RPG0016 - BackEnd sem banco não tem

Ricardo Alves dos Santos Junior - 202208681158

Universidade Estácio de Sá

Nível 3: Back-end Sem Banco Não Tem – 2023.3 – Terceiro Semestre

Link do Projeto Completo: https://github.com/RicardoASJunior/CadastroBD

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 – Criando o Banco de Dados:

Criação da Classe Pessoa:

Pessoa.java

src > cadastrobd > model > Pessoa.java > Pessoa > getEstado()

```java
package cadastrobd.model;

public class Pessoa {
    // Campos da classe
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    // Construtor padrão
    public Pessoa() {
        this.id = 0;
        this.nome = "";
        this.logradouro = "";
        this.cidade = "";
        this.estado = "";
        this.telefone = "";
        this.email = "";
    }

    // Construtor completo
    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;
    }



    // Getters e setters
    public int getId() {
        return this.id;
    }

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

    public String getNome() {
        return this.nome;
    }
```

```java
    public String getNome() {
        return this.nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return this.logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return this.cidade;
    }

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

    public String getEstado() {
        return this.estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return this.telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return this.email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```java
    // Método exibir
    public void exibir() {
        System.out.println(x:"============================");
        System.out.println("ID: " + this.id);
        System.out.println("Nome: " + this.nome);
        System.out.println("Logradouro: " + this.logradouro);
        System.out.println("Cidade: " + this.cidade);
        System.out.println("Estado: " + this.estado);
        System.out.println("Telefone: " + this.telefone);
        System.out.println("Email: " + this.email);
    }
}
```

Criação da classe PessoaFisica:

PessoaFisica.java

src > cadastrobd > model > PessoaFisica.java > PessoaFisica > exibir()

```java
package cadastrobd.model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    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;
    }

    public String getCpf() {
        return cpf;
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + this.cpf );
        System.out.println(x:"============================");
    }

    public void remove(PessoaFisica pessoaFisica) {
    }
}

```

Criação da classe PessoaJuridica:

PessoaJuridica.java

src > cadastrobd > model > PessoaJuridica.java > {} cadastrobd.model

```java
package cadastrobd.model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    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;
    }

    public String getCnpj() {
        return cnpj;
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + this.cnpj);
        System.out.println(x:"============================");

    }
}

```

Criação da classe ConectoBD:

ConectorBD.java

src > cadastrobd > model > ConectorBD.java > ...

```java
package cadastrobd.model;
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 {
 // Constantes para armazenar os dados de conexão com o banco de dados
 // Você pode alterar esses valores de acordo com a sua configuração

 private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=Loja;user=loja;password=loja;encrypt=true;trustServerCertificate=true;";


 public static Connection getConnection() throws SQLException {
     try {
         // Carregar o driver JDBC do SQL Server
         Class.forName(className:"com.microsoft.sqlserver.jdbc.SQLServerDriver");

     } catch (ClassNotFoundException e) {
         // Tratar a exceção se o driver não for encontrado
         e.printStackTrace();
     }
     return DriverManager.getConnection(URL);
 }
 // Criar o pacote cadastrobd.model.util package cadastrobd.model.util;
 //Método getPrepared, para retornar um objeto do tipo PreparedStatement a partir de um SQL fornecido com parâmetro
 public static PreparedStatement getPrepared(String sql) throws SQLException {
     // Obter uma conexão com o banco de dados
     Connection con = getConnection();
     // Criar e retornar um objeto do tipo PreparedStatement usando o SQL fornecido
     return con.prepareStatement(sql);
 }

 // Método getSelect, para retornar o ResultSet relacionado a uma consulta
 public static ResultSet getSelect(String sql) throws SQLException {
     // Obter um objeto do tipo PreparedStatement usando o método getPrepared
     PreparedStatement ps = getPrepared(sql);
     // Executar a consulta e retornar o ResultSet resultante
     return ps.executeQuery();
 }

 // Método close sobrecarregado para Statement, para garantir o fechamento do objeto
 public static void close(Statement st) {
     // Verificar se o objeto não é nulo
```

```java
    // Método close sobrecarregado para Statement, para garantir o fechamento do objeto
    public static void close(Statement st) {
        // Verificar se o objeto não é nulo
        if (st != null) {
            try {
                // Fechar o objeto
                st.close();
            } catch (SQLException e) {
                // Imprimir a mensagem de erro
                e.printStackTrace();
            }
        }
    }

    // Método close sobrecarregado para ResultSet, para garantir o fechamento do objeto
    public static void close(ResultSet rs) {
        // Verificar se o objeto não é nulo
        if (rs != null) {
            try {
                // Fechar o objeto
                rs.close();
            } catch (SQLException e) {
                // Imprimir a mensagem de erro
                e.printStackTrace();
            }
        }
    }

    // Método close sobrecarregado para Connection, para garantir o fechamento do objeto
    public static void close(Connection con) {
        // Verificar se o objeto não é nulo
        if (con != null) {
            try {
                // Fechar o objeto
                con.close();
            } catch (SQLException e) {
                // Imprimir a mensagem de erro
                e.printStackTrace();
            }
        }
    }
    // Classe SequenceManager
    public class SequenceManager {
        // Método getValue, recebendo o nome da sequência como parâmetro e retornando o próximo valor
        public static int getValue(String sequence) throws SQLException {
            // Inicializar uma variável para armazenar o valor da sequência
            int value = 0;
            // Criar uma string com o SQL para obter o valor atual da sequência
            // Criar uma string com o SQL para obter o valor atual da sequência
            String sql = "SELECT NEXT VALUE FOR " + sequence ;
            // Obter o ResultSet relacionado à consulta usando o método getSelect da classe ConectorBD
            ResultSet rs = ConectorBD.getSelect(sql);
            // Verificar se o ResultSet tem algum resultado
            if (rs.next()) {
                // Obter o valor da sequência a partir da primeira coluna do ResultSet
                value = rs.getInt(columnIndex:1);
            }
            // Fechar o ResultSet usando o método close da classe ConectorBD
            ConectorBD.close(rs);
            // Retornar o valor da sequência
            return value;
        }
    }
}
```

Criação da classe PessoaFisicaDAO:

PessoaFisicaDAO.java

src > cadastrobd > model > PessoaFisicaDAO.java > PessoaFisicaDAO > getPessoa(int)

```java
package cadastrobd.model;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

//import cadastrobd.model.ConectorBD.SequenceManager;

public class PessoaFisicaDAO {

    // Método para obter uma pessoa física a partir do seu id
    public PessoaFisica getPessoa(int id) throws SQLException {
        PessoaFisica pf = null;
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaFisica pf ON p.id = pf.pessoa_id WHERE p.id = ?");
            ps.setInt(parameterIndex:1, id);
            rs = ps.executeQuery();
            if (rs.next()) {
                pf = new PessoaFisica();
                pf.setId(rs.getInt(columnLabel:"id"));
                pf.setNome(rs.getString(columnLabel:"nome"));
                pf.setLogradouro(rs.getString(columnLabel:"logradouro"));
                pf.setCidade(rs.getString(columnLabel:"cidade"));
                pf.setEstado(rs.getString(columnLabel:"estado"));
                pf.setTelefone(rs.getString(columnLabel:"telefone"));
                pf.setEmail(rs.getString(columnLabel:"email"));
                pf.setCpf(rs.getString(columnLabel:"cpf"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
        if (pf != null){
            pf.exibir();
        }else{
            System.out.println(x:"Este ID não existe!");
        }

        return pf;
    }

```

```java
    // Método para obter todas as pessoas físicas do banco de dados
    public List<PessoaFisica> getPessoas() throws SQLException {
        List<PessoaFisica> pessoas = new ArrayList<>();
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaFisica pf ON p.id = pf.pessoa_id");
            rs = ps.executeQuery();
            while (rs.next()) {
                PessoaFisica pf = new PessoaFisica();
                pf.setId(rs.getInt(columnLabel:"id"));
                pf.setNome(rs.getString(columnLabel:"nome"));
                pf.setLogradouro(rs.getString(columnLabel:"logradouro"));
                pf.setCidade(rs.getString(columnLabel:"cidade"));
                pf.setEstado(rs.getString(columnLabel:"estado"));
                pf.setTelefone(rs.getString(columnLabel:"telefone"));
                pf.setEmail(rs.getString(columnLabel:"email"));
                pf.setCpf(rs.getString(columnLabel:"cpf"));
                pessoas.add(pf);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
        return pessoas;
    }

    // Método para incluir uma pessoa física nas tabelas Pessoa e PessoaFisica
    public void incluir(PessoaFisica pf) throws SQLException {
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        try {
            // Inserir na tabela Pessoa
            ps = conn.prepareStatement(sql:"INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
            ps.setString(parameterIndex:1, pf.getNome());
            ps.setString(parameterIndex:2, pf.getLogradouro());
            ps.setString(parameterIndex:3, pf.getCidade());
            ps.setString(parameterIndex:4, pf.getEstado());
            ps.setString(parameterIndex:5, pf.getTelefone());
            ps.setString(parameterIndex:6, pf.getEmail());

            int rows = ps.executeUpdate();

            // Verificar se a inserção foi bem sucedida
            if (rows > 0) {
```

```java
            if (rows > 0) {
                // Obter o ResultSet com o id gerado pelo banco
                ResultSet rs = ps.getGeneratedKeys();

                // Verificar se o ResultSet tem algum resultado
                if (rs.next()) {
                    // Obter o id da pessoa a partir da primeira coluna do ResultSet
                    int id = rs.getInt(columnIndex:1);

                    // Usar o mesmo objeto PreparedStatement para inserir os dados na tabela PessoaFisica
                    ps = conn.prepareStatement(sql:"INSERT INTO PessoaFisica (cpf, pessoa_id) VALUES (?, ?)");
                    ps.setString(parameterIndex:1, pf.getCpf());
                    ps.setInt(parameterIndex:2, id);
                    rows = ps.executeUpdate();

                    if (rows > 0) {
                        System.out.println(x:"Pessoa física inserida com sucesso!");
                    } else {
                        System.out.println(x:"Falha ao inserir a pessoa física!");
                    }
                } else {
                    System.out.println(x:"Falha ao obter o id da pessoa!");
                }
            } else {
                System.out.println(x:"Falha ao inserir a pessoa!");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
    }

    // Método para alterar os dados de uma pessoa física
    public void alterar(PessoaFisica pf) throws SQLException {
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        try {
            // Alterar na tabela Pessoa
            ps = conn.prepareStatement(sql:"UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?");
            ps.setString(parameterIndex:1, pf.getNome());
            ps.setString(parameterIndex:2, pf.getLogradouro());
            ps.setString(parameterIndex:3, pf.getCidade());
            ps.setString(parameterIndex:4, pf.getEstado());
            ps.setString(parameterIndex:5, pf.getTelefone());
            ps.setString(parameterIndex:6, pf.getEmail());
            ps.setInt(parameterIndex:7, pf.getId());
```

```
            ps.setInt(parameterIndex:7, pf.getId());
            ps.executeUpdate();

            // Alterar na tabela PessoaFisica
            ps = conn.prepareStatement(sql:"UPDATE PessoaFisica SET cpf = ? WHERE pessoa_id = ?");
            ps.setString(parameterIndex:1, pf.getCpf());
            ps.setInt(parameterIndex:2, pf.getId());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
    }

    // Método para remover uma pessoa física das tabelas Pessoa e PessoaFisica
    public void excluir(int id) throws SQLException {
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        try {
            // Excluir da tabela PessoaFisica
            ps = conn.prepareStatement(sql:"DELETE FROM PessoaFisica WHERE pessoa_id = ?");
            ps.setInt(parameterIndex:1, id);
            ps.executeUpdate();

            // Excluir da tabela Pessoa
            ps = conn.prepareStatement(sql:"DELETE FROM Pessoa WHERE id = ?");
            ps.setInt(parameterIndex:1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
    }
}
```

Criação da Classe PessoaJuridicaDAO:

PessoaJuridicaDAO.java

src > cadastrobd > model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > getPessoa(int)

```java
package cadastrobd.model;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;


public class PessoaJuridicaDAO {

    // Método para obter uma pessoa física a partir do seu id
    public PessoaJuridica getPessoa(int id) throws SQLException {
        PessoaJuridica pj = null;
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.id = pj.pessoa_id WHERE p.id = ?");
            ps.setInt(parameterIndex:1, id);
            rs = ps.executeQuery();
            if (rs.next()) {
                pj = new PessoaJuridica();
                pj.setId(rs.getInt(columnLabel:"id"));
                pj.setNome(rs.getString(columnLabel:"nome"));
                pj.setLogradouro(rs.getString(columnLabel:"logradouro"));
                pj.setCidade(rs.getString(columnLabel:"cidade"));
                pj.setEstado(rs.getString(columnLabel:"estado"));
                pj.setTelefone(rs.getString(columnLabel:"telefone"));
                pj.setEmail(rs.getString(columnLabel:"email"));
                pj.setCnpj(rs.getString(columnLabel:"cnpj"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
        if (pj != null){
            pj.exibir();
        }else{
            System.out.println(x:"Este ID não existe!");
        }
        return pj;
    }

    // Método para obter todas as pessoas físicas do banco de dados
    public List<PessoaJuridica> getPessoas() throws SQLException {
```

```java
    // Método para obter todas as pessoas físicas do banco de dados
    public List<PessoaJuridica> getPessoas() throws SQLException {
        List<PessoaJuridica> pessoas = new ArrayList<>();
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.id = pj.pessoa_id");
            rs = ps.executeQuery();
            while (rs.next()) {
                PessoaJuridica pj = new PessoaJuridica();
                pj.setId(rs.getInt(columnLabel:"id"));
                pj.setNome(rs.getString(columnLabel:"nome"));
                pj.setLogradouro(rs.getString(columnLabel:"logradouro"));
                pj.setCidade(rs.getString(columnLabel:"cidade"));
                pj.setEstado(rs.getString(columnLabel:"estado"));
                pj.setTelefone(rs.getString(columnLabel:"telefone"));
                pj.setEmail(rs.getString(columnLabel:"email"));
                pj.setCnpj(rs.getString(columnLabel:"cnpj"));
                pessoas.add(pj);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
        return pessoas;
    }

    // Método para incluir uma pessoa física nas tabelas Pessoa e PessoaJuridica
    public void incluir(PessoaJuridica pj) throws SQLException {
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        try {
            // Inserir na tabela Pessoa
            ps = conn.prepareStatement(sql:"INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
            ps.setString(parameterIndex:1, pj.getNome());
            ps.setString(parameterIndex:2, pj.getLogradouro());
            ps.setString(parameterIndex:3, pj.getCidade());
            ps.setString(parameterIndex:4, pj.getEstado());
            ps.setString(parameterIndex:5, pj.getTelefone());
            ps.setString(parameterIndex:6, pj.getEmail());

            int rows = ps.executeUpdate();

            // Verificar se a inserção foi bem sucedida
            if (rows > 0) {
```

```
            // Verificar se a inserção foi bem sucedida
            if (rows > 0) {
                // Obter o ResultSet com o id gerado pelo banco
                ResultSet rs = ps.getGeneratedKeys();

                // Verificar se o ResultSet tem algum resultado
                if (rs.next()) {
                    // Obter o id da pessoa a partir da primeira coluna do ResultSet
                    int id = rs.getInt(columnIndex:1);

                    // Usar o mesmo objeto PreparedStatement para inserir os dados na tabela PessoaJurídica
                    ps = conn.prepareStatement(sql:"INSERT INTO PessoaJuridica (cnpj, pessoa_id) VALUES (?, ?)");
                    ps.setString(parameterIndex:1, pj.getCnpj());
                    ps.setInt(parameterIndex:2, id);
                    rows = ps.executeUpdate();

                    if (rows > 0) {
                        System.out.println(x:"Pessoa jurídica inserida com sucesso!");
                    } else {
                        System.out.println(x:"Falha ao inserir a pessoa jurídica!");
                    }
                } else {
                    System.out.println(x:"Falha ao obter o id da pessoa!");
                }
            } else {
                System.out.println(x:"Falha ao inserir a pessoa!");
            }


        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
    }

    // Método para alterar os dados de uma pessoa física
    public void alterar(PessoaJuridica pj) throws SQLException {
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        try {
            // Alterar na tabela Pessoa
            ps = conn.prepareStatement(sql:"UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?");
            ps.setString(parameterIndex:1, pj.getNome());
            ps.setString(parameterIndex:2, pj.getLogradouro());
            ps.setString(parameterIndex:3, pj.getCidade());
            ps.setString(parameterIndex:4, pj.getEstado());
```

```
            ps.setString(parameterIndex:2, pj.getLogradouro());
            ps.setString(parameterIndex:3, pj.getCidade());
            ps.setString(parameterIndex:4, pj.getEstado());
            ps.setString(parameterIndex:5, pj.getTelefone());
            ps.setString(parameterIndex:6, pj.getEmail());
            ps.setInt(parameterIndex:7, pj.getId());
            ps.executeUpdate();

            // Alterar na tabela PessoaJuridica
            ps = conn.prepareStatement(sql:"UPDATE PessoaJuridica SET cnpj = ? WHERE pessoa_id = ?");
            ps.setString(parameterIndex:1, pj.getCnpj());
            ps.setInt(parameterIndex:2, pj.getId());
            ps.executeUpdate();
            System.out.println(x:"Pessoa jurídica alterada com sucesso!");
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
    }

    // Método para remover uma pessoa física das tabelas Pessoa e PessoaJuridica
    public void excluir(int id) throws SQLException {
        Connection conn = ConectorBD.getConnection();
        PreparedStatement ps = null;
        try {
            // Excluir da tabela PessoaJuridica
            ps = conn.prepareStatement(sql:"DELETE FROM PessoaJuridica WHERE pessoa_id = ?");
            ps.setInt(parameterIndex:1, id);
            ps.executeUpdate();

            // Excluir da tabela Pessoa
            ps = conn.prepareStatement(sql:"DELETE FROM Pessoa WHERE id = ?");
            ps.setInt(parameterIndex:1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
        }
    }
}
```

Criação da classe Main:

Main.java

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

public class Main {
    Run | Debug
    public static void main(String[] args) throws SQLException {
        // Instanciar uma pessoa fisica e persis4tir no banco de dados
        PessoaFisica pf = new PessoaFisica(id:17, nome:"João", logradouro:"Rua A", cidade:"São Paulo",
        estado:"SP", telefone:"1111-1111", email:"teste@teste.com",
        cpf:"123.456.789-00");
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
        pfDAO.getPessoa(id:1);

        pfDAO.incluir(pf);

        // // // Alterar os dados da pessoa física no banco
        pf.setNome(nome:"João Pedro Paulo");
        pf.setTelefone(telefone:"2222-2222");
        pf.setCpf(cpf:"234.456.456-12");
        pfDAO.alterar(pf);
        System.out.println(x:"Pessoa física alterada com sucesso!");

        // Consultar todas as pessoas físicas do banco de dados e listar no console
        List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
        System.out.println(x:"Pessoas físicas cadastradas:");
        for (PessoaFisica p : pessoasFisicas) {
        p.exibir();
        }

        // // //Excluir a pessoa física criada anteriormente no banco
        pfDAO.excluir(pf.getId());
        System.out.println(x:"Pessoa física excluída com sucesso!");

        // Instanciar uma pessoa jurídica e persistir no banco de dados
        PessoaJuridica pj = new PessoaJuridica(id:2, nome:"Empresa X", logradouro:"Rua B", cidade:"Rio de Janeiro",
                                               estado:"RJ", telefone:"3333-3333", email:"teste@teste.com",
                                               cnpj:"12.345.6780001-90");
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
        pjDAO.getPessoa(id:3);
        pjDAO.incluir(pj);
        System.out.println(x:"Pessoa jurídica incluída com sucesso!");

        // Alterar os dados da pessoa jurídica no banco
        pj.setNome(nome:"Empresa Y");
        pjDAO.alterar(pj);
        System.out.println(x:"Pessoa jurídica alterada com sucesso!");

        // Consultar todas as pessoas jurídicas do banco e listar no console
        List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
        System.out.println(x:"Pessoas jurídicas cadastradas:");
        for (PessoaJuridica p : pessoasJuridicas) {
        p.exibir();
        }

        // Excluir a pessoa jurídica criada anteriormente no banco
        pjDAO.excluir(pj.getId());
        System.out.println(x:"Pessoa jurídica excluída com sucesso!");
    }
}
```

Resultado obtido ao executar a classe Main:

```
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\ricas\OneDrive\Documentos\Projetos VsCode\Java\CadastroBD> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '@C:\Users\ricas\AppData\Local\Temp\cp_ci94a6ul0i1l1ej7t2ehzhjmk.argfile' 'Main'
==============================
ID: 1
Nome: João Pedro
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 2222-2222
Email: teste@teste.com
CPF: 123.456.789-00
==============================
Pessoa física inserida com sucesso!
Pessoa física alterada com sucesso!
Pessoas físicas cadastradas:
==============================
ID: 1
Nome: João Pedro
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 2222-2222
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 16
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 19
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 21
Nome: Ricardo
Logradouro: kasdka
Cidade: aqui
Estado: RS
```

```
==============================
ID: 21
Nome: Ricardo
Logradouro: kasdka
Cidade: aqui
Estado: RS
Telefone: 12312312
Email: skdgj@sdg.com
CPF: 12345542
==============================
==============================
ID: 23
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 25
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 27
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 29
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
==============================
ID: 31
Nome:
```

```
==============================
ID: 31
Nome:
Logradouro: Ricard
Cidade: jaragua
Estado: sc
Telefone: asd
Email: asd
CPF: asd
==============================
==============================
ID: 32
Nome:
Logradouro: Ricardo
Cidade: jara
Estado: sa
Telefone: sa
Email: sa
CPF: 123
==============================
==============================
ID: 35
Nome: João
Logradouro: Rua A
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com
CPF: 123.456.789-00
==============================
Pessoa física excluída com sucesso!
==============================
ID: 3
Nome: Empresa ABC
Logradouro: Avenida Brasil, 456
Cidade: Fortaleza
Estado: CE
Telefone: 85912345678
Email: empresa@abc.com
CNPJ: 12345.678/0001-90
==============================
Pessoa jurídica inserida com sucesso!
Pessoa jurídica incluída com sucesso!
Pessoa jurídica alterada com sucesso!
Pessoa jurídica alterada com sucesso!
Pessoas jurídicas cadastradas:
==============================
ID: 3
Nome: Empresa ABC
Logradouro: Avenida Brasil, 456
```

```
==============================
ID: 3
Nome: Empresa ABC
Logradouro: Avenida Brasil, 456
Cidade: Fortaleza
Estado: CE
Telefone: 85912345678
Email: empresa@abc.com
CNPJ: 12345.678/0001-90
==============================
==============================
ID: 22
Nome: Empresa X
Logradouro: Rua B
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 3333-3333
Email: teste@teste.com
CNPJ: 12.345.6780001-90
==============================
==============================
ID: 24
Nome: Empresa X
Logradouro: Rua B
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 3333-3333
Email: teste@teste.com
CNPJ: 12.345.6780001-90
==============================
==============================
ID: 26
Nome:
Logradouro:
Cidade:
Estado:
Telefone:
Email:
CNPJ: 2345234
==============================
==============================
ID: 28
Nome: Empresa X
Logradouro: Rua B
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 3333-3333
Email: teste@teste.com
CNPJ: 12.345.6780001-90
==============================
==============================
```


Para executar

Ao Final da execução todas as funções funcionaram como foi descrito no inicio da atividade.

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

Estes componentes faclitam a criação de código fazendo com que o desenvolvedor tenha mais agilidade para criar suas aplicações. O JDBC foi de extrema importância para facilitar a integração do banco de dados com minha aplicação java com componentes já prontos.

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

O Statement é usado para executar instruções SQL fixas, ou seja, instruções de texto puro. Já o PreparedStatement é usado para executar instruções SQL parametrizadas.

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

A adoção do padrão DAO pode melhorar a manutenibilidade do software de várias maneiras. Primeiro, ele permite que você separe a lógica de acesso a dados da lógica de negócios, tornando o código mais fácil de entender e modificar. Em segundo lugar, ele fornece uma camada de abstração que permite que você altere o esquema do banco de dados sem afetar a lógica de negócios. Em terceiro lugar, ele permite que você reutilize o código de acesso a dados em vários aplicativos, economizando tempo e esforço.

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

Em um modelo estritamente relacional, a herança é geralmente refletida por meio de uma tabela para cada classe, incluindo a classe base e as classes derivadas. Cada tabela contém não apenas os dados da classe filha, mas também os dados da classe pai. Isso permite que consultas “independentes” entre as classes sejam mais simples e rápidas.

2º Procedimento – Alimentando a Base

Criação da classe App:

App.java 4

src > App.java > App > main(String[])

```java
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;

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

public class App {
    // Connect to your database.
    // Replace server name, username, and password with your credentials
    Run | Debug
    public static void main(String[] args) throws SQLException {

        Scanner scanner = new Scanner(System.in);
        int opcao = 1;
        int id;
        String nome;
        String logradouro;
        String cidade;
        String estado;
        String telefone;
        String email;
        String cpf;
        String cnpj;
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        while (opcao != 0) {
            System.out.println(x:"Insira a opção desejada:");
            System.out.println(x:"1 - Incluir Pessoa");
            System.out.println(x:"2 - Alterar Pessoa");
            System.out.println(x:"3 - Excluir Pessoa");
            System.out.println(x:"4 - Buscar pelo ID");
            System.out.println(x:"5 - Exibir Todos");
            System.out.println(x:"0 - Finalizar execução");
            opcao = scanner.nextInt();

            switch (opcao) {
                case 1: // código para incluir PessoaJuridica
                    System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
                    String tipo = scanner.next().toUpperCase();

                    if (tipo.charAt(index:0) == 'F') {
```

```
46
47                  if (tipo.charAt(index:0) == 'F') {
48                      PessoaFisica pessoaFisica = new PessoaFisica();
49                      boolean valido = false;
50
51                      id = 1;
52                      scanner.nextLine();
53
54                      System.out.println(x:"Digite o Nome: ");
55                      nome = scanner.nextLine();
56                      pessoaFisica.setNome(nome);
57
58                      System.out.println(x:"Digite o Logradouro: ");
59                      logradouro = scanner.nextLine();
60                      pessoaFisica.setLogradouro(logradouro);
61
62                      System.out.println(x:"Digite a Cidade: ");
63                      cidade = scanner.nextLine();
64                      pessoaFisica.setCidade(cidade);
65
66                      System.out.println(x:"Digite o Estado: ");
67                      estado = scanner.nextLine();
68                      pessoaFisica.setEstado(estado);
69
70                      System.out.println(x:"Digite o Telefone: ");
71                      telefone = scanner.nextLine();
72                      pessoaFisica.setTelefone(telefone);
73
74                      System.out.println(x:"Digite o Email: ");
75                      email = scanner.nextLine();
76                      pessoaFisica.setEmail(email);
77
78                      System.out.println(x:"Digite o CPF: ");
79                      cpf = scanner.next();
80                      pessoaFisica.setCpf(cpf);
81
82                      pessoaFisicaDAO.incluir(pessoaFisica);
83
84                  } else if (tipo.charAt(index:0) == 'J') {
85                      PessoaJuridica pessoaJuridica = new PessoaJuridica();
86
87                      boolean valido = false;
88
89                      id = 1;
90                      scanner.nextLine();
91
92                      System.out.println(x:"Digite o Nome: ");
93                      nome = scanner.nextLine();
```

```
92                  System.out.println(x:"Digite o Nome: ");
93                  nome = scanner.nextLine();
94                  pessoaJuridica.setNome(nome);
95
96                  System.out.println(x:"Digite o Logradouro: ");
97                  logradouro = scanner.nextLine();
98                  pessoaJuridica.setLogradouro(logradouro);
99
100                 System.out.println(x:"Digite a Cidade: ");
101                 cidade = scanner.nextLine();
102                 pessoaJuridica.setCidade(cidade);
103
104                 System.out.println(x:"Digite o Estado: ");
105                 estado = scanner.nextLine();
106                 pessoaJuridica.setEstado(estado);
107
108                 System.out.println(x:"Digite o Telefone: ");
109                 telefone = scanner.nextLine();
110                 pessoaJuridica.setTelefone(telefone);
111
112                 System.out.println(x:"Digite o Email: ");
113                 email = scanner.nextLine();
114                 pessoaJuridica.setEmail(email);
115
116                 System.out.println(x:"Digite o CNPJ: ");
117                 cnpj = scanner.next();
118                 pessoaJuridica.setCnpj(cnpj);
119
120                 pessoaJuridicaDAO.incluir(pessoaJuridica);
121             } else {
122                 System.out.println(x:"Tipo inválido");
123             }
124             break;
125
126         // ------------------------------------------------------------------------
127         case 2: // código para alterar
128             System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
129             tipo = scanner.next().toUpperCase();
130
131             if (tipo.charAt(index:0) == 'F') {
132                 boolean valido = false;
133                 while (!valido) {
134                     try {
135                         System.out.println(x:"Digite o ID que deseja alterar: ");
136                         id = scanner.nextInt();
137                         System.out.println(x:"Pessoa que irá ser alterada: ");
138
139                         PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
```

```
138
139                         PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
140                         if (pessoaFisica != null) {
141                             scanner.nextLine();
142                             System.out.println(x:"Digite o Nome: ");
143                             nome = scanner.nextLine();
144                             pessoaFisica.setNome(nome);
145
146                             System.out.println(x:"Digite o Logradouro: ");
147                             logradouro = scanner.nextLine();
148                             pessoaFisica.setLogradouro(logradouro);
149
150                             System.out.println(x:"Digite a Cidade: ");
151                             cidade = scanner.nextLine();
152                             pessoaFisica.setCidade(cidade);
153
154                             System.out.println(x:"Digite o Estado: ");
155                             estado = scanner.nextLine();
156                             pessoaFisica.setEstado(estado);
157
158                             System.out.println(x:"Digite o Telefone: ");
159                             telefone = scanner.nextLine();
160                             pessoaFisica.setTelefone(telefone);
161
162                             System.out.println(x:"Digite o Email: ");
163                             email = scanner.nextLine();
164                             pessoaFisica.setEmail(email);
165
166                             System.out.println(x:"Digite o CPF: ");
167                             cpf = scanner.next();
168                             pessoaFisica.setCpf(cpf);
169
170                             pessoaFisicaDAO.alterar(pessoaFisica);
171                         }
172
173                         valido = true;
174                     } catch (InputMismatchException e) {
175                         System.out.println(x:"Erro: você deve digitar um inteiro.");
176                         scanner.nextLine();
177                     }
178                 }
179
180             } else if (tipo.charAt(index:0) == 'J') {
181                 boolean valido = false;
182                 while (!valido) {
183                     try {
184                         System.out.println(x:"Digite o ID que deseja alterar: ");
```

```
                        System.out.println(x:"Digite o ID que deseja alterar: ");
                        id = scanner.nextInt();
                        System.out.println(x:"Pessoa que irá ser alterada: ");
                        PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

                        if (pessoaJuridica != null) {
                            scanner.nextLine();
                            System.out.println(x:"Digite o Nome: ");
                            nome = scanner.nextLine();
                            pessoaJuridica.setNome(nome);

                            System.out.println(x:"Digite o Logradouro: ");
                            logradouro = scanner.nextLine();
                            pessoaJuridica.setLogradouro(logradouro);

                            System.out.println(x:"Digite a Cidade: ");
                            cidade = scanner.nextLine();
                            pessoaJuridica.setCidade(cidade);

                            System.out.println(x:"Digite o Estado: ");
                            estado = scanner.nextLine();
                            pessoaJuridica.setEstado(estado);

                            System.out.println(x:"Digite o Telefone: ");
                            telefone = scanner.nextLine();
                            pessoaJuridica.setTelefone(telefone);

                            System.out.println(x:"Digite o Email: ");
                            email = scanner.nextLine();
                            pessoaJuridica.setEmail(email);

                            System.out.println(x:"Digite o CNPJ: ");
                            cnpj = scanner.next();
                            pessoaJuridica.setCnpj(cnpj);

                            pessoaJuridicaDAO.alterar(pessoaJuridica);
                        }

                        valido = true;
                    } catch (InputMismatchException e) {
                        System.out.println(x:"Erro: você deve digitar um inteiro.");
                        scanner.nextLine();
                    }
                }
            } else {
                System.out.println(x:"Tipo inválido");
            }
            break;
```

```
231                     break;
232
233                 // ------------------------------------------------------------------------
234                 case 3: // código para excluir
235                     System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
236                     tipo = scanner.next().toUpperCase();
237
238                     if (tipo.charAt(index:0) == 'F') {
239                         boolean valido = false;
240                         while (!valido) {
241                             try {
242                                 System.out.println(x:"Digite o ID que deseja excluir: ");
243                                 id = scanner.nextInt();
244                                 pessoaFisicaDAO.excluir(id);
245                                 valido = true;
246                             } catch (InputMismatchException e) {
247                                 System.out.println(x:"Erro: você deve digitar um inteiro.");
248                                 scanner.nextLine();
249                             }
250                         }
251
252                     } else if (tipo.charAt(index:0) == 'J') {
253                         boolean valido = false;
254                         while (!valido) {
255                             try {
256                                 System.out.println(x:"Digite o ID que deseja excluir: ");
257                                 id = scanner.nextInt();
258                                 pessoaJuridicaDAO.excluir(id);
259                                 valido = true;
260                             } catch (InputMismatchException e) {
261                                 System.out.println(x:"Erro: você deve digitar um inteiro.");
262                                 scanner.nextLine();
263                             }
264                         }
265                     } else {
266                         System.out.println(x:"Tipo inválido");
267                     }
268                     break;
269
270                 // ------------------------------------------------------------------------
271                 case 4: // código para obter pelo ID
272                     System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
273                     tipo = scanner.next().toUpperCase();
274
275                     if (tipo.charAt(index:0) == 'F') {
276                         boolean valido = false;
277                         while (!valido) {
278                             try {
```

```
                        try {
                            System.out.println("Digite o ID que deseja obter: ");
                            id = scanner.nextInt();
                            pessoaFisicaDAO.getPessoa(id);

                            valido = true;
                        } catch (InputMismatchException e) {
                            System.out.println("Erro: você deve digitar um inteiro.");
                            scanner.nextLine();
                        }
                    }

                } else if (tipo.charAt(0) == 'J') {
                    boolean valido = false;
                    while (!valido) {
                        try {
                            System.out.println("Digite o ID que deseja obter: ");
                            id = scanner.nextInt();
                            pessoaJuridicaDAO.getPessoa(id);

                            valido = true;
                        } catch (InputMismatchException e) {
                            System.out.println("Erro: você deve digitar um inteiro.");
                            scanner.nextLine();
                        }
                    }
                } else {
                    System.out.println("Tipo inválido");
                }
                break;

            // ------------------------------------------------------------------------
            case 5: // código para obter todos
                System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
                tipo = scanner.next().toUpperCase();

                if (tipo.charAt(0) == 'F') {
                    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
                    System.out.println("Pessoas físicas cadastradas:");
                    for (PessoaFisica p : pessoasFisicas) {
                        p.exibir();
                    }
                } else if (tipo.charAt(0) == 'J') {
                    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
                    System.out.println("Pessoas jurídicas cadastradas:");
                    for (PessoaJuridica p : pessoasJuridicas) {
                        p.exibir();
```

```
                    System.out.println("Pessoas jurídicas cadastradas:");
                    for (PessoaJuridica p : pessoasJuridicas) {
                        p.exibir();
                    }
                } else {
                    System.out.println("Tipo inválido");
                }
                break;

            // ------------------------------------------------------------------------
            case 0:
                System.out.println("Finalizando execução...");
                break;
            default:
                System.out.println("Opção inválida");
                break;
        }
    }
    scanner.close();

}
}
```

Ao executar a Classe App:

```
PS C:\Users\ricas\OneDrive\Documentos\Projetos VsCode\Java\CadastroBD> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '@C:\Users\ricas\AppData\Local\Temp\cp_ci94a6ul0i1l1ej7t2ehzhjmk.argfile' 'App'
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
```

Digitando 1, F:

```
1
F - Pessoa Física | J - Pessoa Jurídica
F
Digite o Nome:
Ricardo
Digite o Logradouro:
Rua Aqui de casa
Digite a Cidade:
Curitiba
Digite o Estado:
PR
Digite o Telefone:
1245232344
Digite o Email:
Ricardo@Alves.com
Digite o CPF:
1231232345
Pessoa física inserida com sucesso!
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
```

Digitando 4, F, 38 (38 é referente ao ID que é gerado pelo Banco de Dados):

```
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
4
F - Pessoa Física | J - Pessoa Jurídica
f
Digite o ID que deseja obter:
38
==============================
ID: 38
Nome: Ricardo
Logradouro: Rua Aqui de casa
Cidade: Curitiba
Estado: PR
Telefone: 1245232344
Email: Ricardo@Alves.com
CPF: 1231232345
==============================
```

Digitando 2, F, 38 (38 é o ID referente a pessoa criada anteriormente):

```
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
2
F - Pessoa Física | J - Pessoa Jurídica
F
Digite o ID que deseja alterar:
38
Pessoa que irá ser alterada:
==============================
ID: 38
Nome: Ricardo
Logradouro: Rua Aqui de casa
Cidade: Curitiba
Estado: PR
Telefone: 1245232344
Email: Ricardo@Alves.com
CPF: 1231232345
==============================
Digite o Nome:
Ricardo
Digite o Logradouro:
Rua Manoel Gomes
Digite a Cidade:
Jaragua do Sul
Digite o Estado:
SC
Digite o Telefone:
12356332
Digite o Email:
ricardo@alves.com
Digite o CPF:
23442344123
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
```

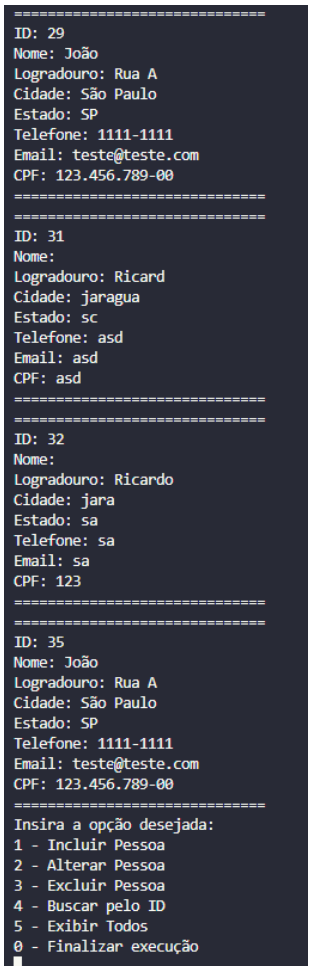
Digitando 4, F, 38 novamente:

```
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
4
F - Pessoa Física | J - Pessoa Jurídica
f
Digite o ID que deseja obter:
38
==============================
ID: 38
Nome: Ricardo
Logradouro: Rua Manoel Gomes
Cidade: Jaragua do Sul
Estado: SC
Telefone: 12356332
Email: ricardo@alves.com
CPF: 23442344123
==============================
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
```

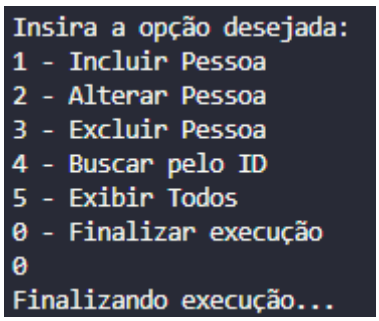
Digitando 3, F, 38 e em seguida 4, F, 38:

```
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
3
F - Pessoa Física | J - Pessoa Jurídica
F
Digite o ID que deseja excluir:
38
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
4
F - Pessoa Física | J - Pessoa Jurídica
f
Digite o ID que deseja obter:
38
Este ID não existe!
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
```

Digitando 5, F:



Digitando 0:



Ao final do projeto todas as funcionalidades estão funcionando como o proposto. Para o desenvolvimento usei o editor Visual Studio Code, e incluí ao projeto os drivers JDBC como pedido.

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

A persistência de dados em arquivos é mais fácil de der executada, porém não tem a robustês do banco de dados. Por sua vez o banco de dados é mais robusto para suportar muitos arquivos e persistência simultânea de dados. Porem o banco tem uma certa dificuldade para ser implementado e com o banco de dados é preciso ter um ambiente mais robusto e configurado para que a aplicação funcione.

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

O uso de expressões lambda em Java pode simplificar a impressão dos valores contidos nas entidades, tornando o código mais conciso e legível.

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, sem o uso de um objeto, precisam ser marcados como static porque o método main é um método estático. Isso significa que ele pertence à classe e não a uma instância da classe. Como resultado, ele pode ser chamado diretamente pela classe, sem a necessidade de criar um objeto da classe.

Conclusão

Neste trabalho, desenvolvemos uma aplicação usando SQL Server, Java e JDBC para integrar a aplicação com o banco de dados. A aplicação permite criar, alterar, excluir, visualizar e buscar pessoas jurídicas e físicas, usando os conceitos de orientação a objetos e persistência de dados. Demonstramos o funcionamento da aplicação através de testes e exemplos, mostrando as vantagens de usar essas tecnologias para gerenciar informações de forma eficiente e segura. Concluímos que a aplicação atende aos requisitos propostos e pode ser usada como uma ferramenta útil para o cadastro e a consulta de pessoas.