PessoaJuridica extends Pessoa {

private String cnpj;

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 void imprimirInformacoes() {

    System.out.println("Estado: " + getEstado());

    System.out.println("Telefone: " + getTelefone());

    System.out.println("E-mail: " + getEmail());

    System.out.println("CNPJ: " + this.cnpj);

    System.out.println("Id: " + getId());

    System.out.println("Nome: " + getNome());

    System.out.println("Logradouro: " + getLogradouro());

    System.out.println("Cidade: " + getCidade());

}


public String getCnpj() {

    return cnpj;

}


public void setCnpj(String cnpj) {

    this.cnpj = cnpj;

}

}
```

4) PessoaFisicaDAO.java

```
package cadastrobd.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import cadastrobd.model.util.ConectorBD;

public class PessoaFisicaDAO {

private ConectorBD conector;

public PessoaFisicaDAO() {

this.conector = new ConectorBD();

  }


  public void incluir(PessoaFisica pessoaFisica) {

    String sql = "INSERT INTO Pessoa (nome, logradouro, numero, bairro, cidade,
estado) VALUES (?, ?, ?, ?, ?, ?)";

    String sqlPF = "INSERT INTO PessoaFisica (cpf, id) VALUES (?,
SCOPE_IDENTITY())";


    try (Connection con = conector.getConnection()) {

      try (PreparedStatement stmt = con.prepareStatement(sql)) {

        stmt.setString(1, pessoaFisica.getNome());

        stmt.setString(2, pessoaFisica.getLogradouro());

        stmt.setInt(3, pessoaFisica.getNumero());

        stmt.setString(4, pessoaFisica.getBairro());

        stmt.setString(5, pessoaFisica.getCidade());

        stmt.setString(6, pessoaFisica.getEstado());

        stmt.executeUpdate();

      }
```

```
        try (PreparedStatement stmtPF = con.prepareStatement(sqlPF)) {
            stmtPF.setString(1, pessoaFisica.getCpf());
            stmtPF.executeUpdate();
        }

        System.out.println("Pessoa física inserida no banco: " +
pessoaFisica.getNome());
    } catch (SQLException e) {
        throw new RuntimeException("Erro ao incluir pessoa física: " +
e.getMessage(), e);
    }
  }
}
```

5) PessoaJuridicaDAO.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 {

  private final ConectorBD conector;
  public PessoaJuridicaDAO() {
    this.conector = new ConectorBD();
  }
  public PessoaJuridica getPessoa(int id) {
    PessoaJuridica pessoaJuridica = null;
```

```
    String sql = "SELECT * FROM Pessoa p JOIN PessoaJuridica pj ON p.id = pj.id
WHERE p.id = ?";

    try (Connection conn = conector.getConnection();
         PreparedStatement stmt = conector.getPrepared(conn, sql)) {

        stmt.setInt(1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            pessoaJuridica = new PessoaJuridica(
                rs.getInt("id"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return pessoaJuridica;
}
```

```
public List<PessoaJuridica> getPessoas() {
    List<PessoaJuridica> lista = new ArrayList<>();
    String sql = "SELECT * FROM Pessoa p JOIN PessoaJuridica pj ON p.id = pj.id";

    try (Connection conn = conector.getConnection();
         PreparedStatement stmt = conector.getPrepared(conn, sql);
         ResultSet rs = stmt.executeQuery()) {

        while (rs.next()) {
            PessoaJuridica pessoaJuridica = new PessoaJuridica(
                rs.getInt("id"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
            lista.add(pessoaJuridica);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return lista;
```

```
  }

  public void incluir(PessoaJuridica pessoaJuridica) {
    String sqlPessoa = "INSERT INTO Pessoa (id, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (id, cnpj) VALUES (?, ?)";

    try (Connection conn = conector.getConnection()) {
      conn.setAutoCommit(false);

      try (PreparedStatement stmtPessoa = conector.getPrepared(conn, sqlPessoa);
           PreparedStatement stmtPessoaJuridica = conector.getPrepared(conn, sqlPessoaJuridica)) {

        int novoId = SequenceManager.getValue("seq_pessoa");
        stmtPessoa.setInt(1, novoId);
        stmtPessoa.setString(2, pessoaJuridica.getNome());
        stmtPessoa.setString(3, pessoaJuridica.getLogradouro());
        stmtPessoa.setString(4, pessoaJuridica.getCidade());
        stmtPessoa.setString(5, pessoaJuridica.getEstado());
        stmtPessoa.setString(6, pessoaJuridica.getTelefone());
        stmtPessoa.setString(7, pessoaJuridica.getEmail());
        stmtPessoa.executeUpdate();

        stmtPessoaJuridica.setInt(1, novoId);
        stmtPessoaJuridica.setString(2, pessoaJuridica.getCnpj());
        stmtPessoaJuridica.executeUpdate();
```

```
            conn.commit();

        } catch (SQLException e) {
            conn.rollback();
            e.printStackTrace();
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
  }

  public void alterar(PessoaJuridica pessoaJuridica) {
    String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?,
estado = ?, telefone = ?, email = ? WHERE id = ?";
    String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE id = ?";

    try (Connection conn = conector.getConnection();
         PreparedStatement stmtPessoa = conector.getPrepared(conn, sqlPessoa);
         PreparedStatement stmtPessoaJuridica = conector.getPrepared(conn,
sqlPessoaJuridica)) {

        stmtPessoa.setString(1, pessoaJuridica.getNome());
        stmtPessoa.setString(2, pessoaJuridica.getLogradouro());
        stmtPessoa.setString(3, pessoaJuridica.getCidade());
        stmtPessoa.setString(4, pessoaJuridica.getEstado());
        stmtPessoa.setString(5, pessoaJuridica.getTelefone());
        stmtPessoa.setString(6, pessoaJuridica.getEmail());
```

```
            stmtPessoa.setInt(7, pessoaJuridica.getId());
            stmtPessoa.executeUpdate();

            stmtPessoaJuridica.setString(1, pessoaJuridica.getCnpj());
            stmtPessoaJuridica.setInt(2, pessoaJuridica.getId());
            stmtPessoaJuridica.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void excluir(int id) {
        String sqlPessoa = "DELETE FROM Pessoa WHERE id = ?";
        String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE id = ?";

        try (Connection conn = conector.getConnection();
             PreparedStatement stmtPessoa = conector.getPrepared(conn, sqlPessoa);
             PreparedStatement stmtPessoaJuridica = conector.getPrepared(conn,
sqlPessoaJuridica)) {

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

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

        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

6) CadastroBDTeste.java

```
public class CadastroBDTeste {
    public static void main(String[] args) {
        PessoaFisica pessoaFisica = new PessoaFisica(
            21,
            "JJC",
            "Rua 11, Centro",
            "Riacho do Sul",
            "PA",
            "1212-1212",
            "jjc@riacho.com",
            "11111111111"
        );

        PessoaJuridica pessoaJuridica = new PessoaJuridica(
            1,
            "João Ltda",
            "Rua 11, Centro",
            "Riacho do Sul",
            "PA",
            "1111-1111",
            "joao@riacho.com",
            "11111111111111"
```

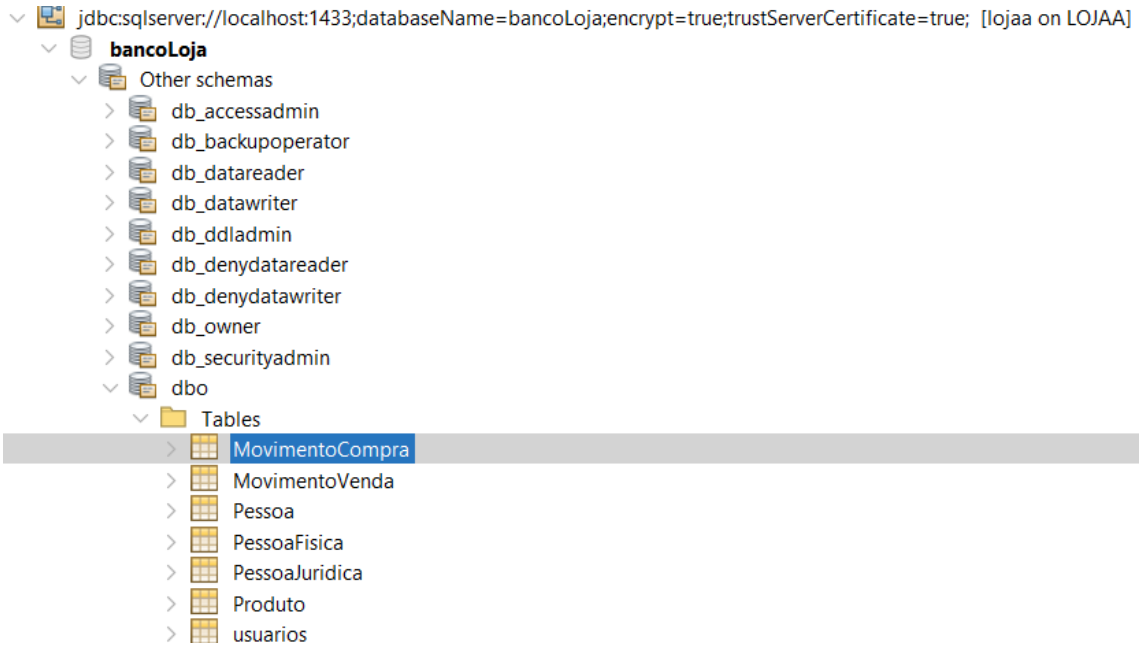
```
        );

        System.out.println("Dados de Pessoa Física:");
        pessoaFisica.imprimirInformacoes();

        System.out.println("\nDados de Pessoa Jurídica:");
        pessoaJuridica.imprimirInformacoes();
    }
}
```

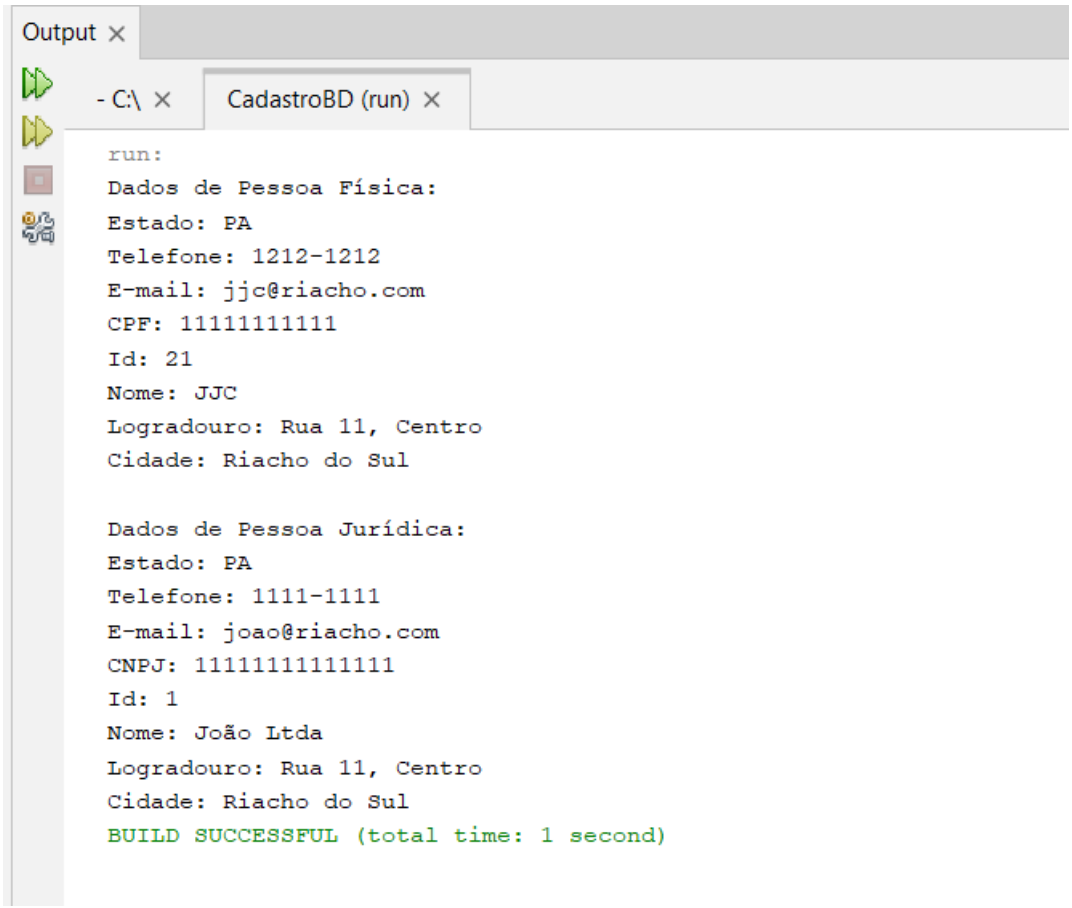
7) ConectorBD.java

```
package cadastrobd.model.util;

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

public class ConectorBD {
    public static Connection getConnection() throws SQLException {
        String url =
"jdbc:sqlserver://localhost:1433;databaseName=bancoLoja;encrypt=true;trustSe
rverCertificate=true;";
        String user = "lojaa";
        String password = "lojaa";

        Connection conn = DriverManager.getConnection(url, user, password);
        return conn;
```

```
    }


    public static PreparedStatement getPrepared(Connection conn, String sql)
throws SQLException {
        return conn.prepareStatement(sql);
    }
}
```

4 – Resultado dos códigos executados:





5 – Análise e Conclusão:

a. **Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?**
   A persistência em arquivo armazena dados em arquivos locais, como texto ou JSON, sendo simples e fácil de implementar, mas com pouca organização e mais difícil de consultar e manter.
   A persistência em banco de dados usa um sistema especializado para armazenar dados de forma estruturada e eficiente, permitindo consultas rápidas, transações e melhor segurança, sendo mais adequada para grandes volumes de dados ou sistemas complexos.

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

   O uso de operadores lambda no Java simplificou a impressão dos valores das entidades ao permitir que você crie funções anônimas de forma mais concisa e legível. Com lambdas, você pode substituir classes anônimas e loops complexos por expressões mais diretas, tornando o código mais enxuto e fácil de manter.

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

   Métodos acionados diretamente pelo método main precisam ser marcados como static porque o método main em Java é estático. Um método estático pertence à classe, e não a uma instância (objeto) dela. Como o main é o ponto de entrada do programa e é chamado pela JVM sem criar um objeto da classe, qualquer método que ele chamar diretamente também precisa ser estático para ser acessado sem uma instância. Se o método não for estático, seria necessário criar um objeto da classe para chamá-lo, o que o main não faz diretamente.