



## **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 X
src > cadastrobd > model > Pessoa.java > Pessoa > getEstado()
1  package cadastrobd.model;
2
3  public class Pessoa {
4      // Campos da classe
5      private int id;
6      private String nome;
7      private String logradouro;
8      private String cidade;
9      private String estado;
10     private String telefone;
11     private String email;
12
13     // Construtor padrão
14     public Pessoa() {
15         this.id = 0;
16         this.nome = "";
17         this.logradouro = "";
18         this.cidade = "";
19         this.estado = "";
20         this.telefone = "";
21         this.email = "";
22     }
23
24     // Construtor completo
25     public Pessoa(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
26         this.id = id;
27         this.nome = nome;
28         this.logradouro = logradouro;
29         this.cidade = cidade;
30         this.estado = estado;
31         this.telefone = telefone;
32         this.email = email;
33     }
34
35
36
37     // Getters e setters
38     public int getId() {
39         return this.id;
40     }
41
42     public void setId(int id) {
43         this.id = id;
44     }
45
46     public String getNome() {
47         return this.nome;
48     }
49 }
```

```

46     public String getNome() {
47         return this.nome;
48     }
49
50     public void setNome(String nome) {
51         this.nome = nome;
52     }
53
54     public String getLogradouro() {
55         return this.logradouro;
56     }
57
58     public void setLogradouro(String logradouro) {
59         this.logradouro = logradouro;
60     }
61
62     public String getCidade() {
63         return this.cidade;
64     }
65
66     public void setCidade(String cidade) {
67         this.cidade = cidade;
68     }
69
70     public String getEstado() {
71         return this.estado;
72     }
73
74     public void setEstado(String estado) {
75         this.estado = estado;
76     }
77
78     public String getTelefone() {
79         return this.telefone;
80     }
81
82     public void setTelefone(String telefone) {
83         this.telefone = telefone;
84     }
85
86     public String getEmail() {
87         return this.email;
88     }
89
90     public void setEmail(String email) {
91         this.email = email;
92     }
93

```

```

93
94     // Método exibir
95     public void exibir() {
96         System.out.println(x:"=====");
97         System.out.println("ID: " + this.id);
98         System.out.println("Nome: " + this.nome);
99         System.out.println("Logradouro: " + this.logradouro);
100        System.out.println("Cidade: " + this.cidade);
101        System.out.println("Estado: " + this.estado);
102        System.out.println("Telefone: " + this.telefone);
103        System.out.println("Email: " + this.email);
104    }
105

```

## Criação da classe PessoaFisica:

```
PessoaFisica.java X
src > cadastrobd > model > PessoaFisica.java > PessoaFisica > exibir()
1 package cadastrobd.model;
2
3 import java.io.Serializable;
4
5 public class PessoaFisica extends Pessoa implements Serializable {
6     private String cpf;
7
8     public PessoaFisica() {
9     }
10
11     public PessoaFisica(int id, String nome, String logradouro,
12                         String cidade, String estado, String telefone,
13                         String email, String cpf) {
14         super(id, nome, logradouro, cidade, estado, telefone, email);
15         this.cpf = cpf;
16     }
17
18     public String getCpf() {
19         return cpf;
20     }
21
22     public void setCpf(String cpf) {
23         this.cpf = cpf;
24     }
25
26     @Override
27     public void exibir() {
28         super.exibir();
29         System.out.println("CPF: " + this.cpf );
30         System.out.println(x:"=====");
31     }
32
33     public void remove(PessoaFisica pessoaFisica) {
34     }
35 }
36
```

## Criação da classe PessoaJuridica:

```
PessoaJuridica.java X
src > cadastrobd > model > PessoaJuridica.java > {} cadastrobd.model
1 package cadastrobd.model;
2
3
4 import java.io.Serializable;
5
6 public class PessoaJuridica extends Pessoa implements Serializable {
7     private String cnpj;
8
9     public PessoaJuridica() {
10    }
11
12     public PessoaJuridica(int id, String nome, String logradouro,
13                         String cidade, String estado, String telefone, String email,
14                         String cnpj) {
15         super(id, nome, logradouro, cidade, estado, telefone, email);
16         this.cnpj = cnpj;
17     }
18
19     public String getCnpj() {
20         return cnpj;
21     }
22
23     public void setCnpj(String cnpj) {
24         this.cnpj = cnpj;
25     }
26
27     @Override
28     public void exibir() {
29         super.exibir();
30         System.out.println("CNPJ: " + this.cnpj);
31         System.out.println(x:"=====");
32     }
33 }
34
35
```

## Criação da classe ConectoBD:

```
ConectorBD.java X
src > cadastrobd > model > ConectorBD.java > ...
1 package cadastrobd.model;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.sql.Statement;
8
9
10
11 public class ConectorBD {
12     // Constantes para armazenar os dados de conexão com o banco de dados
13     // Você pode alterar esses valores de acordo com a sua configuração
14
15     private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=Loja;user=loja;password=loja;encrypt=true;trustServerCertificate=true;";
16
17
18     public static Connection getConnection() throws SQLException {
19         try {
20             // Carregar o driver JDBC do SQL Server
21             Class.forName(className:"com.microsoft.sqlserver.jdbc.SQLServerDriver");
22
23         } catch (ClassNotFoundException e) {
24             // Tratar a exceção se o driver não for encontrado
25             e.printStackTrace();
26         }
27         return DriverManager.getConnection(URL);
28     }
29     // Criar o pacote cadastrobd.model.util package cadastrobd.model.util;
30     //Método getPrepared, para retornar um objeto do tipo PreparedStatement a partir de um SQL fornecido com parâmetro
31     public static PreparedStatement getPrepared(String sql) throws SQLException {
32         // Obter uma conexão com o banco de dados
33         Connection con = getConnection();
34         // Criar e retornar um objeto do tipo PreparedStatement usando o SQL fornecido
35         return con.prepareStatement(sql);
36     }
37
38     // Método getSelect, para retornar o ResultSet relacionado a uma consulta
39     public static ResultSet getSelect(String sql) throws SQLException {
40         // Obter um objeto do tipo PreparedStatement usando o método getPrepared
41         PreparedStatement ps = getPrepared(sql);
42         // Executar a consulta e retornar o ResultSet resultante
43         return ps.executeQuery();
44     }
45
46     // Método close sobrecarregado para Statement, para garantir o fechamento do objeto
47     public static void close(Statement st) {
48         // Verificar se o objeto não é nulo
```

```

46 // Método close sobrecarregado para Statement, para garantir o fechamento do objeto
47 public static void close(Statement st) {
48     // Verificar se o objeto não é nulo
49     if (st != null) {
50         try {
51             // Fechar o objeto
52             st.close();
53         } catch (SQLException e) {
54             // Imprimir a mensagem de erro
55             e.printStackTrace();
56         }
57     }
58 }
59
60 // Método close sobrecarregado para ResultSet, para garantir o fechamento do objeto
61 public static void close(ResultSet rs) {
62     // Verificar se o objeto não é nulo
63     if (rs != null) {
64         try {
65             // Fechar o objeto
66             rs.close();
67         } catch (SQLException e) {
68             // Imprimir a mensagem de erro
69             e.printStackTrace();
70         }
71     }
72 }
73
74 // Método close sobrecarregado para Connection, para garantir o fechamento do objeto
75 public static void close(Connection con) {
76     // Verificar se o objeto não é nulo
77     if (con != null) {
78         try {
79             // Fechar o objeto
80             con.close();
81         } catch (SQLException e) {
82             // Imprimir a mensagem de erro
83             e.printStackTrace();
84         }
85     }
86 }
87 // Classe SequenceManager
88 public class SequenceManager {
89     // Método getValue, recebendo o nome da sequência como parâmetro e retornando o próximo valor
90     public static int getValue(String sequence) throws SQLException {
91         // Inicializar uma variável para armazenar o valor da sequência
92         int value = 0;
93         // Criar uma string com o SQL para obter o valor atual da sequência
94
95         String sql = "SELECT NEXT VALUE FOR " + sequence ;
96         // Obter o ResultSet relacionado à consulta usando o método getSelect da classe ConectorBD
97         ResultSet rs = ConectorBD.getSelect(sql);
98         // Verificar se o ResultSet tem algum resultado
99         if (rs.next()) {
100             // Obter o valor da sequência a partir da primeira coluna do ResultSet
101             value = rs.getInt(columnIndex:1);
102         }
103         // Fechar o ResultSet usando o método close da classe ConectorBD
104         ConectorBD.close(rs);
105         // Retornar o valor da sequência
106         return value;
107     }
108 }
109

```

Criação da classe PessoaFisicaDAO:

```

PessoaFisicaDAO.java X
src > cadastrobd > model > PessoaFisicaDAO.java > PessoaFisicaDAO > getPessoa(int)
1  package cadastrobd.model;
2  import java.sql.Statement;
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.util.ArrayList;
8  import java.util.List;
9
10 //import cadastrobd.model.ConectorBD.SequenceManager;
11
12 public class PessoaFisicaDAO {
13
14     // Método para obter uma pessoa física a partir do seu id
15     public PessoaFisica getPessoa(int id) throws SQLException {
16         PessoaFisica pf = null;
17         Connection conn = ConectorBD.getConnection();
18         PreparedStatement ps = null;
19         ResultSet rs = null;
20         try {
21             ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaFisica pf ON p.id = pf.pessoa_id WHERE p.id = ?");
22             ps.setInt(parameterIndex:1, id);
23             rs = ps.executeQuery();
24             if (rs.next()) {
25                 pf = new PessoaFisica();
26                 pf.setId(rs.getInt(columnLabel:"id"));
27                 pf.setNome(rs.getString(columnLabel:"nome"));
28                 pf.setLogradouro(rs.getString(columnLabel:"logradouro"));
29                 pf.setCidade(rs.getString(columnLabel:"cidade"));
30                 pf.setEstado(rs.getString(columnLabel:"estado"));
31                 pf.setTelefone(rs.getString(columnLabel:"telefone"));
32                 pf.setEmail(rs.getString(columnLabel:"email"));
33                 pf.setCpf(rs.getString(columnLabel:"cpf"));
34             }
35         } catch (SQLException e) {
36             e.printStackTrace();
37         } finally {
38             ConectorBD.close(conn);
39         }
40         if (pf != null){
41             pf.exibir();
42         }else{
43             System.out.println(x:"Este ID não existe!");
44         }
45
46         return pf;
47     }
48

```

```

49 // Método para obter todas as pessoas físicas do banco de dados
50 public List<PessoaFisica> getPessoas() throws SQLException {
51     List<PessoaFisica> pessoas = new ArrayList<>();
52     Connection conn = ConectorBD.getConnection();
53     PreparedStatement ps = null;
54     ResultSet rs = null;
55     try {
56         ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaFisica pf ON p.id = pf.pessoa_id");
57         rs = ps.executeQuery();
58         while (rs.next()) {
59             PessoaFisica pf = new PessoaFisica();
60             pf.setId(rs.getInt(columnLabel:"id"));
61             pf.setNome(rs.getString(columnLabel:"nome"));
62             pf.setLogradouro(rs.getString(columnLabel:"logradouro"));
63             pf.setCidade(rs.getString(columnLabel:"cidade"));
64             pf.setEstado(rs.getString(columnLabel:"estado"));
65             pf.setTelefone(rs.getString(columnLabel:"telefone"));
66             pf.setEmail(rs.getString(columnLabel:"email"));
67             pf.setCpf(rs.getString(columnLabel:"cpf"));
68             pessoas.add(pf);
69         }
70     } catch (SQLException e) {
71         e.printStackTrace();
72     } finally {
73         ConectorBD.close(conn);
74     }
75     return pessoas;
76 }
77
78 // Método para incluir uma pessoa física nas tabelas Pessoa e PessoaFisica
79 public void incluir(PessoaFisica pf) throws SQLException {
80     Connection conn = ConectorBD.getConnection();
81     PreparedStatement ps = null;
82     try {
83         // Inserir na tabela Pessoa
84         ps = conn.prepareStatement(sql:"INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
85         ps.setString(parameterIndex:1, pf.getNome());
86         ps.setString(parameterIndex:2, pf.getLogradouro());
87         ps.setString(parameterIndex:3, pf.getCidade());
88         ps.setString(parameterIndex:4, pf.getEstado());
89         ps.setString(parameterIndex:5, pf.getTelefone());
90         ps.setString(parameterIndex:6, pf.getEmail());
91
92         int rows = ps.executeUpdate();
93
94         // Verificar se a inserção foi bem sucedida
95         if (rows > 0) {
96             if (rows > 0) {
97                 // Obter o ResultSet com o id gerado pelo banco
98                 ResultSet rs = ps.getGeneratedKeys();
99
100                 // Verificar se o ResultSet tem algum resultado
101                 if (rs.next()) {
102                     // Obter o id da pessoa a partir da primeira coluna do ResultSet
103                     int id = rs.getInt(columnIndex:1);
104
105                     // Usar o mesmo objeto PreparedStatement para inserir os dados na tabela PessoaFisica
106                     ps = conn.prepareStatement(sql:"INSERT INTO PessoaFisica (cpf, pessoa_id) VALUES (?, ?)");
107                     ps.setString(parameterIndex:1, pf.getCpf());
108                     ps.setInt(parameterIndex:2, id);
109                     rows = ps.executeUpdate();
110
111                     if (rows > 0) {
112                         System.out.println(x:"Pessoa fisica inserida com sucesso!");
113                     } else {
114                         System.out.println(x:"Falha ao inserir a pessoa física!");
115                     }
116                 } else {
117                     System.out.println(x:"Falha ao obter o id da pessoa!");
118                 }
119             } else {
120                 System.out.println(x:"Falha ao inserir a pessoa!");
121             }
122         } catch (SQLException e) {
123             e.printStackTrace();
124         } finally {
125             ConectorBD.close(conn);
126         }
127     }
128
129 // Método para alterar os dados de uma pessoa física
130 public void alterar(PessoaFisica pf) throws SQLException {
131     Connection conn = ConectorBD.getConnection();
132     PreparedStatement ps = null;
133     try {
134         // Alterar na tabela Pessoa
135         ps = conn.prepareStatement(sql:"UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?");
136         ps.setString(parameterIndex:1, pf.getNome());
137         ps.setString(parameterIndex:2, pf.getLogradouro());
138         ps.setString(parameterIndex:3, pf.getCidade());
139         ps.setString(parameterIndex:4, pf.getEstado());
140         ps.setString(parameterIndex:5, pf.getTelefone());
141         ps.setString(parameterIndex:6, pf.getEmail());
142         ps.setInt(parameterIndex:7, pf.getId());

```



```

140         ps.setString(parameterIndex:6, pf.getNome());
141         ps.setInt(parameterIndex:7, pf.getId());
142         ps.executeUpdate();
143
144         // Alterar na tabela PessoaFisica
145         ps = conn.prepareStatement(sql:"UPDATE PessoaFisica SET cpf = ? WHERE pessoa_id = ?");
146         ps.setString(parameterIndex:1, pf.getCpf());
147         ps.setInt(parameterIndex:2, pf.getId());
148         ps.executeUpdate();
149     } catch (SQLException e) {
150         e.printStackTrace();
151     } finally {
152         ConectorBD.close(conn);
153     }
154 }
155
156 // Método para remover uma pessoa física das tabelas Pessoa e PessoaFisica
157 public void excluir(int id) throws SQLException {
158     Connection conn = ConectorBD.getConnection();
159     PreparedStatement ps = null;
160     try {
161         // Excluir da tabela PessoaFisica
162         ps = conn.prepareStatement(sql:"DELETE FROM PessoaFisica WHERE pessoa_id = ?");
163         ps.setInt(parameterIndex:1, id);
164         ps.executeUpdate();
165
166         // Excluir da tabela Pessoa
167         ps = conn.prepareStatement(sql:"DELETE FROM Pessoa WHERE id = ?");
168         ps.setInt(parameterIndex:1, id);
169         ps.executeUpdate();
170     } catch (SQLException e) {
171         e.printStackTrace();
172     } finally {
173         ConectorBD.close(conn);
174     }
175 }
176 }

```

Criação da Classe PessoaJuridicaDAO:

```

PessoaJuridicaDAO.java X
src > cadastrobd > model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > getPessoa(int)
1 package cadastrobd.model;
2 import java.sql.Connection;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.sql.Statement;
7 import java.util.ArrayList;
8 import java.util.List;
9
10
11 public class PessoaJuridicaDAO {
12
13     // Método para obter uma pessoa física a partir do seu id
14     public PessoaJuridica getPessoa(int id) throws SQLException {
15         PessoaJuridica pj = null;
16         Connection conn = ConectorBD.getConnection();
17         PreparedStatement ps = null;
18         ResultSet rs = null;
19         try {
20             ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.id = pj.pessoa_id WHERE p.id = ?");
21             ps.setInt(parameterIndex:1, id);
22             rs = ps.executeQuery();
23             if (rs.next()) {
24                 pj = new PessoaJuridica();
25                 pj.setId(rs.getInt(columnLabel:"id"));
26                 pj.setNome(rs.getString(columnLabel:"nome"));
27                 pj.setLogradouro(rs.getString(columnLabel:"logradouro"));
28                 pj.setCidade(rs.getString(columnLabel:"cidade"));
29                 pj.setEstado(rs.getString(columnLabel:"estado"));
30                 pj.setTelefone(rs.getString(columnLabel:"telefone"));
31                 pj.setEmail(rs.getString(columnLabel:"email"));
32                 pj.setCnpj(rs.getString(columnLabel:"cnpj"));
33             }
34         } catch (SQLException e) {
35             e.printStackTrace();
36         } finally {
37             ConectorBD.close(conn);
38         }
39         if (pj != null){
40             pj.exibir();
41         }else{
42             System.out.println("Este ID não existe!");
43         }
44         return pj;
45     }
46
47     // Método para obter todas as pessoas físicas do banco de dados
48     public List<PessoaJuridica> getPessoas() throws SQLException {

```

```

47     // Método para obter todas as pessoas físicas do banco de dados
48     public List<PessoaJuridica> getPessoas() throws SQLException {
49         List<PessoaJuridica> pessoas = new ArrayList<>();
50         Connection conn = ConectorBD.getConnection();
51         PreparedStatement ps = null;
52         ResultSet rs = null;
53         try {
54             ps = conn.prepareStatement(sql:"SELECT * FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.id = pj.pessoa_id");
55             rs = ps.executeQuery();
56             while (rs.next()) {
57                 PessoaJuridica pj = new PessoaJuridica();
58                 pj.setId(rs.getInt(columnLabel:"id"));
59                 pj.setNome(rs.getString(columnLabel:"nome"));
60                 pj.setLogradouro(rs.getString(columnLabel:"logradouro"));
61                 pj.setCidade(rs.getString(columnLabel:"cidade"));
62                 pj.setEstado(rs.getString(columnLabel:"estado"));
63                 pj.setTelefone(rs.getString(columnLabel:"telefone"));
64                 pj.setEmail(rs.getString(columnLabel:"email"));
65                 pj.setCnpj(rs.getString(columnLabel:"cnpj"));
66                 pessoas.add(pj);
67             }
68         } catch (SQLException e) {
69             e.printStackTrace();
70         } finally {
71             ConectorBD.close(conn);
72         }
73         return pessoas;
74     }
75
76     // Método para incluir uma pessoa física nas tabelas Pessoa e PessoaJuridica
77     public void incluir(PessoaJuridica pj) throws SQLException {
78         Connection conn = ConectorBD.getConnection();
79         PreparedStatement ps = null;
80         try {
81             // Inserir na tabela Pessoa
82             ps = conn.prepareStatement(sql:"INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
83             ps.setString(parameterIndex:1, pj.getNome());
84             ps.setString(parameterIndex:2, pj.getLogradouro());
85             ps.setString(parameterIndex:3, pj.getCidade());
86             ps.setString(parameterIndex:4, pj.getEstado());
87             ps.setString(parameterIndex:5, pj.getTelefone());
88             ps.setString(parameterIndex:6, pj.getEmail());
89
90             int rows = ps.executeUpdate();
91
92             // Verificar se a inserção foi bem sucedida
93             if (rows > 0) {

```

```

92         // Verificar se a inserção foi bem sucedida
93         if (rows > 0) {
94             // Obter o ResultSet com o id gerado pelo banco
95             ResultSet rs = ps.getGeneratedKeys();
96
97             // Verificar se o ResultSet tem algum resultado
98             if (rs.next()) {
99                 // Obter o id da pessoa a partir da primeira coluna do ResultSet
100                 int id = rs.getInt(columnIndex:1);
101
102                 // Usar o mesmo objeto PreparedStatement para inserir os dados na tabela PessoaJuridica
103                 ps = conn.prepareStatement(sql:"INSERT INTO PessoaJuridica (cnpj, pessoa_id) VALUES (?, ?)");
104                 ps.setString(parameterIndex:1, pj.getCnpj());
105                 ps.setInt(parameterIndex:2, id);
106                 rows = ps.executeUpdate();
107
108                 if (rows > 0) {
109                     System.out.println(x:"Pessoa jurídica inserida com sucesso!");
110                 } else {
111                     System.out.println(x:"Falha ao inserir a pessoa jurídica!");
112                 }
113             } else {
114                 System.out.println(x:"Falha ao obter o id da pessoa!");
115             }
116         } else {
117             System.out.println(x:"Falha ao inserir a pessoa!");
118         }
119     }
120
121     } catch (SQLException e) {
122         e.printStackTrace();
123     } finally {
124         ConectorBD.close(conn);
125     }
126 }
127
128 // Método para alterar os dados de uma pessoa física
129 public void alterar(PessoaJuridica pj) throws SQLException {
130     Connection conn = ConectorBD.getConnection();
131     PreparedStatement ps = null;
132     try {
133         // Alterar na tabela Pessoa
134         ps = conn.prepareStatement(sql:"UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?");
135         ps.setString(parameterIndex:1, pj.getNome());
136         ps.setString(parameterIndex:2, pj.getLogradouro());
137         ps.setString(parameterIndex:3, pj.getCidade());
138         ps.setString(parameterIndex:4, pj.getEstado());
139
140         ps.setString(parameterIndex:2, pj.getLogradouro());
141         ps.setString(parameterIndex:3, pj.getCidade());
142         ps.setString(parameterIndex:4, pj.getEstado());
143         ps.setString(parameterIndex:5, pj.getTelefone());
144         ps.setString(parameterIndex:6, pj.getEmail());
145         ps.setInt(parameterIndex:7, pj.getId());
146         ps.executeUpdate();
147
148         // Alterar na tabela PessoaJuridica
149         ps = conn.prepareStatement(sql:"UPDATE PessoaJuridica SET cnpj = ? WHERE pessoa_id = ?");
150         ps.setString(parameterIndex:1, pj.getCnpj());
151         ps.setInt(parameterIndex:2, pj.getId());
152         ps.executeUpdate();
153         System.out.println(x:"Pessoa jurídica alterada com sucesso!");
154     } catch (SQLException e) {
155         e.printStackTrace();
156     } finally {
157         ConectorBD.close(conn);
158     }
159 }
160
161 // Método para remover uma pessoa física das tabelas Pessoa e PessoaJuridica
162 public void excluir(int id) throws SQLException {
163     Connection conn = ConectorBD.getConnection();
164     PreparedStatement ps = null;
165     try {
166         // Excluir da tabela PessoaJuridica
167         ps = conn.prepareStatement(sql:"DELETE FROM PessoaJuridica WHERE pessoa_id = ?");
168         ps.setInt(parameterIndex:1, id);
169         ps.executeUpdate();
170
171         // Excluir da tabela Pessoa
172         ps = conn.prepareStatement(sql:"DELETE FROM Pessoa WHERE id = ?");
173         ps.setInt(parameterIndex:1, id);
174         ps.executeUpdate();
175     } catch (SQLException e) {
176         e.printStackTrace();
177     } finally {
178         ConectorBD.close(conn);
179     }
180 }
181 }

```

Criação da classe Main:

```

Main.java X
src > Main.java > ...
1  import java.sql.SQLException;
2  import java.util.List;
3  import cadastrbd.model.PessoaFisica;
4  import cadastrbd.model.PessoaFisicaDAO;
5  import cadastrbd.model.PessoaJuridica;
6  import cadastrbd.model.PessoaJuridicaDAO;
7
8  public class Main {
9      public static void main(String[] args) throws SQLException {
10         // Instanciar uma pessoa física e persistir no banco de dados
11         PessoaFisica pf = new PessoaFisica(id:17, nome:"João", logradouro:"Rua A", cidade:"São Paulo",
12         estado:"SP", telefone:"1111-1111", email:"teste@teste.com",
13         cpf:"123.456.789-00");
14         PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
15         pfDAO.getPessoa(id:1);
16
17         pfDAO.incluir(pf);
18
19         // // // Alterar os dados da pessoa física no banco
20         pf.setNome(nome:"João Pedro Paulo");
21         pf.setTelefone(telefone:"2222-2222");
22         pf.setCpf(cpf:"234.456.456-12");
23         pfDAO.alterar(pf);
24         System.out.println(x:"Pessoa física alterada com sucesso!");
25
26         // Consultar todas as pessoas físicas do banco de dados e listar no console
27         List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
28         System.out.println(x:"Pessoas físicas cadastradas:");
29         for (PessoaFisica p : pessoasFisicas) {
30             p.exibir();
31         }
32
33         // // //Excluir a pessoa física criada anteriormente no banco
34         pfDAO.excluir(pf.getId());
35         System.out.println(x:"Pessoa física excluída com sucesso!");
36
37         // Instanciar uma pessoa jurídica e persistir no banco de dados
38         PessoaJuridica pj = new PessoaJuridica(id:2, nome:"Empresa X", logradouro:"Rua B", cidade:"Rio de Janeiro",
39         estado:"RJ", telefone:"3333-3333", email:"teste@teste.com",
40         cnpj:"12.345.6780001-90");
41         PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
42         pjDAO.getPessoa(id:3);
43         pjDAO.incluir(pj);
44         System.out.println(x:"Pessoa jurídica incluída com sucesso!");
45
46         // Alterar os dados da pessoa jurídica no banco
47         pj.setNome(nome:"Empresa Y");
48         pjDAO.alterar(pj);
49         System.out.println(x:"Pessoa jurídica alterada com sucesso!");
50
51         // Consultar todas as pessoas jurídicas do banco e listar no console
52         List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
53         System.out.println(x:"Pessoas jurídicas cadastradas:");
54         for (PessoaJuridica p : pessoasJuridicas) {
55             p.exibir();
56         }
57
58         // Excluir a pessoa jurídica criada anteriormente no banco
59         pjDAO.excluir(pj.getId());
60         System.out.println(x:"Pessoa jurídica excluída com sucesso!");
61     }
62 }
63

```

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_c194a6ul0i11ej7t2ehzhjmk,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
=====
```

```

=====
ID: 30
Nome: CARROS E MOTOS LTMD
Logradouro: rua logo ali
Cidade: canadense
Estado: TO
Telefone: 123452
Email: carros@motos.com
CNPJ: 35634567
=====
=====
ID: 36
Nome: Empresa X
Logradouro: Rua B
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 3333-3333
Email: teste@teste.com
CNPJ: 12.345.6780001-90
=====
Pessoa jurídica excluída com sucesso!
PS C:\Users\ricas\OneDrive\Documentos\Projetos VsCode\Java\CadastroBD>

```

Para executar

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

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

Estes componentes facilitam 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 X

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

```
1  import java.sql.PreparedStatement;
2  import java.sql.ResultSet;
3  import java.sql.SQLException;
4  import java.util.InputMismatchException;
5  import java.util.List;
6  import java.util.Scanner;
7
8  import cadastrobd.model.PessoaFisica;
9  import cadastrobd.model.PessoaFisicaDAO;
10 import cadastrobd.model.PessoaJuridica;
11 import cadastrobd.model.PessoaJuridicaDAO;
12
13 public class App {
14     // Connect to your database.
15     // Replace server name, username, and password with your credentials
16     Run | Debug
17     public static void main(String[] args) throws SQLException {
18
19         Scanner scanner = new Scanner(System.in);
20         int opcao = 1;
21         int id;
22         String nome;
23         String logradouro;
24         String cidade;
25         String estado;
26         String telefone;
27         String email;
28         String cpf;
29         String cnpj;
30         PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
31         PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
32
33         while (opcao != 0) {
34             System.out.println(x:"Insira a opção desejada:");
35             System.out.println(x:"1 - Incluir Pessoa");
36             System.out.println(x:"2 - Alterar Pessoa");
37             System.out.println(x:"3 - Excluir Pessoa");
38             System.out.println(x:"4 - Buscar pelo ID");
39             System.out.println(x:"5 - Exibir Todos");
40             System.out.println(x:"0 - Finalizar execução");
41             opcao = scanner.nextInt();
42
43             switch (opcao) {
44                 case 1: // código para incluir PessoaJuridica
45                     System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
46                     String tipo = scanner.next().toUpperCase();
47
48                     if (tipo.charAt(index:0) == 'F') {
49                         PessoaFisica pessoaFisica = new PessoaFisica();
50                     }
51                 }
52             }
53 }
```

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

```

192         System.out.println(x:"Digite o Nome: ");
193         nome = scanner.nextLine();
194         pessoaJuridica.setNome(nome);
195
196         System.out.println(x:"Digite o Logradouro: ");
197         logradouro = scanner.nextLine();
198         pessoaJuridica.setLogradouro(logradouro);
199
200         System.out.println(x:"Digite a Cidade: ");
201         cidade = scanner.nextLine();
202         pessoaJuridica.setCidade(cidade);
203
204         System.out.println(x:"Digite o Estado: ");
205         estado = scanner.nextLine();
206         pessoaJuridica.setEstado(estado);
207
208         System.out.println(x:"Digite o Telefone: ");
209         telefone = scanner.nextLine();
210         pessoaJuridica.setTelefone(telefone);
211
212         System.out.println(x:"Digite o Email: ");
213         email = scanner.nextLine();
214         pessoaJuridica.setEmail(email);
215
216         System.out.println(x:"Digite o CNPJ: ");
217         cnpj = scanner.next();
218         pessoaJuridica.setCnpj(cnpj);
219
220         pessoaJuridicaDAO.incluir(pessoaJuridica);
221     } else {
222         System.out.println(x:"Tipo inválido");
223     }
224     break;
225
226     // -----
227     case 2: // código para alterar
228         System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
229         tipo = scanner.next().toUpperCase();
230
231         if (tipo.charAt(index:0) == 'F') {
232             boolean valido = false;
233             while (!valido) {
234                 try {
235                     System.out.println(x:"Digite o ID que deseja alterar: ");
236                     id = scanner.nextInt();
237                     System.out.println(x:"Pessoa que irá ser alterada: ");
238
239                     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: ");

```

```

184     System.out.println(x:"Digite o ID que deseja alterar: ");
185     id = scanner.nextInt();
186     System.out.println(x:"Pessoa que irá ser alterada: ");
187     PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
188
189     if (pessoaJuridica != null) {
190         scanner.nextLine();
191         System.out.println(x:"Digite o Nome: ");
192         nome = scanner.nextLine();
193         pessoaJuridica.setNome(nome);
194
195         System.out.println(x:"Digite o Logradouro: ");
196         logradouro = scanner.nextLine();
197         pessoaJuridica.setLogradouro(logradouro);
198
199         System.out.println(x:"Digite a Cidade: ");
200         cidade = scanner.nextLine();
201         pessoaJuridica.setCidade(cidade);
202
203         System.out.println(x:"Digite o Estado: ");
204         estado = scanner.nextLine();
205         pessoaJuridica.setEstado(estado);
206
207         System.out.println(x:"Digite o Telefone: ");
208         telefone = scanner.nextLine();
209         pessoaJuridica.setTelefone(telefone);
210
211         System.out.println(x:"Digite o Email: ");
212         email = scanner.nextLine();
213         pessoaJuridica.setEmail(email);
214
215         System.out.println(x:"Digite o CNPJ: ");
216         cnpj = scanner.next();
217         pessoaJuridica.setCnpj(cnpj);
218
219         pessoaJuridicaDAO.alterar(pessoaJuridica);
220     }
221
222     valido = true;
223 } catch (InputMismatchException e) {
224     System.out.println(x:"Erro: você deve digitar um inteiro.");
225     scanner.nextLine();
226 }
227 }
228 } else {
229     System.out.println(x:"Tipo inválido");
230 }
231 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 {

```

```

278         try {
279             System.out.println(x:"Digite o ID que deseja obter: ");
280             id = scanner.nextInt();
281             pessoaFisicaDAO.getPessoa(id);
282
283             valido = true;
284         } catch (InputMismatchException e) {
285             System.out.println(x:"Erro: você deve digitar um inteiro.");
286             scanner.nextLine();
287         }
288     }
289
290     } else if (tipo.charAt(index:0) == 'J') {
291         boolean valido = false;
292         while (!valido) {
293             try {
294                 System.out.println(x:"Digite o ID que deseja obter: ");
295                 id = scanner.nextInt();
296                 pessoaJuridicaDAO.getPessoa(id);
297
298                 valido = true;
299             } catch (InputMismatchException e) {
300                 System.out.println(x:"Erro: você deve digitar um inteiro.");
301                 scanner.nextLine();
302             }
303         }
304     } else {
305         System.out.println(x:"Tipo inválido");
306     }
307     break;
308
309     // -----
310     case 5: // código para obter todos
311         System.out.println(x:"F - Pessoa Física | J - Pessoa Jurídica");
312         tipo = scanner.next().toUpperCase();
313
314         if (tipo.charAt(index:0) == 'F') {
315             List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
316             System.out.println(x:"Pessoas físicas cadastradas:");
317             for (PessoaFisica p : pessoasFisicas) {
318                 p.exibir();
319             }
320         } else if (tipo.charAt(index:0) == 'J') {
321             List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
322             System.out.println(x:"Pessoas jurídicas cadastradas:");
323             for (PessoaJuridica p : pessoasJuridicas) {
324                 p.exibir();

```

```

322             System.out.println(x:"Pessoas jurídicas cadastradas:");
323             for (PessoaJuridica p : pessoasJuridicas) {
324                 p.exibir();
325             }
326         } else {
327             System.out.println(x:"Tipo inválido");
328         }
329         break;
330
331     // -----
332     case 0:
333         System.out.println(x:"Finalizando execução...");
334         break;
335     default:
336         System.out.println(x:"Opção inválida");
337         break;
338 }
339 scanner.close();
340
341 }
342
343 }
344

```

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_c194a6ul0i111ej7t2ehzhjmk.angfile' '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:

```

=====
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:
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
=====
Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução

```

Digitando 0:

```

Insira a opção desejada:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar execução
0
Finalizando execução...

```

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 ser executada, porém não tem a robustez do banco de dados. Por sua vez o banco de dados é mais robusto para suportar muitos arquivos e persistência simultânea de dados. Porém 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.