



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

Leonardo Schaffer Mota – 2022.05.09098-1

Polo centro – Santo André – SP
BackEnd sem banco não tem– 2023.2 – 2023.4

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.

1º Procedimento – Mapeamento Objeto-Relacional e DAO

Classe Pessoa:

```
5 package cadastrobd.model;
6
7 /**
8  *
9  * @author leosc
10  */
11 public class Pessoa {
12     private int id;
13     private String nome;
14     private String logradouro;
15     private String cidade;
16     private String estado;
17     private String telefone;
18     private String email;
19
20     public Pessoa(){
21     }
22
23     public Pessoa(int id, String nome, String logradouro, String cidade, String
24         this.id = id;
25         this.nome = nome;
26         this.logradouro = logradouro;
27         this.cidade = cidade;
28         this.estado = estado;
29         this.telefone = telefone;
30         this.email = email;
31     }
32
33     public int getId() {
34         return id;
35     }
36 }
```

```
37 public void setId(int id) {
38     this.id = id;
39 }
40
41 public String getNome() {
42     return nome;
43 }
44
45 public void setNome(String nome) {
46     this.nome = nome;
47 }
48
49 public String getLogradouro() {
50     return logradouro;
51 }
52
53 public void setLogradouro(String logradouro) {
54     this.logradouro = logradouro;
55 }
56
57 public String getCidade() {
58     return cidade;
59 }
60
61 public void setCidade(String cidade) {
62     this.cidade = cidade;
63 }
64
65 public String getEstado() {
66     return estado;
67 }
68
69 public void setEstado(String estado) {
70     this.estado = estado;
71 }
72
73 public String getTelefone() {
74     return telefone;
75 }
76
77 public void setTelefone(String telefone) {
78     this.telefone = telefone;
79 }
80
81 public String getEmail() {
82     return email;
83 }
84
85 public void setEmail(String email) {
86     this.email = email;
87 }
88
89 public void exibir() {
90     System.out.println("ID: " + id);
91     System.out.println("Nome: " + nome);
92     System.out.println("Logradouro: " + logradouro);
93     System.out.println("Cidade: " + cidade);
94     System.out.println("Estado: " + estado);
95     System.out.println("Telefone: " + telefone);
96     System.out.println("Email: " + email);
97 }
98 }
```

Classe Pessoa fisica:

```
5 package cadastrobd.model;
6
7 /**
8  *
9  * @author leosc
10  */
11 public class PessoaFisica extends Pessoa {
12     private String cpf;
13
14     public PessoaFisica(){
15     }
16
17     public PessoaFisica(int id, String nome, String logradouro, String cidade, S
18         super(id, nome, logradouro, cidade, estado, telefone, email);
19         this.cpf = cpf;
20     }
21
22     public String getCpf() {
23         return cpf;
24     }
25
26     public void setCpf(String cpf) {
27         this.cpf = cpf;
28     }
29
30     @Override
31     public void exibir() {
32         super.exibir();
33         System.out.println("CPF: " + cpf);
34     }
35 }
```

Classe Pessoa juridica:

```
11 public class PessoaJuridica extends Pessoa {
12     private String cnpj;
13
14     public PessoaJuridica() {
15     }
16
17     public PessoaJuridica(int id, String nome, String logradouro, String cidade,
18         super(id, nome, logradouro, cidade, estado, telefone, email);
19         this.cnpj = cnpj;
20     }
21
22     public String getCnpj() {
23         return cnpj;
24     }
25
26     public void setCnpj(String cnpj) {
27         this.cnpj = cnpj;
28     }
29
30     @Override
31     public void exibir() {
32         super.exibir();
33         System.out.println("CNPJ: " + cnpj);
34     }
35 }
```

Classe ConectorBD:

```
17 public class ConectorBD {
18     private String url;
19     private String usuario;
20     private String senha;
21
22     public ConectorBD(String url, String usuario, String senha) {
23         this.url = url;
24         this.usuario = usuario;
25         this.senha = senha;
26     }
27
28     public Connection getConnection() throws SQLException {
29         return DriverManager.getConnection(url, user:usuario, password: senha);
30     }
31
32     public PreparedStatement getPrepared(String sql) throws SQLException {
33         Connection conn = getConnection();
34         return conn.prepareStatement(string: sql);
35     }
36
37     public ResultSet getSelect(String sql) throws SQLException {
38         Connection conn = getConnection();
39         PreparedStatement stmt = conn.prepareStatement(string: sql);
40         return stmt.executeQuery();
41     }
42
43     public void close(Connection conn) {
44         if (conn != null) {
45             try {
46                 conn.close();
47             } catch (SQLException e) {
48                 e.printStackTrace();
49             }
50         }
51     }
52
53     public void close(Statement stmt) {
54         if (stmt != null) {
55             try {
56                 stmt.close();
57             } catch (SQLException e) {
58                 e.printStackTrace();
59             }
60         }
61     }
62
63     public void close(ResultSet rs) {
64         if (rs != null) {
65             try {
66                 rs.close();
67             } catch (SQLException e) {
68                 e.printStackTrace();
69             }
70         }
71     }
72
73     public void close(Connection conn, Statement stmt, ResultSet rs) {
74         close(rs);
75         close(stmt);
76         close(conn);
77     }
78 }
```

Classe SequenceManager:

```
15 public class SequenceManager {
16     private ConectorBD conector;
17
18     public SequenceManager(ConectorBD conector) {
19         this.conector = conector;
20     }
21
22     public int getValue(String sequenceName) {
23         int nextValue = -1;
24         Connection conn = null;
25         PreparedStatement stmt = null;
26         ResultSet rs = null;
27
28         try {
29             conn = conector.getConnection();
30             String query = "SELECT NEXT VALUE FOR " + sequenceName + " AS NextVa";
31             stmt = conn.prepareStatement(string: query);
32             rs = stmt.executeQuery();
33
34             if (rs.next()) {
35                 nextValue = rs.getInt(string: "NextValue");
36             }
37         } catch (SQLException e) {
38             e.printStackTrace();
39         } finally {
40             conector.close(rs);
41             conector.close(stmt);
42             conector.close(conn);
43         }
44
45         return nextValue;
46     }
47 }
```

Classe PessoaFisicaDAO:

```
19 public class PessoaFisicaDAO {
20     private ConectorBD conector;
21
22     public PessoaFisicaDAO(ConectorBD conector) {
23         this.conector = conector;
24     }
25
26     public PessoaFisicaDAO(){
27     }
28
29     public PessoaFisica getPessoa(int id) {
30         Connection conn = null;
31         PreparedStatement stmt = null;
32         ResultSet rs = null;
33         PessoaFisica pessoa = null;
34
35         try {
36             conn = conector.getConnection();
37             String query = "SELECT * FROM Pessoas P JOIN PessoasFisicas PF ON P.";
38             stmt = conn.prepareStatement(string: query);
39             stmt.setInt(i: 1, i1: id);
40             rs = stmt.executeQuery();
```

```

42         if (rs.next()) {
43             pessoa = new PessoaFisica();
44             pessoa.setId(id: rs.getInt(string: "idPessoa"));
45             pessoa.setNome(nome: rs.getString(string: "nome"));
46             pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
47             pessoa.setCidade(cidade: rs.getString(string: "cidade"));
48             pessoa.setEstado(estado: rs.getString(string: "estado"));
49             pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
50             pessoa.setEmail(email: rs.getString(string: "email"));
51             pessoa.setCpf(cpf: rs.getString(string: "cpf"));
52         }
53     } catch (SQLException e) {
54         e.printStackTrace();
55     } finally {
56         conector.close(rs);
57         conector.close(stmt);
58         conector.close(conn);
59     }
60
61     return pessoa;
62 }
63
64 public List<PessoaFisica> getPessoas() {
65     List<PessoaFisica> pessoas = new ArrayList<>();
66     Connection conn = null;
67     PreparedStatement stmt = null;
68     ResultSet rs = null;
69
70     try {
71         conn = conector.getConnection();
72         String query = "SELECT * FROM Pessoas P JOIN PessoasFisicas PF ON P.";
73         stmt = conn.prepareStatement(string: query);
74         rs = stmt.executeQuery();
75
76         while (rs.next()) {
77             PessoaFisica pessoa = new PessoaFisica();
78             pessoa.setId(id: rs.getInt(string: "idPessoa"));
79             pessoa.setNome(nome: rs.getString(string: "nome"));
80             pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
81             pessoa.setCidade(cidade: rs.getString(string: "cidade"));
82             pessoa.setEstado(estado: rs.getString(string: "estado"));
83             pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
84             pessoa.setEmail(email: rs.getString(string: "email"));
85             pessoa.setCpf(cpf: rs.getString(string: "cpf"));
86             pessoas.add(e: pessoa);
87         }
88     } catch (SQLException e) {
89     } finally {
90         conector.close(rs);
91         conector.close(stmt);
92         conector.close(conn);
93     }
94
95     return pessoas;
96 }
97
98 public boolean incluir(PessoaFisica pessoa) {
99     Connection conn = null;
100     PreparedStatement stmtPessoa = null;
101     PreparedStatement stmtPessoaFisica = null;
102     boolean sucesso = false;
103

```

```

104     try {
105         conn = conector.getConnection();
106         conn.setAutoCommit(bln: false);
107
108         // Inserir na tabela Pessoas
109         String queryPessoa = "INSERT INTO Pessoas ("
110             + "idPessoa, nome, logradouro, cidade, estado, telefone,"
111             + " email) VALUES (?, ?, ?, ?, ?, ?, ?)";
112         stmtPessoa = conn.prepareStatement(string: queryPessoa,
113             i: PreparedStatement.RETURN_GENERATED_KEYS);
114         stmtPessoa.setInt(i: 1, i1: pessoa.getId());
115         stmtPessoa.setString(i: 2, string: pessoa.getNome());
116         stmtPessoa.setString(i: 3, string: pessoa.getLogradouro());
117         stmtPessoa.setString(i: 4, string: pessoa.getCidade());
118         stmtPessoa.setString(i: 5, string: pessoa.getEstado());
119         stmtPessoa.setString(i: 6, string: pessoa.getTelefone());
120         stmtPessoa.setString(i: 7, string: pessoa.getEmail());
121         stmtPessoa.executeUpdate();
122
123         if (pessoa.getId() != -1) {
124             // Inserir na tabela PessoasFisicas
125             String queryPessoaFisica = "INSERT INTO PessoasFisicas ("
126                 + "idPFisica, cpf) VALUES (?, ?)";
127             stmtPessoaFisica = conn.prepareStatement(string: queryPessoaFisica);
128             stmtPessoaFisica.setInt(i: 1, i1: pessoa.getId());
129             stmtPessoaFisica.setString(i: 2, string: pessoa.getCpf());
130             stmtPessoaFisica.executeUpdate();
131             conn.commit();
132             sucesso = true;
133         } else {
134             conn.rollback();
135         }
136     } catch (SQLException e) {
137         if (conn != null) {
138             try {
139                 conn.rollback();
140             } catch (SQLException ex) {
141                 ex.printStackTrace();
142             }
143         }
144     } finally {
145         conector.close(stmt: stmtPessoaFisica);
146         conector.close(stmt: stmtPessoa);
147         conector.close(conn);
148     }
149
150     return sucesso;
151 }
152
153 public boolean alterar(PessoaFisica pessoa) {
154     Connection conn = null;
155     PreparedStatement stmtPessoa = null;
156     PreparedStatement stmtPessoaFisica = null;
157     boolean sucesso = false;
158
159     try {
160         conn = conector.getConnection();
161         conn.setAutoCommit(bln: false);
162
163         // Atualizar tabela Pessoas
164         String queryPessoa = "UPDATE Pessoas SET nome = ?, logradouro = ?,"
165             + " cidade = ?, estado = ?, telefone = ?,"
166             + " email = ? WHERE idPessoa = ?";

```

```

167 stmtPessoa = conn.prepareStatement(string: queryPessoa);
168 stmtPessoa.setString(i: 1, string: pessoa.getNome());
169 stmtPessoa.setString(i: 2, string: pessoa.getLogradouro());
170 stmtPessoa.setString(i: 3, string: pessoa.getCidade());
171 stmtPessoa.setString(i: 4, string: pessoa.getEstado());
172 stmtPessoa.setString(i: 5, string: pessoa.getTelefone());
173 stmtPessoa.setString(i: 6, string: pessoa.getEmail());
174 stmtPessoa.setInt(i: 7, i1: pessoa.getId());
175 stmtPessoa.executeUpdate();
176
177 // Atualizar tabela PessoasFisicas
178 String queryPessoaFisica = "UPDATE PessoasFisicas SET cpf = ? WHERE
179 stmtPessoaFisica = conn.prepareStatement(string: queryPessoaFisica);
180 stmtPessoaFisica.setString(i: 1, string: pessoa.getCpf());
181 stmtPessoaFisica.setInt(i: 2, i1: pessoa.getId());
182 stmtPessoaFisica.executeUpdate();
183
184 conn.commit();
185 sucesso = true;
186 } catch (SQLException e) {
187     e.printStackTrace();
188     if (conn != null) {
189         try {
190             conn.rollback();
191         } catch (SQLException ex) {
192             ex.printStackTrace();
193         }
194     }
195 } finally {
196     conector.close(stmt: stmtPessoaFisica);
197     conector.close(stmt: stmtPessoa);
198     conector.close(conn);
199 }
200
201 return sucesso;
202 }
203
204 public boolean excluir(int id) {
205     Connection conn = null;
206     PreparedStatement stmt = null;
207     boolean sucesso = false;
208
209     try {
210         conn = conector.getConnection();
211         conn.setAutoCommit(bln: false);
212
213         // Excluir da tabela PessoasFisicas
214         String queryPessoaFisica = "DELETE FROM PessoasFisicas WHERE idPFisi
215         stmt = conn.prepareStatement(string: queryPessoaFisica);
216         stmt.setInt(i: 1, i1: id);
217         stmt.executeUpdate();
218
219         // Excluir da tabela Pessoas
220         String queryPessoa = "DELETE FROM Pessoas WHERE idPessoa = ?";
221         stmt = conn.prepareStatement(string: queryPessoa);
222         stmt.setInt(i: 1, i1: id);
223         stmt.executeUpdate();
224
225         conn.commit();
226         sucesso = true;
227
228     } catch (SQLException e) {
229         e.printStackTrace();

```



```

230         if (conn != null) {
231             try {
232                 conn.rollback();
233             } catch (SQLException ex) {
234                 ex.printStackTrace();
235             }
236         }
237     } finally {
238         conector.close(stmt);
239         conector.close(conn);
240         if (sucesso) {
241             System.out.println(x: "Pessoa excluída com sucesso!");
242         } else {
243             System.out.println(x: "Erro ao excluir pessoa.");
244         }
245     }
246
247     return sucesso;
248 }
249 }

```

Classe PessoaJuridicaDAO:

```

19 public class PessoaJuridicaDAO {
20     private ConectorBD conector;
21
22
23     public PessoaJuridicaDAO(){
24     }
25
26     public PessoaJuridicaDAO(ConectorBD conector) {
27         this.conector = conector;
28     }
29
30
31     public PessoaJuridica getPessoa(int id) {
32         String sql = "SELECT * FROM Pessoas P INNER JOIN PessoasJuridicas PJ ON"
33             + " P.idPessoa = PJ.idPJuridica WHERE P.idPessoa = ?";
34         try (Connection conn = conector.getConnection();
35             PreparedStatement stmt = conn.prepareStatement(string: sql)) {
36             stmt.setInt(i: 1, i1: id);
37             try (ResultSet rs = stmt.executeQuery()) {
38                 if (rs.next()) {
39                     PessoaJuridica pessoaJuridica =
40                         new PessoaJuridica(id: rs.getInt(string: "idPessoa"),
41                             nome: rs.getString(string: "nome"),
42                             logradouro: rs.getString(string: "logradouro"),
43                             cidade: rs.getString(string: "cidade"),
44                             estado: rs.getString(string: "estado"),
45                             telefone: rs.getString(string: "telefone"),
46                             email: rs.getString(string: "email"),
47                             cnpj: rs.getString(string: "cnpj"));
48                     return pessoaJuridica;
49                 }
50             }
51         } catch (SQLException e) {
52         }
53         return null;
54     }
55 }

```

```

56 public List<PessoaJuridica> getPessoas() {
57     List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
58     String sql = "SELECT * FROM Pessoas P INNER JOIN PessoasJuridicas PJ ON P.idPessoa = PJ.idPessoa";
59     try (Connection conn = conector.getConnection();
60         PreparedStatement stmt = conn.prepareStatement(string: sql);
61         ResultSet rs = stmt.executeQuery()) {
62         while (rs.next()) {
63             PessoaJuridica pessoaJuridica =
64                 new PessoaJuridica(id: rs.getInt(string: "idPessoa"),
65                                     nome: rs.getString(string: "nome"),
66                                     logradouro: rs.getString(string: "logradouro"),
67                                     cidade: rs.getString(string: "cidade"),
68                                     estado: rs.getString(string: "estado"),
69                                     telefone: rs.getString(string: "telefone"),
70                                     email: rs.getString(string: "email"),
71                                     cnpj: rs.getString(string: "cnpj"));
72             pessoasJuridicas.add(e: pessoaJuridica);
73         }
74     } catch (SQLException e) {
75         return null;
76     }
77     return pessoasJuridicas;
78 }
79
80 public boolean incluir(PessoaJuridica pessoaJuridica) {
81     String sqlPessoa = "INSERT INTO Pessoas (idPessoa, nome, logradouro,"
82         + " cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
83     String sqlPessoaJuridica = "INSERT INTO PessoasJuridicas (idPessoaJuridica,"
84         + " cnpj) VALUES (?, ?)";
85
86     try (Connection conn = conector.getConnection();
87         PreparedStatement stmtPessoa = conn.prepareStatement(string: sqlPessoa);
88         PreparedStatement stmtPessoaJuridica = conn.prepareStatement(string: sqlPessoaJuridica)) {
89
90         stmtPessoa.setInt(i: 1, i1: pessoaJuridica.getId());
91         stmtPessoa.setString(i: 2, string: pessoaJuridica.getNome());
92         stmtPessoa.setString(i: 3, string: pessoaJuridica.getLogradouro());
93         stmtPessoa.setString(i: 4, string: pessoaJuridica.getCidade());
94         stmtPessoa.setString(i: 5, string: pessoaJuridica.getEstado());
95         stmtPessoa.setString(i: 6, string: pessoaJuridica.getTelefone());
96         stmtPessoa.setString(i: 7, string: pessoaJuridica.getEmail());
97         stmtPessoa.executeUpdate();
98
99         stmtPessoaJuridica.setInt(i: 1, i1: pessoaJuridica.getId());
100        stmtPessoaJuridica.setString(i: 2, string: pessoaJuridica.getCnpj());
101        stmtPessoaJuridica.executeUpdate();
102
103        return true;
104    } catch (SQLException e) {
105        return false;
106    }
107 }
108
109 public boolean alterar(PessoaJuridica pessoaJuridica) {
110     String sqlPessoa = "UPDATE Pessoas SET nome=?, logradouro=?, cidade=?, "
111         + " estado=?, telefone=?, email=? WHERE idPessoa=?";
112     String sqlPessoaJuridica = "UPDATE PessoasJuridicas SET cnpj=? WHERE idPessoaJuridica=?";
113
114     try (Connection conn = conector.getConnection();
115         PreparedStatement stmtPessoa = conn.prepareStatement(string: sqlPessoa);
116         PreparedStatement stmtPessoaJuridica = conn.prepareStatement(string: sqlPessoaJuridica)) {
117

```

```

118         stmtPessoa.setString(i: 1, string: pessoaJuridica.getNome());
119         stmtPessoa.setString(i: 2, string: pessoaJuridica.getLogradouro());
120         stmtPessoa.setString(i: 3, string: pessoaJuridica.getCidade());
121         stmtPessoa.setString(i: 4, string: pessoaJuridica.getEstado());
122         stmtPessoa.setString(i: 5, string: pessoaJuridica.getTelefone());
123         stmtPessoa.setString(i: 6, string: pessoaJuridica.getEmail());
124         stmtPessoa.setInt(i: 7, i1: pessoaJuridica.getId());
125         stmtPessoa.executeUpdate();
126
127         stmtPessoaJuridica.setString(i: 1, string: pessoaJuridica.getCnpj());
128         stmtPessoaJuridica.setInt(i: 2, i1: pessoaJuridica.getId());
129         stmtPessoaJuridica.executeUpdate();
130
131         return true;
132     } catch (SQLException e) {
133         return false;
134     }
135 }
136
137 public boolean excluir(int id) {
138     String sqlPessoa = "DELETE FROM Pessoas WHERE idPessoa=?";
139     String sqlPessoaJuridica = "DELETE FROM PessoasJuridicas WHERE idPJuridica=?";
140     try (Connection conn = conector.getConnection();
141         PreparedStatement stmtPessoaJuridica = conn.prepareStatement(string: sqlPessoaJuridica);
142         PreparedStatement stmtPessoa = conn.prepareStatement(string: sqlPessoa);) {
143
144         // Excluir da tabela PessoasJuridicas primeiro
145         stmtPessoaJuridica.setInt(i: 1, i1: id);
146         stmtPessoaJuridica.executeUpdate();
147
148         // Em seguida, excluir da tabela Pessoas
149         stmtPessoa.setInt(i: 1, i1: id);
150         stmtPessoa.executeUpdate();
151
152         return true;
153     } catch (SQLException e) {
154         return false;
155     }
156 }

```

CadastroBDTeste:

```

18 public class CadastroBDTeste {
19
20     public static void main(String[] args) {
21         ConectorBD conector = new ConectorBD
22             (url: "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true",
23              usuario: "loja",
24              senha: "loja");
25
26         PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conector);
27         PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conector);
28         PessoaFisica pessoaFisica = new PessoaFisica();
29         PessoaJuridica pessoaJuridica = new PessoaJuridica();
30
31         //Instanciar uma pessoa física e persistir no banco de dados.
32         pessoaFisica.setId(id: 7);
33         pessoaFisica.setNome(nome: "João");
34         pessoaFisica.setLogradouro(logradouro: "Rua 12, casa 3, Quitanda");
35         pessoaFisica.setCidade(cidade: "Riacho do Sul");
36         pessoaFisica.setEstado(estados: "PA");
37         pessoaFisica.setTelefone(telefone: "1111-1111");
38         pessoaFisica.setEmail(email: "joao@riacho.com");

```

```

39     pessoaFisica.setCpf(cpf: "11111111111");
40
41     pessoaFisica.exibir();
42
43     pessoaFisicaDAO.incluir(pessoa: pessoaFisica);
44     System.out.println(x: "Pessoa Fisica incluída");
45
46     pessoaJuridica.setId(id: 8);
47     pessoaJuridica.setNome(nome: "JJC");
48     pessoaJuridica.setLogradouro(logradouro: "Rua 11, Centro");
49     pessoaJuridica.setCidade(cidade: "Riacho do Norte");
50     pessoaJuridica.setEstado(estado: "PA");
51     pessoaJuridica.setTelefone(telefone: "1212-1212");
52     pessoaJuridica.setEmail(email: "jjc@riacho.com");
53     pessoaJuridica.setCnpj(cnpj: "22222222222");
54
55     pessoaJuridica.exibir();
56
57     pessoaJuridicaDAO.incluir(pessoaJuridica);
58     System.out.println(x: "Pessoa Juridica incluída");
59
60     //Alterar os dados da pessoa física no banco.
61     pessoaFisica.setId(id: 7);
62     pessoaFisica.setNome(nome: "Paulo André");
63     pessoaFisica.setLogradouro(logradouro: "Rua 20");
64     pessoaFisica.setCidade(cidade: "Riacho do Norte");
65     pessoaFisica.setEstado(estado: "SP");
66     pessoaFisica.setTelefone(telefone: "7777-8888");
67     pessoaFisica.setEmail(email: "pauloandre@riacho.com");
68     pessoaFisica.setCpf(cpf: "99999999999");
69
70     pessoaFisicaDAO.alterar(pessoa: pessoaFisica);
71     System.out.println(x: "Pessoa Fisica alterada");
72
73     pessoaJuridica.setId(id: 8);
74     pessoaJuridica.setNome(nome: "PA-eletric");
75     pessoaJuridica.setLogradouro(logradouro: "Rua 70");
76     pessoaJuridica.setCidade(cidade: "Riacho");
77     pessoaJuridica.setEstado(estado: "TO");
78     pessoaJuridica.setTelefone(telefone: "8888-1111");
79     pessoaJuridica.setEmail(email: "pa@riacho.com");
80     pessoaJuridica.setCnpj(cnpj: "11111111111");
81
82     pessoaJuridicaDAO.alterar(pessoaJuridica);
83     System.out.println(x: "Pessoa Juridica alterada");
84
85     //Consultar todas as pessoas físicas do banco de dados e listar no console
86     System.out.println(x: "Exibir todas Pessoa Fisicas");
87     List<PessoaFisica> pessoasF = pessoaFisicaDAO.getPessoas();
88     for (PessoaFisica pessoa : pessoasF) {
89         pessoa.exibir();
90     }
91
92     System.out.println(x: "Exibir todas Pessoa Juridicas");
93     List<PessoaJuridica> pessoasJ = pessoaJuridicaDAO.getPessoas();
94     for (PessoaJuridica pessoa : pessoasJ) {
95         pessoa.exibir();
96     }
97
98     //Excluir a pessoa física criada anteriormente no banco.
99     pessoaFisicaDAO.excluir(id: 7);
100    System.out.println(x: "Pessoa fisica excluída!!");

```

```

101
102     pessoaJuridicaDAO.excluir(id: 8 );
103     System.out.println(x: "Pessoa Juridica excluida!!");
104
105     System.out.println(x: "programa encerrado!!");
106 }
107 }
108

```

Resultados exibidos no console:

```

Output - CadastroBD (run)
ID: 1
Nome: Jo o
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111
Pessoa Fisica incluida
ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 22222222222
Pessoa Juridica incluida
Pessoa Fisica alterada
Pessoa Juridica alterada
Exibir todas Pessoas Fisicas
ID: 1
Nome: Paulo Andr 
Logradouro: Rua 20
Cidade: Riacho do Norte
Estado: SP
Telefone: 7777-8888
Email: pauloandre@riacho.com
CPF: 99999999999
ID: 7
Nome: Paulo Andr 
Logradouro: Rua 20
Cidade: Riacho do Norte
Estado: SP
Telefone: 7777-8888
Email: pauloandre@riacho.com
CPF: 99999999999
Exibir todas Pessoas Juridicas
ID: 2
Nome: PA-eletric
Logradouro: Rua 70
Cidade: Riacho
Estado: TO

```

```
Telefone: 8888-1111
Email: pa@riacho.com
CNPJ: 1111111111
Pessoa excluída com sucesso!
Pessoa física excluída!!
Pessoa Jurídica excluída!!
programa encerrado!!
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Os componentes de middleware como o JDBC desempenham um papel fundamental na facilitação da comunicação e interação entre aplicativos e sistemas de gerenciamento de banco de dados. Eles proporcionam abstração, portabilidade, segurança e eficiência, contribuindo para a criação de aplicativos robustos, flexíveis e de alto desempenho.

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

O PreparedStatement é uma escolha mais segura e eficiente para manipulação de dados em bancos de dados, especialmente quando envolve a inserção de valores dinâmicos em consultas SQL. O Statement, embora simples, pode expor o aplicativo a vulnerabilidades de segurança se não for usado com cuidado.

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

O padrão DAO melhora a manutenibilidade do software ao promover a modularidade, a separação de responsabilidades, a flexibilidade e a testabilidade. Ele ajuda a criar um código mais organizado, menos propenso a erros e mais fácil de manter e evoluir ao longo do tempo.

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

A herança é refletida no banco de dados por meio da criação de tabelas separadas para cada classe herdeira PessoaFísica e PessoaJurídica, que compartilham campos comuns com a classe base Pessoa. Isso permite armazenar dados específicos de cada tipo de pessoa enquanto mantém uma estrutura organizada e eficiente no banco de dados. As tabelas separadas mantêm a integridade dos dados e possibilitam consultas específicas para cada tipo de pessoa.

2º Procedimento – Alimentando a Base

Classe SetPessoas, criada para reutilizar linhas de código:

```
16 public class SetPessoas {
17
18     public static void setDadosGerais(Pessoa pessoa, Scanner scanner) {
19         System.out.println("Digite o nome:");
20         String nome = scanner.nextLine();
21
22         System.out.println("Digite a Rua:");
23         String logradouro = scanner.nextLine();
24
25         System.out.println("Digite a cidade:");
26         String cidade = scanner.nextLine();
27
28         System.out.println("Digite o estado:");
29         String estado = scanner.nextLine();
30
31         System.out.println("Digite o telefone:");
32         String telefone = scanner.nextLine();
33
34         System.out.println("Digite o E-mail:");
35         String email = scanner.nextLine();
36
37         pessoa.setNome(nome);
38         pessoa.setLogradouro(logradouro);
39         pessoa.setCidade(cidade);
40         pessoa.setEstado(estado);
41         pessoa.setTelefone(telefone);
42         pessoa.setEmail(email);
43     }
44
45     public static void setDadosPessoaFisica(PessoaFisica pessoaFisica, Scanner scanner) {
46         setDadosGerais(pessoa: pessoaFisica, scanner);
47
48         System.out.println("Digite o CPF:");
49         String cpf = scanner.nextLine();
50         pessoaFisica.setCpf(cpf);
51     }
52
53     public static void setDadosPessoaJuridica(PessoaJuridica pessoaJuridica, Scanner scanner) {
54         setDadosGerais(pessoa: pessoaJuridica, scanner);
55
56         System.out.println("Digite o CNPJ:");
57         String cnpj = scanner.nextLine();
58         pessoaJuridica.setCnpj(cnpj);
59     }
60
61     public static void juridicas(PessoaJuridica pessoaJuridica, Scanner scanner) {
62         SetPessoas.setDadosPessoaJuridica(pessoaJuridica, scanner);
63     }
64
65     public static void fisicas(PessoaFisica pessoaFisica, Scanner scanner) {
66         SetPessoas.setDadosPessoaFisica(pessoaFisica, scanner);
67     }
68
69 }
70 }
```


Classe Main:

```
21 public class CadastroBD {
22     /**
23      * @param args the command line arguments
24      */
25     public static void main(String[] args) {
26         ConectorBD conector = new ConectorBD
27             (url: "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true
28              usuario: "loja",
29              senha: "loja");
30         // TODO code application logic here
31         SequenceManager sequenceManager = new SequenceManager(conector);
32         Scanner scanner = new Scanner(System.in);
33
34         PessoaFisica pessoaFisica = new PessoaFisica();
35         PessoaJuridica pessoaJuridica = new PessoaJuridica();
36
37         PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conector);
38         PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conector);
39         boolean menu = true;
40         boolean caso;
41         int numMenu;
42         String tipoPessoa;
43         int id;
44
45         while(menu == true){
46             System.out.println(x: "=====");
47             System.out.println(x: "1 - Incluir pessoa");
48             System.out.println(x: "2 - Alterar pessoa ");
49             System.out.println(x: "3 - Excluir pessoa ");
50             System.out.println(x: "4 - Exibir pelo ID ");
51             System.out.println(x: "5 - Exibir todos ");
52             System.out.println(x: "0 - Finalizar Programa");
53             System.out.println(x: "=====");
54
55             System.out.println(x: "Digite o numero desejado: ");
56             numMenu = scanner.nextInt();
57             scanner.nextLine();
58
59             switch(numMenu){
60                 case 1:
61                     caso = true;
62                     while(caso == true){
63                         System.out.println(x: "Selecionado Incluir pessoa");
64                         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Jur
65                         tipoPessoa = scanner.nextLine().toUpperCase();
66
67                         switch(tipoPessoa){
68                             case "F" -> {
69                                 id = sequenceManager.getValue(sequenceName: "seqId"
70                                 pessoaFisica.setId(id);
71                                 SetPessoas.fisicas(pessoaFisica, scanner);
72                                 pessoaFisicaDAO.incluir(pessoa: pessoaFisica);
73                                 caso = false;
74                                 break;
75                             }
76
77                             case "J" -> {
78                                 id = sequenceManager.getValue(sequenceName: "seqId"
79                                 pessoaJuridica.setId(id);
80                                 SetPessoas.juridicas(pessoaJuridica, scanner);
81                                 pessoaJuridicaDAO.incluir(pessoaJuridica);

```



```

83         caso = false;
84         break;
85     }
86
87     default -> {
88         System.out.println(x: "comando incorreto");
89         caso = true;
90     }
91 }
92
93 }
94 break;
95
96 case 2:
97     caso = true;
98     while(caso == true){
99         System.out.println(x: "Selecione alterar pessoa");
100        System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
101        tipoPessoa = scanner.nextLine().toUpperCase();
102
103        switch(tipoPessoa){
104            case "F" -> {
105                System.out.println(x: "Digite o ID:");
106                id = scanner.nextInt();
107                scanner.nextLine();
108                PessoaFisica pessoaF = pessoaFisicaDAO.getPessoa(id);
109                pessoaF.exibir();
110                pessoaFisica.setId(id);
111                SetPessoas.fisicas(pessoaFisica, scanner);
112                pessoaFisicaDAO.alterar(pessoa: pessoaFisica);
113                System.out.println(x: "Pessoa alterada com sucesso!");
114                caso = false;
115                break;
116            }
117
118            case "J" -> {
119                System.out.println(x: "Digite o ID:");
120                id = scanner.nextInt();
121                scanner.nextLine();
122                PessoaJuridica pessoaJ = pessoaJuridicaDAO.getPessoa(id);
123                pessoaJ.exibir();
124                pessoaJuridica.setId(id);
125                SetPessoas.juridicas(pessoaJuridica, scanner);
126                pessoaJuridicaDAO.alterar(pessoaJuridica);
127                System.out.println(x: "Pessoa alterada com sucesso!");
128                caso = false;
129                break;
130            }
131
132            default -> {
133                System.out.println(x: "comando incorreto");
134                caso = true;
135            }
136        }
137    }
138    break;
139
140 case 3:
141     caso = true;
142     while(caso == true){
143         System.out.println(x: "Selecione excluir pessoa");
144         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
145         tipoPessoa = scanner.nextLine().toUpperCase();
146
147         switch(tipoPessoa){
148             case "F" -> {
149                 System.out.println(x: "Digite o ID:");
150                 id = scanner.nextInt();
151                 scanner.nextLine();
152                 PessoaFisica pessoaF = pessoaFisicaDAO.getPessoa(id);
153                 pessoaF.exibir();
154                 pessoaFisicaDAO.excluir(id);
155                 caso = false;
156                 break;
157             }
158
159             case "J" -> {
160                 System.out.println(x: "Digite o ID:");
161                 id = scanner.nextInt();
162                 scanner.nextLine();
163                 PessoaJuridica pessoaj = pessoaJuridicaDAO.getPessoa(id);
164                 pessoaj.exibir();
165                 pessoaJuridicaDAO.excluir(id);

```

```

166         caso = false;
167         break;
168     }
169
170     default -> {
171         System.out.println(x: "comando incorreto");
172         caso = true;
173     }
174 }
175
176 }
177
178 break;
179
180 case 4:
181     caso = true;
182     while(caso == true){
183         System.out.println(x: "Selecione para exibir pessoa");
184         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
185         tipoPessoa = scanner.nextLine().toUpperCase();
186
187         switch(tipoPessoa){
188             case "F" -> {
189                 System.out.println(x: "Digite o ID:");
190                 id = scanner.nextInt();
191                 scanner.nextLine();
192                 PessoaFisica pessoaF = pessoaFisicaDAO.getPessoa(id);
193                 pessoaF.exibir();
194                 caso = false;
195             }
196
197             case "J" -> {
198                 System.out.println(x: "Digite o ID:");
199                 id = scanner.nextInt();
200                 scanner.nextLine();
201                 PessoaJuridica pessoaJ = pessoaJuridicaDAO.getPessoa(id);
202                 pessoaJ.exibir();
203                 caso = false;
204             }
205
206             default -> {
207                 System.out.println(x: "comando incorreto");
208                 caso = true;
209             }
210         }
211     }
212 }
213
214 break;
215
216 case 5:
217     caso = true;
218     while(caso == true){
219         System.out.println(x: "Selecione para exibir todos");
220         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
221         tipoPessoa = scanner.nextLine().toUpperCase();
222
223         switch(tipoPessoa){
224             case "F" -> {
225                 List<PessoaFisica> pessoasF = pessoaFisicaDAO.getPessoas();
226                 for (PessoaFisica pessoa : pessoasF) {
227                     pessoa.exibir();
228                 }
229                 caso = false;
230                 break;
231             }
232
233             case "J" -> {
234                 List<PessoaJuridica> pessoasJ = pessoaJuridicaDAO.getPessoas();
235                 for (PessoaJuridica pessoa : pessoasJ) {
236                     pessoa.exibir();
237                 }
238                 caso = false;
239                 break;
240             }
241
242             default -> {
243                 System.out.println(x: "comando incorreto");
244                 caso = true;
245             }
246         }
247     }
248 }
249
250 break;

```

```

251
252
253
254
255
256
257
258
259
260
    case 0:
        System.out.println(x: "Programa encerrado!!");
        menu = false;
    }
}
}
}
}
}

```

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

A persistência em arquivo e a persistência em banco de dados diferem em como os dados são armazenados. A persistência em arquivo envolve gravar dados diretamente em arquivos, sendo simples e portátil, mas menos escalável e estruturada. Em contraste, a persistência em banco de dados usa sistemas gerenciados para armazenar dados, oferecendo estrutura, consultas avançadas e escalabilidade, embora com maior complexidade e dependência de um sistema de gerenciamento de banco de dados.

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

O uso de operadores lambda no Java permite uma forma mais concisa e legível de iterar e processar coleções de dados. Nas versões mais recentes do Java, os operadores lambda simplificaram a impressão dos valores contidos nas entidades, reduzindo a necessidade de escrever loops tradicionais e permitindo a definição de funções inline para manipulação de dados. Isso resulta em um código mais limpo e eficiente, aumentando a produtividade e a clareza na programação.

- 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 em Java precisam ser marcados como static porque o método main é estático e pertence à classe em si, não a uma instância dela. Isso significa que ele pode ser chamado sem a necessidade de criar um objeto da classe. Métodos não estáticos pertencem a instâncias da classe e requerem um objeto para serem invocados, o que não é adequado quando chamados diretamente do método main. Marcando os métodos como static, eles podem ser acessados diretamente sem a criação de instâncias.

Conclusão

Neste trabalho, desenvolvi um sistema de cadastro de pessoas que abrange tanto pessoas físicas quanto jurídicas. Utilizei uma abordagem orientada a objetos, criando

classes como Pessoa, PessoaFisica e PessoaJuridica, cada uma com seus respectivos atributos e métodos. Implementei operações de CRUD (criação, leitura, atualização e exclusão) através de classes DAO, garantindo a interação com o banco de dados de forma eficiente e organizada. A aplicação conta com um menu de fácil utilização, permitindo ao usuário realizar diversas ações, como inclusão, alteração, exclusão e exibição de pessoas.