Backend sem banco não tem

Rogério Benedito Geraldo – Matricula 202110027921

Campus Nova Ourinhos
Backend sem banco não tem — 3° Semestre Letivo
Link do Github: https://github.com/Rogeriobg/backend-sem-banco-n-o-tem.git

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.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Códigos usados na prática:

ConectorDB.java

```
package cadastro.model.util;


/**

*

* @author rbgor

*/

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;
```

```
import java.sql.Statement;


public class ConectorBD {
   private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;";
   private static final String USUARIO = "loja";
   private static final String SENHA = "loja";



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



   public static PreparedStatement getPrepared(String sql) throws SQLException {
      return getConnection().prepareStatement(sql);
   }



   public static ResultSet getSelect(String sql) throws SQLException {
      Statement statement = getConnection().createStatement();
      return statement.executeQuery(sql);
   }



   public static void close(Statement statement) {
```

```
    if (statement != null) {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}


public static void close(ResultSet resultSet) {
    if (resultSet != null) {
        try {
            resultSet.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}


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

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

public static void main(String[] args) {
    try (Connection connection = getConnection()) {
        if (connection != null) {
            System.out.println("Conexão estabelecida com sucesso!");
        }
    } catch (SQLException e) {
        System.err.println("Erro  ao conectar ao banco de dados: " + e.getMessage());
    }
}
}
```

SequenceManager.java

```
package cadastro.model.util;

/**
*
* @author rbgor
*/
import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

public class SequenceManager {

    public static int getValue(String sequenceName) throws SQLException {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        int nextValue = -1;

        try {

            connection = ConectorBD.getConnection();


            String sql = "SELECT nextval(?) AS next_value";


            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(1, sequenceName);


            resultSet = preparedStatement.executeQuery();
```

```
            if (resultSet.next()) {
                nextValue = resultSet.getInt("next_value");
            }
        } finally {

            ConectorBD.close(resultSet);
            ConectorBD.close(preparedStatement);
            ConectorBD.close(connection);
        }

        return nextValue;
    }
}
```

CadastroDB.java

```
package cadastrobd;

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

    /**
     * @param args the command line arguments
     */
```

```
    public static void main(String[] args) {

    }

}
```

CadastroDBTeste.java

```
package cadastrobd;

/*
* 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 rbgor
*/

import cadastrodb.model.PessoaFisica;
import cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;

import java.sql.SQLException;
```

```
import java.util.List;

public class CadastroBDTeste {
    public static void main(String[] args) {

        PessoaFisica pessoaFisica = new PessoaFisica();
        pessoaFisica.setNome("Fulano");
        pessoaFisica.setLogradouro("Rua Teste, 123");
        pessoaFisica.setCidade("Cidade Teste");
        pessoaFisica.setEstado("TE");
        pessoaFisica.setTelefone("(00)  1234-5678");
        pessoaFisica.setEmail("fulano@teste.com");
        pessoaFisica.setCpf("12345678910");

        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

        try {

            pessoaFisicaDAO.incluir(pessoaFisica);



            pessoaFisica.setNome("Fulano  Alterado");
            pessoaFisicaDAO.alterar(pessoaFisica);



            List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
```

```
    System.out.println("Pessoas Físicas:");
    for (PessoaFisica pf : pessoasFisicas) {
        pf.exibir();
        System.out.println();
    }


    pessoaFisicaDAO.excluir(pessoaFisica.getId());
} catch (SQLException e) {
    e.printStackTrace();
}


PessoaJuridica pessoaJuridica = new PessoaJuridica();
pessoaJuridica.setNome("Empresa Teste");
pessoaJuridica.setLogradouro("Av. Teste, 456");
pessoaJuridica.setCidade("Cidade Teste");
pessoaJuridica.setEstado("TE");
pessoaJuridica.setTelefone("(00) 9876-5432");
pessoaJuridica.setEmail("empresa@teste.com");
pessoaJuridica.setCnpj("12345678000190");

PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

try {
```

```
            pessoaJuridicaDAO.incluir(pessoaJuridica);



            pessoaJuridica.setNome("Empresa  Alterada");
            pessoaJuridicaDAO.alterar(pessoaJuridica);



            List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
            System.out.println("\nPessoas Jurídicas:");
            for (PessoaJuridica pj : pessoasJuridicas) {
                pj.exibir();
                System.out.println();
            }



            pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Pessoa.java

```
package cadastrodb.model;

/**
*
* @author rbgor
*/
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.println("ID: " + id);
        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);
    }
}
```

PessoaJisica.java

```
package cadastrodb.model;
```

```
/**
*
* @author rbgor
*/
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;
    }


    @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 cadastrodb.model;

/**
*
* @author rbgor
*/
import cadastro.model.util.ConectorBD;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {
    public void incluir(PessoaFisica pessoa) throws SQLException {
        String sqlPessoa = "INSERT INTO Pessoa (Nome, Logradouro, Cidade, Estado, Telefone, Email) VALUES (?, ?, ?, ?, ?, ?)";
        String sqlPessoaFisica = "INSERT INTO PessoaFisica (PessoaID, Pessoa_PessoaID, CPF) VALUES (?, ?, ?)";
```

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

            try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS)) {
                stmtPessoa.setString(1, pessoa.getNome());
                stmtPessoa.setString(2, pessoa.getLogradouro());
                stmtPessoa.setString(3, pessoa.getCidade());
                stmtPessoa.setString(4, pessoa.getEstado());
                stmtPessoa.setString(5, pessoa.getTelefone());
                stmtPessoa.setString(6, pessoa.getEmail());
                stmtPessoa.executeUpdate();

                ResultSet rs = stmtPessoa.getGeneratedKeys();
                if (rs.next()) {
                    int pessoaId = rs.getInt(1);
                    pessoa.setId(pessoaId);

                    try (PreparedStatement stmtPessoaFisica =
conn.prepareStatement(sqlPessoaFisica)) {
                        stmtPessoaFisica.setInt(1, pessoaId);
                        stmtPessoaFisica.setInt(2, pessoaId);
                        stmtPessoaFisica.setString(3, pessoa.getCpf());
                        stmtPessoaFisica.executeUpdate();
                    }
                }
                conn.commit();
```

```
        } catch (SQLException e) {
            conn.rollback();
            throw e;
        }
    }
}

public void alterar(PessoaFisica pessoa) throws SQLException {
    String sqlPessoa = "UPDATE Pessoa SET Nome = ?, Logradouro = ?, Cidade = ?,
Estado = ?, Telefone = ?, Email = ? WHERE PessoaID = ?";
    String sqlPessoaFisica = "UPDATE PessoaFisica SET CPF = ? WHERE PessoaID
= ?";

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

        try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
            stmtPessoa.setString(1, pessoa.getNome());
            stmtPessoa.setString(2, pessoa.getLogradouro());
            stmtPessoa.setString(3, pessoa.getCidade());
            stmtPessoa.setString(4, pessoa.getEstado());
            stmtPessoa.setString(5, pessoa.getTelefone());
            stmtPessoa.setString(6, pessoa.getEmail());
            stmtPessoa.setInt(7, pessoa.getId());
            stmtPessoa.executeUpdate();
```

```
            try (PreparedStatement stmtPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
                stmtPessoaFisica.setString(1, pessoa.getCpf());
                stmtPessoaFisica.setInt(2, pessoa.getId());
                stmtPessoaFisica.executeUpdate();
            }
            conn.commit();
        } catch (SQLException e) {
            conn.rollback();
            throw e;
        }
    }
}

public List<PessoaFisica> getPessoas() throws SQLException {
    List<PessoaFisica> pessoas = new ArrayList<>();
    String sql = "SELECT p.PessoaID, p.Nome, p.Logradouro, p.Cidade, p.Estado, p.Telefone, p.Email, pf.CPF " +
                 "FROM Pessoa p JOIN PessoaFisica pf ON p.PessoaID = pf.PessoaID";

    try (Connection conn = ConectorBD.getConnection(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            PessoaFisica pessoa = new PessoaFisica();
            pessoa.setId(rs.getInt("PessoaID"));
            pessoa.setNome(rs.getString("Nome"));
            pessoa.setLogradouro(rs.getString("Logradouro"));
```

```
            pessoa.setCidade(rs.getString("Cidade"));
            pessoa.setEstado(rs.getString("Estado"));
            pessoa.setTelefone(rs.getString("Telefone"));
            pessoa.setEmail(rs.getString("Email"));
            pessoa.setCpf(rs.getString("CPF"));
            pessoas.add(pessoa);
        }
    }
    return pessoas;
}

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

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

        try (PreparedStatement stmtPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
            stmtPessoaFisica.setInt(1, id);
            stmtPessoaFisica.executeUpdate();

            try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
                stmtPessoa.setInt(1, id);
                stmtPessoa.executeUpdate();
            }
```

```
                conn.commit();
            } catch (SQLException e) {
                conn.rollback();
                throw e;
            }
        }
    }
}
```

PessoaJuridica.java

```
package cadastrodb.model;

/**
*
* @author rbgor
*/
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;
    }


    @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 cadastrodb.model;

/**
*
* @author rbgor
```

```
*/
import cadastro.model.util.ConectorBD;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {
   public void incluir(PessoaJuridica pessoa) throws SQLException {
      String sqlPessoa = "INSERT INTO Pessoa (Nome, Logradouro, Cidade, Estado, Telefone, Email) VALUES (?, ?, ?, ?, ?, ?)";
      String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (PessoaID, Pessoa_PessoaID, CNPJ) VALUES (?, ?, ?)";

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

         try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa, Statement.RETURN_GENERATED_KEYS)) {
            stmtPessoa.setString(1, pessoa.getNome());
            stmtPessoa.setString(2, pessoa.getLogradouro());
            stmtPessoa.setString(3, pessoa.getCidade());
            stmtPessoa.setString(4, pessoa.getEstado());
            stmtPessoa.setString(5, pessoa.getTelefone());
            stmtPessoa.setString(6, pessoa.getEmail());
            stmtPessoa.executeUpdate();

            ResultSet rs = stmtPessoa.getGeneratedKeys();
```

```
            if (rs.next()) {
                int pessoaId = rs.getInt(1);
                pessoa.setId(pessoaId);

                try (PreparedStatement stmtPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)) {
                    stmtPessoaJuridica.setInt(1, pessoaId);
                    stmtPessoaJuridica.setInt(2, pessoaId);
                    stmtPessoaJuridica.setString(3, pessoa.getCnpj());
                    stmtPessoaJuridica.executeUpdate();
                }
            }
            conn.commit();
        } catch (SQLException e) {
            conn.rollback();
            throw e;
        }
    }
}

public void alterar(PessoaJuridica pessoa) throws SQLException {
    String sqlPessoa = "UPDATE Pessoa SET Nome = ?, Logradouro = ?, Cidade = ?, Estado = ?, Telefone = ?, Email = ? WHERE PessoaID = ?";
    String sqlPessoaJuridica = "UPDATE PessoaJuridica SET CNPJ = ? WHERE PessoaID = ?";

    try (Connection conn = ConectorBD.getConnection()) {
```

```
            conn.setAutoCommit(false);

            try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
                stmtPessoa.setString(1, pessoa.getNome());
                stmtPessoa.setString(2, pessoa.getLogradouro());
                stmtPessoa.setString(3, pessoa.getCidade());
                stmtPessoa.setString(4, pessoa.getEstado());
                stmtPessoa.setString(5, pessoa.getTelefone());
                stmtPessoa.setString(6, pessoa.getEmail());
                stmtPessoa.setInt(7, pessoa.getId());
                stmtPessoa.executeUpdate();

                try (PreparedStatement stmtPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)) {
                    stmtPessoaJuridica.setString(1, pessoa.getCnpj());
                    stmtPessoaJuridica.setInt(2, pessoa.getId());
                    stmtPessoaJuridica.executeUpdate();
                }
                conn.commit();
            } catch (SQLException e) {
                conn.rollback();
                throw e;
            }
        }
    }

    public List<PessoaJuridica> getPessoas() throws SQLException {
```

```
        List<PessoaJuridica> pessoas = new ArrayList<>();
        String sql = "SELECT p.PessoaID, p.Nome, p.Logradouro, p.Cidade, p.Estado, p.Telefone, p.Email, pj.CNPJ " +
                     "FROM Pessoa p JOIN PessoaJuridica pj ON p.PessoaID = pj.PessoaID";

        try (Connection conn = ConectorBD.getConnection(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                PessoaJuridica pessoa = new PessoaJuridica();
                pessoa.setId(rs.getInt("PessoaID"));
                pessoa.setNome(rs.getString("Nome"));
                pessoa.setLogradouro(rs.getString("Logradouro"));
                pessoa.setCidade(rs.getString("Cidade"));
                pessoa.setEstado(rs.getString("Estado"));
                pessoa.setTelefone(rs.getString("Telefone"));
                pessoa.setEmail(rs.getString("Email"));
                pessoa.setCnpj(rs.getString("CNPJ"));
                pessoas.add(pessoa);
            }
        }
        return pessoas;
    }

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

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

            try (PreparedStatement stmtPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)) {
                stmtPessoaJuridica.setInt(1, id);
                stmtPessoaJuridica.executeUpdate();

                try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
                    stmtPessoa.setInt(1, id);
                    stmtPessoa.executeUpdate();
                }
                conn.commit();
            } catch (SQLException e) {
                conn.rollback();
                throw e;
            }
        }
    }
}
```

Resultados executando Run files em CadastroDBTeste.java:

```
run:
Pessoas Físicas:
ID: 7
```

Nome: Joao

Logradouro: Rua 12,casa 3, Quitanda

Cidade: Riacho do Sul

Estado: PA

Telefone: 1111-1111

Email: joao@riacho.com

CPF: 11111111111

ID: 15

Nome: JJC

Logradouro: Rua 11,Centro

Cidade: Riacho do Norte

Estado: PA

Telefone: 1212-1212

Email: jjc@riacho.com

CPF: 22222222222

ID: 16

Nome: JJC2

Logradouro: Rua 11,Centro B

Cidade: Riacho do Norte

Estado: PA

Telefone: 1313-1313

Email: jjc2@riacho.com

CPF: 33333333333

ID: 24

Nome: Fulano Alterado

Logradouro: Rua Teste, 123

Cidade: Cidade Teste

Estado: TE

Telefone: (00) 1234-5678

Email: fulano@teste.com

CPF: 12345678910

Pessoas Jurídicas:

ID: 17

Nome: JJC3

Logradouro: Rua 11,Centro C

Cidade: Riacho do Norte

Estado: PA

Telefone: 1414-1414

Email: jjc4@riacho.com

CNPJ: 44444444444444

ID: 18

Nome: JJC4

Logradouro: Rua 11,Centro D

Cidade: Riacho do Norte

Estado: PA

Telefone: 1515-1515

Email: jjc5@riacho.com

CNPJ: 55555555555555

ID: 25

Nome: Empresa Alterada

Logradouro: Av. Teste, 456

Cidade: Cidade Teste

Estado: TE

Telefone: (00) 9876-5432

Email: empresa@teste.com

CNPJ: 12345678000190

BUILD SUCCESSFUL (total time: 1 second)

A) Qual a importância dos componentes de middleware, como o JDBC?

Componentes de middleware, como o JDBC, são cruciais para a integração de sistemas, facilitando a comunicação entre aplicativos e bancos de dados, assegurando eficiência, escalabilidade e portabilidade.

B) Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

`Statement` executa consultas SQL simples e diretas. `PreparedStatement` é mais seguro e eficiente para consultas parametrizadas, prevenindo SQL injection e melhorando o desempenho em execuções repetidas.

C) Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO melhora a manutenibilidade do software ao separar a lógica de acesso a dados da lógica de negócios, facilitando alterações, testes e reuso do código.

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

Na herança em um modelo relacional, utilizamos mapeamento de tabelas, onde cada classe filha tem sua própria tabela ligada à tabela da classe pai via chave estrangeira.

2º Procedimento | Alimentando a Base

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

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

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

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

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

Conclusão

Elabore uma análise crítica da sua Missão Prática.