

| | |
|---|--|
|  | <p>Universidade Estácio</p> <p>Campus Polo Centro - Barão de Cocais – MG</p> <p>Curso de Desenvolvimento Full Stack</p> <p>Relatório da Missão Prática 3 – Mundo 3</p> |
| Disciplina: | RPG0016 - BackEnd sem banco não tem |
| Nome: | Wesley Borges do Carmo de Oliveira – 202305150171 |
| Turma: | 9001 – 3º Semestre |

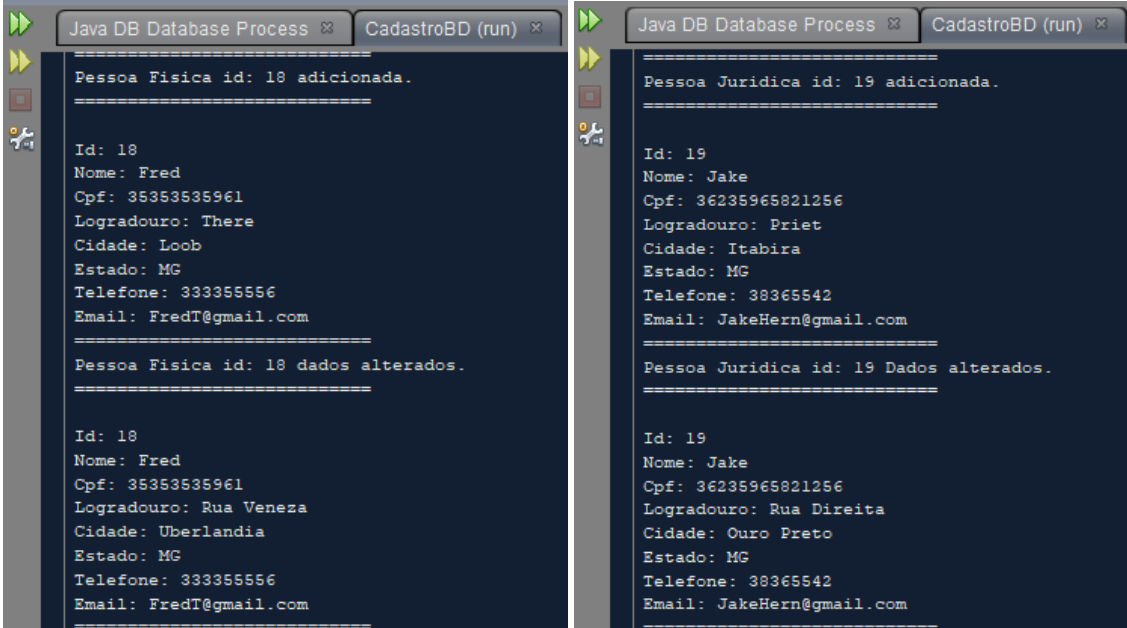
Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

Objetivos da prática

1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.
5. No final do exercício, o aluno terá criado um aplicativo cadastral com uso
6. do SQL Server na persistência de dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO.

Resultados



```

=====
Pessoa Fisica id: 18 adicionada.
=====

Id: 18
Nome: Fred
Cpf: 35353535961
Logradouro: There
Cidade: Loob
Estado: MG
Telefone: 333355556
Email: FredT@gmail.com
=====

Pessoa Fisica id: 18 dados alterados.
=====

Id: 18
Nome: Fred
Cpf: 35353535961
Logradouro: Rua Veneza
Cidade: Uberlandia
Estado: MG
Telefone: 333355556
Email: FredT@gmail.com
=====

```

```

=====
Pessoa Juridica id: 19 adicionada.
=====

Id: 19
Nome: Jake
Cpf: 36235965821256
Logradouro: Priet
Cidade: Itabira
Estado: MG
Telefone: 38365542
Email: JakeHern@gmail.com
=====

Pessoa Juridica id: 19 Dados alterados.
=====

Id: 19
Nome: Jake
Cpf: 36235965821256
Logradouro: Rua Direita
Cidade: Ouro Preto
Estado: MG
Telefone: 38365542
Email: JakeHern@gmail.com
=====

```

Figura 01: CadastroBDTest

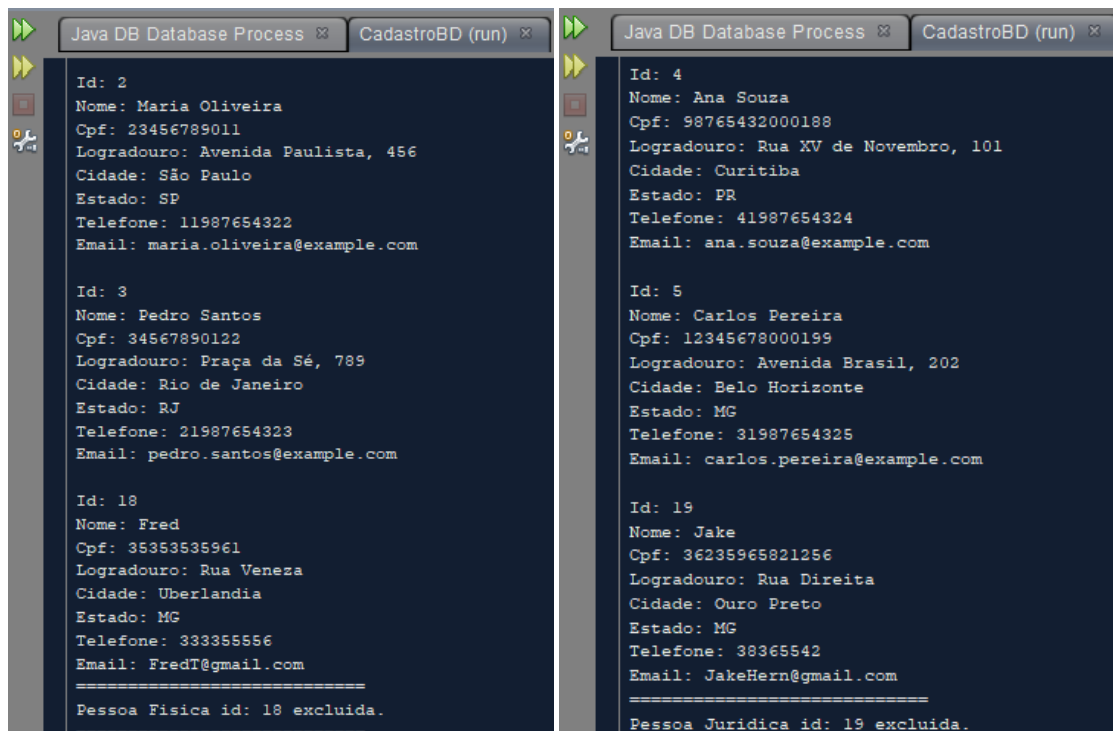


Figura 02: CadastroBDTest

Códigos solicitados

```
package cadastrbd.model;

public class Pessoa {
    private Integer id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public Pessoa(){}

    public Pessoa(Integer 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 void setId(int id){
        this.id = id;
    }
}
```

```

public Integer getId(){
    return id;
}
public void setNome(String nome){
    this.nome = nome;
}
public String getNome(){
    return nome;
}
public void setLogradouro(Stringlogradouro){
    this.logradouro = logradouro;
}

public String getLogradouro(){
    return logradouro;
}

public void setCidade(String cidade){
    this.cidade = cidade;
}

public String getCidade(){
    return cidade;
}
public void setEstado(String estado){
    this.estado = estado;
}
public String getEstado(){
    return estado;
}
public void setTelefone(String telefone){
    this.telefone = telefone;
}
public String getTelefone(){
    return telefone;
}
public void setEmail(String email){
    this.email = email;
}
public String getEmail(){
    return email;
}
public void exibir(){
    System.out.println("\nId: " + this.getId() +
        "\nNome: " + this.getNome() +
        "\nLogradouro: " + this.getLogradouro() +
        "\nCidade: " + this.getCidade() +
        "\nEstado: " + this.getEstado() +
        "\nTelefone: " + this.getTelefone() +
        "\nEmail: " + this.getEmail());
}
}

```

Figura 03: Pessoa

```

package cadastrobd.model;

public class PessoaFisica extends Pessoa {

    private String cpf;

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

    public String getCpf(){
        return cpf;
    }

    @Override
    public void exibir(){
        System.out.println("\nId: " + this.getId() +
            "\nNome: " + this.getNome() +
            "\nCpf: " + this.getCpf() +
            "\nLogradouro: " + this.getLogradouro() +
            "\nCidade: " + this.getCidade() +
            "\nEstado: " + this.getEstado() +
            "\nTelefone: " + this.getTelefone() +
            "\nEmail: " + this.getEmail());
    }
}

```

Figura 04: PessoaFisica

```

package cadastrobd.model;

public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica(Integer 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 setCnpj(String cnpj){
        this.cnpj = cnpj;
    }

    public String getCnpj(){
        return cnpj;
    }
}

```

```

@Override
public void exibir(){
    System.out.println("\nId: " + this.getId() +
        "\nNome: " + this.getNome() +
        "\nCpf: " + this.getCnpj() +
        "\nLogradouro: " + this.getLogradouro() +
        "\nCidade: " + this.getCidade() +
        "\nEstado: " + this.getEstado() +
        "\nTelefone: " + this.getTelefone() +
        "\nEmail: " + this.getEmail());
}
}

```

Figura 05: PessoaJuridica

```

package cadastro.model.util;

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

public class ConectorBD {

    public Connection getConnection() throws Exception {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        return
        DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=loja;e
ncrypt=true;trustServerCertificate =true;",
            "loja", "loja");
    }

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

    public ResultSet getSelect(String sql) throws Exception{
        PreparedStatement ps = getPrepared(sql);
        return ps.executeQuery();
    }

    public void close(Connection conn) throws Exception {
        if(conn != null){
            try{
                conn.close();
            }catch(SQLException e){
                e.printStackTrace();
            }
        }
    }
}

```

```

public void close(Statement st) throws Exception {
    if(st != null){
        try{
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}

public void close(ResultSet rs) throws Exception {
    if(rs != null){
        try{
            rs.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
}

```

Figura 06: ConectorBD

```

package cadastro.model.util;

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

public class SequenceManager {
    public int getValue(String sequence) throws SQLException, Exception{
        ResultSet rs = new ConectorBD().getSelect("SELECT NEXT VALUE FOR
".concat(sequence));
        if(rs.next()){
            return rs.getInt(1);
        }else{
            throw new SQLException("Value not achivable");
        }
    }
}

```

Figura 07: SequenceManager

```

package cadastrobd.model;

import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {

    private final ConectorBD connector;

```

```

public PessoaFisicaDAO(){
    connector = new ConectorBD();
}

public PessoaFisica getPessoa(int id) throws SQLException, Exception{
    String sql = "SELECT pf.Pessoa_idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pf.cpf "
        + "FROM PessoaFisica pf "
        + "INNER JOIN Pessoa p ON pf.Pessoa_idPessoa = p.idPessoa "
        + "WHERE pf.Pessoa_idPessoa = ?";
    PessoaFisica pessoa = null;
    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    try{
        con = connector.getConnection();
        stmt = con.prepareStatement(sql);
        stmt.setInt(1, id);
        rs = stmt.executeQuery();
        if(rs.next()){
            pessoa = new PessoaFisica(
                rs.getInt("Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")
            );
            return pessoa;
        }else{
            return null;
        }
    }finally{
        connector.close(rs);
        connector.close(stmt);
        connector.close(con);
    }
}

public List<PessoaFisica> getPessoas() throws SQLException, Exception{
    String sql = "SELECT pf.Pessoa_idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pf.cpf "
        + "FROM Pessoa p "
        + "INNER JOIN PessoaFisica pf ON p.idPessoa = pf.Pessoa_idPessoa;";
    List<PessoaFisica> lista = new ArrayList<>();
    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    try{
        con = connector.getConnection();
        stmt = con.prepareStatement(sql);
        rs = stmt.executeQuery();

```

```

        while(rs.next()){
            lista.add(new PessoaFisica(
                rs.getInt("Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")
            ));
        }
        return lista;
    }finally{
        connector.close(rs);
        connector.close(stmt);
        connector.close(con);
    }
}

public void incluir(PessoaFisica pessoa) throws SQLException, Exception{
    String sqlPessoa = "INSERT INTO Pessoa(idPessoa,nome,logradouro,cidade,estado,telefone,email) VALUES(?,?,?,?,?,?,?);";
    String sqlPessoaFisica = "INSERT INTO PessoaFisica(Pessoa_idPessoa,cpf)VALUES(?,?);";
    Connection con = null;
    PreparedStatement stmt = null;
    PreparedStatement stmtPf = null;
    try{
        con = connector.getConnection();
        con.setAutoCommit(false);

        SequenceManager sequenceManager = new SequenceManager();
        int idNovaPessoa = sequenceManager.getValue("orderPessoa");

        stmt = con.prepareStatement(sqlPessoa);
        stmt.setInt(1, idNovaPessoa);
        stmt.setString(2, pessoa.getNome());
        stmt.setString(3, pessoa.getLogradouro());
        stmt.setString(4, pessoa.getCidade());
        stmt.setString(5, pessoa.getEstado());
        stmt.setString(6, pessoa.getTelefone());
        stmt.setString(7, pessoa.getEmail());
        stmt.executeUpdate();

        pessoa.setId(idNovaPessoa);

        stmtPf = con.prepareStatement(sqlPessoaFisica);
        stmtPf.setInt(1, idNovaPessoa);
        stmtPf.setString(2, pessoa.getCpf());
        stmtPf.executeUpdate();

        con.commit();
    }catch(SQLException e){

```



```

        System.err.println("Adf: " + e);
    }finally{
        connector.close(stmtPf);
        connector.close(stmt);
        connector.close(con);
    }
}

public void alterar(PessoaFisica pessoa) throws SQLException, Exception{
    String sqlPessoa = "UPDATE PESSOA SET nome=?,logradouro=?,cidade=?,estado=?,telefone=?,email=? WHERE idPessoa = ?";
    String sqlPessoaFisica = "UPDATE PESSOAFISICA SET cpf=? WHERE Pessoa_idPessoa = ?";
    Connection con = null;
    PreparedStatement stmt = null;
    PreparedStatement stmtPf = null;
    try{
        con = connector.getConnection();
        stmt = con.prepareStatement(sqlPessoa);
        stmtPf = con.prepareStatement(sqlPessoaFisica);
        stmt.setString(1, pessoa.getNome());
        stmt.setString(2, pessoa.getLogradouro());
        stmt.setString(3, pessoa.getCidade());
        stmt.setString(4, pessoa.getEstado());
        stmt.setString(5, pessoa.getTelefone());
        stmt.setString(6, pessoa.getEmail());
        stmt.setInt(7, pessoa.getId());
        stmt.executeUpdate();
        stmtPf.setString(1, pessoa.getCpf());
        stmtPf.setInt(2, pessoa.getId());
        stmtPf.executeUpdate();
    }catch(Exception e){
        System.err.println("Alf: " + e);
    }finally{
        connector.close(stmtPf);
        connector.close(stmt);
        connector.close(con);
    }
}

public void excluir(int chave) throws SQLException, Exception{
    String sqlPessoa = "DELETE FROM PESSOA WHERE idPessoa = ?";
    String sqlPessoaFisica = "DELETE FROM PESSOAFISICA WHERE Pessoa_idPessoa = ?";
    Connection con = null;
    PreparedStatement stmt = null;
    PreparedStatement stmtPf = null;
    try{
        con = connector.getConnection();
        stmtPf = con.prepareStatement(sqlPessoaFisica);
        stmtPf.setInt(1, chave);
        stmtPf.executeUpdate();
        stmt = con.prepareStatement(sqlPessoa);
        stmt.setInt(1, chave);
    }
}

```

```

        stmt.executeUpdate();
    }catch(Exception e){
        System.err.println("Exf: " + e);
    }finally{
        connector.close(stmt);
        connector.close(stmtPf);
        connector.close(con);
    }
}
}
}

```

Figura 08: PessoaFisicaDAO

```

package cadastrobd.model;

import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {
    private final ConectorBD connector;

    public PessoaJuridicaDAO(){
        connector = new ConectorBD();
    }

    public PessoaJuridica getPessoa(int id) throws SQLException, Exception{
        String sql = "SELECT pj.Pessoa_idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pj.cnpj "
            + "FROM PessoaJuridica pj "
            + "INNER JOIN Pessoa p ON pj.Pessoa_idPessoa = p.idPessoa "
            + "WHERE pj.Pessoa_idPessoa = ?";
        PessoaJuridica pessoa = null;
        Connection con = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try{
            con = connector.getConnection();
            stmt = con.prepareStatement(sql);
            stmt.setInt(1, id);
            rs = stmt.executeQuery();
            if(rs.next()){
                pessoa = new PessoaJuridica(
                    rs.getInt("Pessoa_idPessoa"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email"),

```

```

        rs.getString("cnpj")
    );
    return pessoa;
}
else{
    return null;
}
}
}finally{
    connector.close(rs);
    connector.close(stmt);
    connector.close(con);
}
}
}

```

```

public List<PessoaJuridica> getPessoas() throws SQLException, Exception{
    String sql = "SELECT pj.Pessoa_idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pj.cnpj "
        + "FROM Pessoa p "
        + "INNER JOIN PessoaJuridica pj ON p.idPessoa = pj.Pessoa_idPessoa;";
    List<PessoaJuridica> lista = new ArrayList<>();
    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    try{
        con = connector.getConnection();
        stmt = con.prepareStatement(sql);
        rs = stmt.executeQuery();
        while(rs.next()){
            lista.add(new PessoaJuridica(
                rs.getInt("Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            ));
        }
        return lista;
    }finally{
        connector.close(rs);
        connector.close(stmt);
        connector.close(con);
    }
}
}

```

```

public void incluir(PessoaJuridica pessoa) throws SQLException, Exception{
    String sqlPessoa = "INSERT INTO
Pessoa(idPessoa,nome,logradouro,cidade,estado,telefone,email)
VALUES(?,?,?,?,?,?,?);";
    String sqlPessoaJuridica = "INSERT INTO
PessoaJuridica(Pessoa_idPessoa,cnpj) VALUES(?,?);";
    Connection con = null;
    PreparedStatement stmt = null;
    PreparedStatement stmtPj = null;
}

```

```

try{
    con = connector.getConnection();
    con.setAutoCommit(false);

    SequenceManager sequenceManager = new SequenceManager();
    int idNovaPessoa = sequenceManager.getValue("orderPessoa");

    stmt = con.prepareStatement(sqlPessoa);
    stmt.setInt(1, idNovaPessoa);
    stmt.setString(2, pessoa.getNome());
    stmt.setString(3, pessoa.getLogradouro());
    stmt.setString(4, pessoa.getCidade());
    stmt.setString(5, pessoa.getEstado());
    stmt.setString(6, pessoa.getTelefone());
    stmt.setString(7, pessoa.getEmail());
    stmt.executeUpdate();

    pessoa.setId(idNovaPessoa);

    stmtPj = con.prepareStatement(sqlPessoaJuridica);
    stmtPj.setInt(1, idNovaPessoa);
    stmtPj.setString(2, pessoa.getCnpj());
    stmtPj.executeUpdate();

    con.commit();
}catch(Exception e){
    System.err.println("Adj: " + e);
}
finally{
    connector.close(stmtPj);
    connector.close(stmt);
    connector.close(con);
}
}

public void alterar(PessoaJuridica pessoa) throws SQLException, Exception{
    String sqlPessoa = "UPDATE PESSOA SET nome=?,logradouro
=?,cidade=?,estado=?,telefone=?,email=? WHERE idPessoa=?";
    String sqlPessoaJuridica = "UPDATE PESSOA JURIDICA SET cnpj=? WHERE
Pessoa_idPessoa=?";
    Connection con = null;
    PreparedStatement stmt = null;
    PreparedStatement stmtPj = null;
    try{
        con = connector.getConnection();
        stmt = con.prepareStatement(sqlPessoa);
        stmtPj = con.prepareStatement(sqlPessoaJuridica);
        stmt.setString(1, pessoa.getNome());
        stmt.setString(2, pessoa.getLogradouro());
        stmt.setString(3, pessoa.getCidade());
        stmt.setString(4, pessoa.getEstado());
        stmt.setString(5, pessoa.getTelefone());
        stmt.setString(6, pessoa.getEmail());
        stmt.setInt(7, pessoa.getId());
        stmt.executeUpdate();
    }
}

```

```

        stmtPj.setString(1, pessoa.getCnpj());
        stmtPj.setInt(2, pessoa.getId());
        stmtPj.executeUpdate();
    }catch(Exception e){
        System.err.println("Alj: " + e);
    }finally{
        connector.close(stmtPj);
        connector.close(stmt);
        connector.close(con);
    }
}

public void excluir(int chave) throws SQLException, Exception{
    String sqlPessoa = "DELETE FROM PESSOA WHERE idPessoa=?";
    String sqlPessoaJuridica = "DELETE FROM PESSOA JURIDICA WHERE
Pessoa_idPessoa=?";
    Connection con = null;
    PreparedStatement stmt = null;
    PreparedStatement stmtPj = null;
    try{
        con = connector.getConnection();
        stmtPj = con.prepareStatement(sqlPessoaJuridica);
        stmt = con.prepareStatement(sqlPessoa);
        stmtPj.setInt(1, chave);
        stmtPj.executeUpdate();
        stmt.setInt(1, chave);
        stmt.executeUpdate();
    }catch(Exception e){
        System.err.println("Exj: " + e);
    }finally{
        connector.close(stmt);
        connector.close(stmtPj);
        connector.close(con);
    }
}
}
}

```

Figura 09: PessoaJuridicaDAO

```

package cadastrobd;

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import java.sql.SQLException;

public class CadastroBDTeste {
    private final PessoaFisicaDAO pessoafisicaDAO;
    private final PessoaJuridicaDAO pessoajuridicaDAO;

    public CadastroBDTeste() {
        pessoafisicaDAO = new PessoaFisicaDAO();
        pessoajuridicaDAO = new PessoaJuridicaDAO();
    }

    private void run(){

```

```

        PessoaFisica pf = new PessoaFisica(null, "Fred", "There", "Loob", "MG",
"333355556", "FredT@gmail.com", "35353535961");
        try{
            pessoafisicaDAO.incluir(pf);
            System.out.println("=====");
            System.out.println("Pessoa Fisica id: " + pf.getId() + " adicionada.");
            System.out.println("=====");
            pf.exibir();
            System.out.println("=====");
            pf.setCidade("Uberlandia");
            pf.setLogradouro("Rua Veneza");
            pessoafisicaDAO.alterar(pf);
            System.out.println("Pessoa Fisica id: " + pf.getId() + " dados alterados.");
            System.out.println("=====");
            pf.exibir();
            System.out.println("=====");
            System.out.println("Exibindo todas Pessoas Fisicas.");
            System.out.println("=====");
            pessoafisicaDAO.getPessoas().forEach(pessoaF -> pessoaF.exibir());
            System.out.println("=====");
            pessoafisicaDAO.excluir(pf.getId());
            System.out.println("Pessoa Fisica id: " + pf.getId() + " excluida.");
        }catch(Exception e){
            System.err.println("F: " + e);
        }
        PessoaJuridica pj = new PessoaJuridica(null, "Jake", "Priet", "Itabira", "MG",
"38365542", "JakeHern@gmail.com", "36235965821256");
        try{
            pessoajuridicaDAO.incluir(pj);
            System.out.println("=====");
            System.out.println("Pessoa Juridica id: " + pj.getId() + " adicionada.");
            System.out.println("=====");
            pj.exibir();
            System.out.println("=====");
            pj.setCidade("Ouro Preto");
            pj.setLogradouro("Rua Direita");
            pessoajuridicaDAO.alterar(pj);
            System.out.println("Pessoa Juridica id: " + pj.getId() + " Dados alterados.");
            System.out.println("=====");
            pj.exibir();
            System.out.println("=====");
            pessoajuridicaDAO.getPessoas().forEach(pessoaJ -> pessoaJ.exibir());
            System.out.println("=====");
            pessoajuridicaDAO.excluir(pj.getId());
            System.out.println("Pessoa Juridica id: " + pj.getId() + " excluida.");
        }catch(Exception e){
            System.err.println("J: " + e);
        }
    }

    public static void main(String[] args) throws SQLException, Exception{
        new CadastroBDTeste().run();
    }
}

```

Figura 10: CadastroBDTeste

Análise e Conclusão

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

Middleware é um software que atua como um intermediário entre diferentes aplicações ou sistemas. Ele facilita a comunicação e o intercâmbio de dados entre esses sistemas, sem a necessidade de modificá-los diretamente.

O middleware é muito importante para otimizar processos, tempo e recursos de ambientes organizacionais, como empresas de tecnologia que realizam os mais diversos tipos de operações virtuais e em sistemas próprios.

O desenvolvimento nativo em nuvem, por exemplo, é totalmente complicado para ser desenvolvido sem o uso desses facilitadores, já que demanda a reunião de linguagens de programação, ferramentas e metodologias diferentes, todos esses aspectos, sem o auxílio de um intermediário, afetam o rendimento dos profissionais, que precisam se esforçar para solucionar problemas complexos em pouco tempo.

Assim, a criação de infraestruturas robustas e diversificadas permite a integração e comunicação entre os diferentes protocolos, ferramentas, fontes de dados e plataformas presentes em uma empresa ou sistema.

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

a principal diferença entre o Statement e o PreparedStatement é que o PreparedStatement é “preparado” no banco de dados, tornando-o mais rápido. Ou seja, se você fizer diversas consultas parecidas, onde só mudam alguns valores, ele executa mais rapidamente do que se você fizer várias consultas usando o Statement.

Isso acontece porque quando uma consulta chega no banco de dados para ser executada, ela tem que passar por diversos passos de preparação até ser realmente processada. Ao usar o Statement, esses passos são feitos a cada consulta. Já no PreparedStatement, esses passos são feitos apenas uma vez.

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

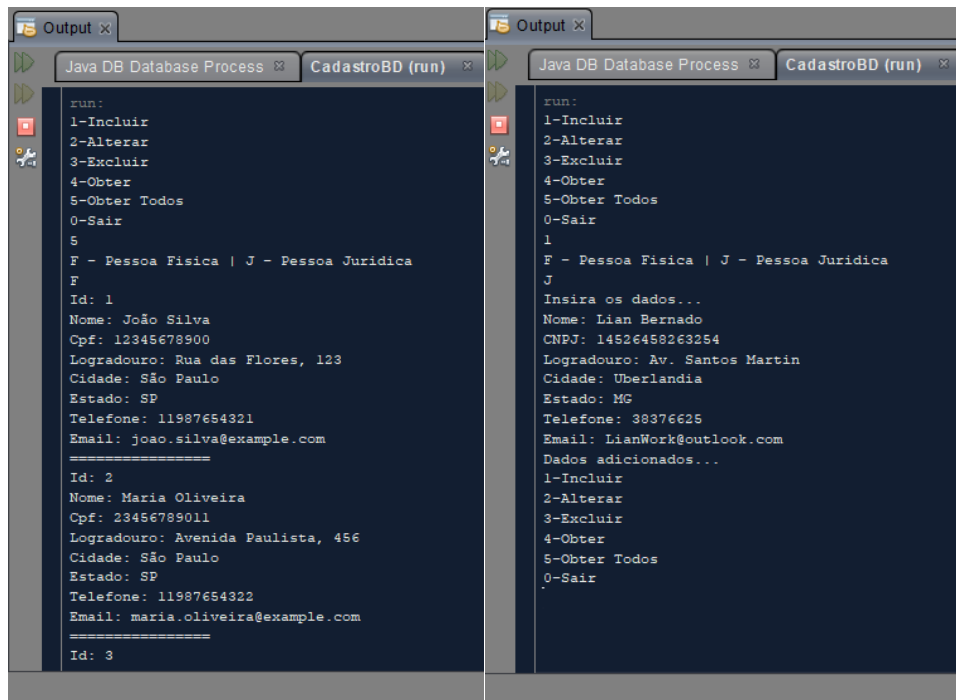
DAO é a camada do sistema (pacotes, classes e métodos) que abstrai todo o acesso ao banco de dados separadamente da lógica de negócio da aplicação. É no DAO que implementamos os métodos do CRUD (Create – Read – Update – Delete). Bem como é por meio do DAO que as operações no banco de dados são realizadas.

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

Cada hierárquica mapeada reflete uma tabela, relacionadas através do mecanismo de especialização padrão do banco de dados relacional (utilização de chaves estrangeiras). Segunda esta modalidade de mapeamento, tenta-se ao máximo manter a normalização de dados, de forma que a estrutura final das tabelas fica bastante parecida com a hierarquia das classes representada na UML. Esta é a técnica que mais naturalmente mapeia objetos para banco de dados relacionais.

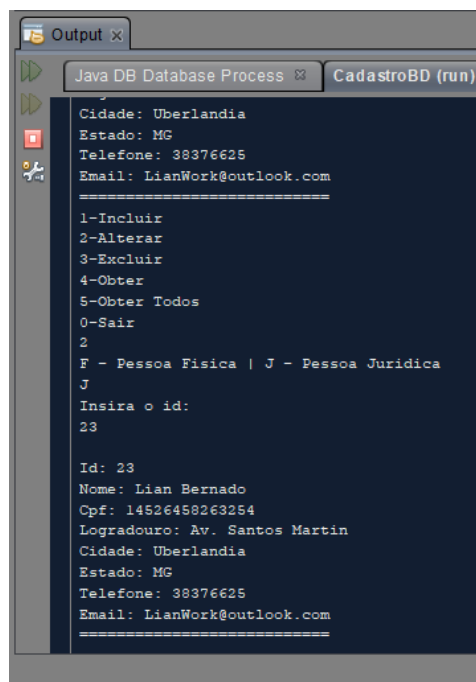
2º Procedimento | Alimentando a Base

Resultados



```
run:
1-Incluir
2-Alterar
3-Excluir
4-Obter
5-Obter Todos
0-Sair
5
F - Pessoa Fisica | J - Pessoa Juridica
F
Id: 1
Nome: João Silva
Cpf: 12345678900
Logradouro: Rua das Flores, 123
Cidade: São Paulo
Estado: SP
Telefone: 11987654321
Email: joao.silva@example.com
=====
Id: 2
Nome: Maria Oliveira
Cpf: 23456789011
Logradouro: Avenida Paulista, 456
Cidade: São Paulo
Estado: SP
Telefone: 11987654322
Email: maria.oliveira@example.com
=====
Id: 3

run:
1-Incluir
2-Alterar
3-Excluir
4-Obter
5-Obter Todos
0-Sair
1
F - Pessoa Fisica | J - Pessoa Juridica
J
Insira os dados...
Nome: Lian Bernado
CNPJ: 14526458263254
Logradouro: Av. Santos Martin
Cidade: Uberlandia
Estado: MG
Telefone: 38376625
Email: LianWork@outlook.com
Dados adicionados...
1-Incluir
2-Alterar
3-Excluir
4-Obter
5-Obter Todos
0-Sair
.
```



```
Cidade: Uberlandia
Estado: MG
Telefone: 38376625
Email: LianWork@outlook.com
=====
1-Incluir
2-Alterar
3-Excluir
4-Obter
5-Obter Todos
0-Sair
2
F - Pessoa Fisica | J - Pessoa Juridica
J
Insira o id:
23

Id: 23
Nome: Lian Bernado
Cpf: 14526458263254
Logradouro: Av. Santos Martin
Cidade: Uberlandia
Estado: MG
Telefone: 38376625
Email: LianWork@outlook.com
=====
```

Figura 11: CadastroBD

Códigos Solicitados.

```
package cadastrbd;

import cadastrbd.model.PessoaFisica;
import cadastrbd.model.PessoaFisicaDAO;
import cadastrbd.model.PessoaJuridica;
import cadastrbd.model.PessoaJuridicaDAO;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class CadastroBD {
    private static final PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
    private static final PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
    private static final BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));

    public static void main(String[] args) throws Exception{
        CadastroBD cadastroBD = new CadastroBD();
        while(true){
            System.out.println("1-Incluir\n2-Alterar\n3-Excluir\n4-Obter\n5-Obter Todos\n0-Sair");
            int opcao = Integer.parseInt(entrada.readLine());
            if(opcao == 0)
                break;
            switch(opcao){
                case 1:
                    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                    String tipoPessoal = entrada.readLine();
                    if(tipoPessoal.equals("F")){
                        System.out.println("Insira os dados...");
                        try{
                            System.out.print("Nome: ");
                            String nome = entrada.readLine();

                            System.out.print("CPF: ");
                            String cpf = entrada.readLine();

                            System.out.print("Logradouro: ");
                            String logradouro = entrada.readLine();

                            System.out.print("Cidade: ");
                            String cidade = entrada.readLine();

                            System.out.print("Estado: ");
                            String estado = entrada.readLine();
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    } else if(tipoPessoal.equals("J")){
                        System.out.println("Insira os dados...");
                        try{
                            System.out.print("Nome: ");
                            String nome = entrada.readLine();

                            System.out.print("CPF: ");
                            String cpf = entrada.readLine();

                            System.out.print("Logradouro: ");
                            String logradouro = entrada.readLine();

                            System.out.print("Cidade: ");
                            String cidade = entrada.readLine();

                            System.out.print("Estado: ");
                            String estado = entrada.readLine();
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                case 2:
                    System.out.println("Insira os dados...");
                    try{
                        System.out.print("Nome: ");
                        String nome = entrada.readLine();

                        System.out.print("CPF: ");
                        String cpf = entrada.readLine();

                        System.out.print("Logradouro: ");
                        String logradouro = entrada.readLine();

                        System.out.print("Cidade: ");
                        String cidade = entrada.readLine();

                        System.out.print("Estado: ");
                        String estado = entrada.readLine();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                case 3:
                    System.out.println("Insira os dados...");
                    try{
                        System.out.print("Nome: ");
                        String nome = entrada.readLine();

                        System.out.print("CPF: ");
                        String cpf = entrada.readLine();

                        System.out.print("Logradouro: ");
                        String logradouro = entrada.readLine();

                        System.out.print("Cidade: ");
                        String cidade = entrada.readLine();

                        System.out.print("Estado: ");
                        String estado = entrada.readLine();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                case 4:
                    System.out.println("Insira os dados...");
                    try{
                        System.out.print("Nome: ");
                        String nome = entrada.readLine();

                        System.out.print("CPF: ");
                        String cpf = entrada.readLine();

                        System.out.print("Logradouro: ");
                        String logradouro = entrada.readLine();

                        System.out.print("Cidade: ");
                        String cidade = entrada.readLine();

                        System.out.print("Estado: ");
                        String estado = entrada.readLine();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                case 5:
                    System.out.println("Insira os dados...");
                    try{
                        System.out.print("Nome: ");
                        String nome = entrada.readLine();

                        System.out.print("CPF: ");
                        String cpf = entrada.readLine();

                        System.out.print("Logradouro: ");
                        String logradouro = entrada.readLine();

                        System.out.print("Cidade: ");
                        String cidade = entrada.readLine();

                        System.out.print("Estado: ");
                        String estado = entrada.readLine();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            }
        }
    }
}
```

```

        System.out.print("Telefone: ");
        String telefone = entrada.readLine();

        System.out.print("Email: ");
        String email = entrada.readLine();

        PessoaFisica novaPessoaFisica = new PessoaFisica(null, nome,
logradouro, cidade, estado, telefone, email, cpf);
        pessoaFisicaDAO.incluir(novaPessoaFisica);
    }catch(Exception e){System.err.println(e);}
}else if(tipoPessoal.equals("J")){
    System.out.println("Insira os dados...");
    try{
        System.out.print("Nome: ");
        String nome = entrada.readLine();

        System.out.print("CNPJ: ");
        String cnpj = entrada.readLine();

        System.out.print("Logradouro: ");
        String logradouro = entrada.readLine();

        System.out.print("Cidade: ");
        String cidade = entrada.readLine();

        System.out.print("Estado: ");
        String estado = entrada.readLine();

        System.out.print("Telefone: ");
        String telefone = entrada.readLine();

        System.out.print("Email: ");
        String email = entrada.readLine();

        PessoaJuridica novaPessoaJuridica = new PessoaJuridica(null,
nome, logradouro, cidade, estado, telefone, email, cnpj);
        pessoaJuridicaDAO.incluir(novaPessoaJuridica);
    }catch(Exception e){System.err.println(e);}
}
    System.out.println("Dados adicionados...");
    break;

case 2:
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoPessoaA = entrada.readLine();
    if(tipoPessoaA.equals("F")){
        System.out.print("Insira o id: ");
        int pessoald = Integer.parseInt(entrada.readLine());

```

```

try{
    pessoaFisicaDAO.getPessoa(pessoald).exibir();
    System.out.println("=====");

    System.out.println("Insira os dados...");

    System.out.print("Nome: ");
    String nome = entrada.readLine();

    System.out.print("CPF: ");
    String cpf = entrada.readLine();

    System.out.print("Logradouro: ");
    String logradouro = entrada.readLine();

    System.out.print("Cidade: ");
    String cidade = entrada.readLine();

    System.out.print("Estado: ");
    String estado = entrada.readLine();

    System.out.print("Telefone: ");
    String telefone = entrada.readLine();

    System.out.print("Email: ");
    String email = entrada.readLine();

    PessoaFisica novaPessoaFisica = new PessoaFisica(pessoald,
nome, logradouro, cidade, estado, telefone, email, cpf);
    pessoaFisicaDAO.alterar(novaPessoaFisica);
}catch(Exception e){System.err.println(e);}
}
else if(tipoPessoaA.equals("J")){
    System.out.println("Insira o id: ");
    int pessoald = Integer.parseInt(entrada.readLine());
    try{
        pessoaJuridicaDAO.getPessoa(pessoald).exibir();
        System.out.println("=====");

        System.out.println("Insira os dados...");

        System.out.print("Nome: ");
        String nome = entrada.readLine();

        System.out.print("CNPJ: ");
        String cnpj = entrada.readLine();

        System.out.print("Logradouro: ");

```

```

        String logradouro = entrada.readLine();

        System.out.print("Cidade: ");
        String cidade = entrada.readLine();

        System.out.print("Estado: ");
        String estado = entrada.readLine();

        System.out.print("Telefone: ");
        String telefone = entrada.readLine();

        System.out.print("Email: ");
        String email = entrada.readLine();

        PessoaJuridica novaPessoaJuridica = new
PessoaJuridica(pessoald, nome, logradouro, cidade, estado, telefone, email, cnpj);
        pessoaJuridicaDAO.alterar(novaPessoaJuridica);
    }catch(Exception e){System.err.println(e);}
    }
    System.out.println("Dados alterados...");
    break;

case 3:
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoPessoaEX = entrada.readLine();
    if(tipoPessoaEX.equals("F")){
        System.out.println("Insira o id: ");
        int pessoald = Integer.parseInt(entrada.readLine());
        try{
            pessoaFisicaDAO.excluir(pessoald);
        }catch(Exception e){System.err.println(e);}
    }else if(tipoPessoaEX.equals("J")){
        System.out.println("Insira o id: ");
        int pessoald = Integer.parseInt(entrada.readLine());
        try{
            pessoaJuridicaDAO.excluir(pessoald);
        }catch(Exception e){System.err.println(e);}
    }
    System.out.println("Pessoa removida...");
    break;

case 4:
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoPessoaO = entrada.readLine();
    if(tipoPessoaO.equals("F")){
        System.out.println("Insira o id: ");
        int pessoald = Integer.parseInt(entrada.readLine());
        try{

```

```

        pessoaFisicaDAO.getPessoa(pessoald).exibir();
    }catch(Exception e){System.err.println(e);}
}else if(tipoPessoaO.equals("J")){
    System.out.println("Insira o id: ");
    int pessoald = Integer.parseInt(entrada.readLine());
    try{
        pessoaJuridicaDAO.getPessoa(pessoald).exibir();
    }catch(Exception e){System.err.println(e);}
}
break;
case 5:
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoPessoaOT = entrada.readLine();
    if(tipoPessoaOT.equals("F")){
        try{
            pessoaFisicaDAO.getPessoas().forEach(pessoa
pessoa.exibir());
            }catch(Exception e){System.err.println(e);}
        }else if(tipoPessoaOT.equals("J")){
            try{
                pessoaJuridicaDAO.getPessoas().forEach(pessoa
pessoa.exibir());
            }catch(Exception e){System.err.println(e);}
        }
        break;

    case 0:
        System.out.println("Finalizando a execução...");
        entrada.close();
        return;

    default:
        System.out.println("Escolha uma opção entre 0 a 5.");
        break;
}
}
}
}
}
}

```

Figura 12: CadastroBD

Análise e Conclusão

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

A persistência em arquivo é uma forma simples de persistir dados em sistemas de computação. Os arquivos são amplamente utilizados para armazenar configurações, logs e outros tipos de informações que não requerem um banco de dados completo, eles são legíveis por humanos, enquanto os arquivos binários são mais eficientes em termos de espaço e desempenho.

A persistência em bancos de dados utiliza sistemas de gerenciamento de banco de dados para armazenar, recuperar e gerenciar dados de maneira estruturada, com uso de tabelas, índices, suporta operações complexas e transações. Também oferece recursos avançados como atomicidade, consistência, isolamento, durabilidade, além de segurança, backup e otimizações para acessos simultâneos.

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

Em Java, a expressão lambda é uma nova característica introduzida na versão 8 da linguagem. Ela permite criar funções anônimas com sintaxe mais concisa e expressiva.

Uma expressão lambda em Java é composta por três partes principais:

- Argumentos: Uma lista de argumentos separados por vírgulas, que especifica os parâmetros que a expressão lambda recebe. Esses parâmetros podem ter tipos explícitos ou podem ser inferidos pelo compilador.
- Setas: O operador "->" separa os argumentos da expressão lambda do corpo da função.
- Corpo: O corpo da função contém a expressão que define a ação a ser realizada pela função. Esse corpo pode ser uma única expressão ou um bloco de código encapsulado em chaves "{}".

A expressão lambda pode ser atribuída a uma variável do tipo de uma interface funcional, que define um único método abstrato. Essa interface pode ser definida pelo próprio programador ou pode ser uma das interfaces funcionais pré-definidas em Java, como a interface "java.util.function.Consumer" ou "java.util.function.Predicate".

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

Em Java, métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static devido à forma como a linguagem Java organiza a memória e a execução do código.

O método main é static porque ele é chamado pela JVM sem a necessidade de criar uma instância da classe. Como a JVM não cria um objeto da classe para executar o main, o método main deve ser static para que possa ser chamado sem precisar de uma instância.