



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.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

---> 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 {
    public static void main(String[] args) {
        // Instanciando ConectorBD e SequenceManager (simulando a inicialização)
        ConectorBD conectorBD = new ConectorBD();
        SequenceManager sequenceManager = new SequenceManager();
        // Instanciando DAOs
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD,
        sequenceManager);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conectorBD,
        sequenceManager);
        // Operações com Pessoa Física
        int idPessoa = 0;
        String nome = "João";
        String logradouro = "Rua Exemplo";
        String cidade = "São Paulo";
        String estado = "SP";
        String telefone = "(11) 99999-9999";
        String email = "joao@example.com";
        String cpf = "1111666617";
        PessoaFisica pessoaFisica = new PessoaFisica(idPessoa, nome, logradouro, cidade,
        estado, telefone, email, cpf);
        idPessoa = pessoaFisicaDAO.incluir(pessoaFisica);
        System.out.println("Pessoa Física incluída com sucesso.");
        pessoaFisica.setId(idPessoa);
        pessoaFisica.setNome("Alterando nome");
        pessoaFisica.setCpf("E111666617");
        pessoaFisicaDAO.alterar(pessoaFisica); // Alterando os dados da pessoa física
        System.out.println("Dados da pessoa física alterados com sucesso.");
        List<PessoaFisica> todasPessoasFisicas = pessoaFisicaDAO.getPessoas();
        for (PessoaFisica pf : todasPessoasFisicas) {
            pf.exibir();
        }
        System.out.println(pessoaFisica.getId());
        pessoaFisicaDAO.excluir(pessoaFisica.getId()); // Excluindo a pessoa física
        System.out.println("Pessoa Física excluída com sucesso.");
        idPessoa = 21;
        nome = "Empresa XYZ";
        logradouro = "Rua Exemplo";
        cidade = "São Paulo";
        estado = "SP";
        telefone = "(11) 99999-9999";
        email = "XYZ@example.com";
        String cnpj = "000000000010";
```

```

// Operações com Pessoa Jurídica
PessoaJuridica pessoaJuridica = new PessoaJuridica(idPessoa, nome, logradouro,
cidade, estado, telefone, email, cnpj);
idPessoa = pessoaJuridicaDAO.incluir(pessoaJuridica);
System.out.println("Pessoa Jurídica incluída com sucesso.");
pessoaJuridica.setId(idPessoa);
pessoaJuridica.setNome("Alterando Empresa");
pessoaJuridica.setCnpj("E000000000010");
pessoaJuridicaDAO.alterar(pessoaJuridica); // Alterando os dados da pessoa jurídica
System.out.println("Dados da pessoa jurídica alterados com sucesso.");
List<PessoaJuridica> todasPessoasJuridicas = pessoaJuridicaDAO.getPessoas();
for (PessoaJuridica pj : todasPessoasJuridicas) {
pj.exibir();
}
pessoaJuridicaDAO.excluir(pessoaJuridica.getId()); // Excluindo a pessoa jurídica
System.out.println(pessoaFisica.getId());
System.out.println("Pessoa Jurídica excluída com sucesso.");
}

}

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.ConektorBD;
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 ConektorBD conector;
    private SequenceManager sequenceManager;
    public PessoaFisicaDAO(ConektorBD 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.id = ?";
        try (
            Connection conn = ConektorBD.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);
            }
            while (rs.next()) {
            }
        } 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();
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(PessoaFisica pessoaFisica) {
String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?,
telefone=?, email=? 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();
    }
} 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 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();
    } 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.id = ?";
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);
}
while (rs.next()) {
}
} 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();
            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();
        }
    } 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();
    } 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=Loja;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

The screenshot shows the MySQL Workbench interface. On the left, the 'Databases' tree view is expanded to show 'jdbcsqllserver://localhost:1433 [sa on dl]'. Under this, 'master' and 'Loja' are selected. 'Loja' contains 'Other schemas' which include 'dbo', 'db_accessadmin', 'db_backupoperator', 'db_databreader', 'db_datawriter', 'db_ddladmin', 'db_denydatareader', 'db_denydatawriter', 'db_owner', and 'db_securityadmin'. Under 'dbo', there are 'Tables' such as 'Movimento', 'Pessoa', 'Pessoafisica', 'Pessoajuridica', 'Produto', 'Usuario', and 'Views', 'Procedures', 'guest', and 'INFORMATION_SCHEMA'. Below 'Loja' are 'Views', 'Procedures', 'guest', and 'INFORMATION_SCHEMA'. At the bottom are 'sys', 'Loja2', 'model', and 'msdb'. On the right, a Java code editor displays the 'CadastroBD.java' file. The code imports various classes from the 'cadastro' package and defines a public class 'CadastroBD' with a main method. The main method instantiates 'ConectorBD' and 'SequenceManager' and prints some output. Below the code editor is an 'Output - CadastroBD (run)' window showing the printed output.

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastrobd;

/**
 *
 * @author Marvin
 */
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 {

    public static void main(String[] args) {
        // Instanciando ConectorBD e SequenceManager (simulando a inicialização)
        ConectorBD conectorBD = new ConectorBD();
        SequenceManager sequenceManager = new SequenceManager();
    }
}

Output - CadastroBD (run) ×
run:
Pessoa Física incluída com sucesso.
Dados da pessoa física alterados com sucesso.
ID: 9
Nome: Alterando nome
Logradouro: Rua Exemplo
Cidade: São Paulo
Estado: SP
Telefone: (11) 99999-9999
Email: joao@example.com
CPF: E111666617
9
Pessoa Física excluída com sucesso.
Pessoa Jurídica incluída com sucesso.
Dados da pessoa jurídica alterados com sucesso.
ID: 10
Nome: Alterando Empresa
Logradouro: Rua Exemplo
Cidade: São Paulo
Estado: SP
Telefone: (11) 99999-9999
Email: XYZ@example.com
CNPJ: E0000000000010
9
Pessoa Jurídica excluída com sucesso.
BUILD SUCCESSFUL (total time: 1 second)

```

The screenshot shows the terminal output for the 'CadastroBD' program. It includes the command 'run:', followed by several lines of text indicating successful operations: 'Pessoa Física incluída com sucesso.', 'Dados da pessoa física alterados com sucesso.', 'ID: 9', 'Nome: Alterando nome', 'Logradouro: Rua Exemplo', 'Cidade: São Paulo', 'Estado: SP', 'Telefone: (11) 99999-9999', 'Email: joao@example.com', 'CPF: E111666617', '9', 'Pessoa Física excluída com sucesso.', 'Pessoa Jurídica incluída com sucesso.', 'Dados da pessoa jurídica alterados com sucesso.', 'ID: 10', 'Nome: Alterando Empresa', 'Logradouro: Rua Exemplo', 'Cidade: São Paulo', 'Estado: SP', 'Telefone: (11) 99999-9999', 'Email: XYZ@example.com', 'CNPJ: E0000000000010', '9', 'Pessoa Jurídica excluída com sucesso.', and finally 'BUILD SUCCESSFUL (total time: 1 second)'.

---> ANÁLISE E CONCLUSÃO

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

Os Middlewares servem como uma ponte entre a aplicação e o banco de dados, simplificando e padronizando a forma como os dados são acessados e manipulados. Permitindo que aplicações Java se conectem a qualquer banco de dados, abstraindo os detalhes específicos de cada um.

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

O Statement é utilizado para executar consultas SQL estáticas em que a consulta não é repetida. Cada execução de uma instrução SQL com um Statement cria um novo caminho de execução. Enquanto o PreparedStatement é utilizado para consultas SQL que precisam ser executadas várias vezes com diferentes parâmetros. Ele pré-compila a consulta, tornando-a mais rápida para execuções subsequentes e oferece maior segurança

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

O padrão DAO separa a lógica de acesso, da lógica de negócios do software. Isso significa que qualquer alteração na forma como os dados são armazenados ou acessados não impacta diretamente o restante do código. Com DAO, você cria uma camada de abstração que facilita a modificação, teste e manutenção.

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

A herança pode ser refletida como uma Tabela única, que contém todas as subclasses com colunas adicionais para cada propriedade específica; Tabelas por classe, onde Cada classe (super e sub) tem sua própria tabela. e, como Tabela concreta, onde cada tabela representa uma subclasse concreta, duplicando as colunas comuns.