



BackEnd sem banco não tem

Jonathan da Silva Araujo - 202205178111

Campus Nova Iguaçu – RJ

BackEnd sem banco não tem (RPG0016) – 9001 – 2023.2 (3º Semestre)

Objetivo da Prática

1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar Sistemas cadastrais com persistência em banco relacional.

Procedimentos Realizados

Classe Pessoa

```
1 package model;
2 import java.io.Serializable;
3
4 2 usages 2 inheritors Jonathan Araujo *
5 public class Pessoa implements Serializable{
6     2 usages
7     int id;
8     2 usages
9     String nome;
10    2 usages
11    String logradouro;
12    2 usages
13    String cidade;
14    2 usages
15    String estado;
16    2 usages
17    String telefone;
18    2 usages
19    String email;
20
21    no usages Jonathan Araujo
22    public int getId() {
23        return id;
24    }
25
26    no usages Jonathan Araujo
27    public void setId(int id) {
28        this.id = id;
29    }
30 }
```

```

21     public String getNome() {
22         return nome;
23     }
24
25     no usages new *
26     public void setNome(String nome) {
27         this.nome = nome;
28     }
29
30     no usages new *
31     public String getLogradouro() {
32         return logradouro;
33     }
34
35     no usages new *
36     public void setLogradouro(String logradouro) {this.logradouro = logradouro;}
37
38     no usages new *
39     public String getCidade() {return cidade;}
40
41     no usages new *
42     public void setCidade(String cidade) {this.cidade = cidade;}
43
44     no usages new *
45     public String getEstado() {return estado;}
46
47     no usages new *
48     public void setEstado(String estado) {this.estado = estado;}
49
50     no usages new *
51     public String getTelefone() {return telefone;}
52
53     no usages new *
54     public void setTelefone(String telefone) {this.telefone = telefone;}
55
56     public String getEmail() {return email;}
57
58     no usages new *
59     public void setEmail(String email) {this.email = email;}
60 }

```

Classe PessoaFisica

```

1  package model;
2  import java.io.Serializable;
3
4  no usages Jonathan Araujo *
5  public class PessoaFisica extends Pessoa implements Serializable{
6      2 usages
7      long cpf;
8
9      no usages Jonathan Araujo
10     public long getCpf() { return cpf; }
11
12     no usages Jonathan Araujo
13     public void setCpf(long cpf) { this.cpf = cpf; }
14 }
15

```

Classe PessoaJuridica

```
1 package model;
2 import java.io.Serializable;
3
4 no usages Jonathan Araujo *
5 public class PessoaJuridica extends Pessoa implements Serializable{
6     2 usages
7     long cnpj;
8
9
10
11 no usages Jonathan Araujo
12 public long getCnpj() { return cnpj; }
13
14 no usages Jonathan Araujo
15 public void setCnpj(long cnpj) { this.cnpj = cnpj; }
16 }
```

Classe Metodos

```
1 package model;
2
3 import java.sql.*;
4 import java.util.Scanner;
5
6 11 usages Jonathan Araujo *
7 public class Metodos {
8     2 usages new *
9     public static void restauraObjeto(String tipoPessoa, Connection connection) throws SQLException {
10         if (tipoPessoa == "pf") {
11             try {
12                 String sql = "SELECT * FROM fisica";
13                 PreparedStatement preparedStatement = connection.prepareStatement(sql);
14                 ResultSet resultSet = preparedStatement.executeQuery();
15                 while (resultSet.next()) {
16                     int id = resultSet.getInt( columnLabel: "idpessoa");
17                     String nome = resultSet.getString( columnLabel: "nome");
18                     String logradouro = resultSet.getString( columnLabel: "logradouro");
19                     String cidade = resultSet.getString( columnLabel: "cidade");
20                     String estado = resultSet.getString( columnLabel: "estado");
21                     String telefone = resultSet.getString( columnLabel: "telefone");
22                     String email = resultSet.getString( columnLabel: "email");
23                     String cpf = resultSet.getString( columnLabel: "cpf");
24
25                     System.out.println("Id: " + id);
26                     System.out.println("Nome: " + nome);
27                     System.out.println("Logradouro: " + logradouro);
28                     System.out.println("Cidade: " + cidade);
29                     System.out.println("Estado: " + estado);
30                     System.out.println("Telefone: " + telefone);
31                     System.out.println("Email: " + email);
32                 }
33             } catch (SQLException e) {
34                 e.printStackTrace();
35             }
36         }
37     }
38 }
```

```

30         System.out.println("CPF: " + cpf);
31         System.out.println("#####");
32     }
33     resultSet.close();
34     preparedStatement.close();
35 } catch (SQLException e) {
36     e.printStackTrace();
37 }
38 } else if (tipoPessoa == "pj") {
39     try {
40         String sql = "SELECT * FROM juridica";
41         PreparedStatement preparedStatement = connection.prepareStatement(sql);
42         ResultSet resultSet = preparedStatement.executeQuery();
43         while (resultSet.next()) {
44             int id = resultSet.getInt( columnLabel: "idpessoa");
45             String nome = resultSet.getString( columnLabel: "nome");
46             String logradouro = resultSet.getString( columnLabel: "logradouro");
47             String cidade = resultSet.getString( columnLabel: "cidade");
48             String estado = resultSet.getString( columnLabel: "estado");
49             String telefone = resultSet.getString( columnLabel: "telefone");
50             String email = resultSet.getString( columnLabel: "email");
51             String cnpj = resultSet.getString( columnLabel: "cnpj");
52
53             System.out.println("Id: " + id);
54             System.out.println("Nome: " + nome);
55             System.out.println("Logradouro: " + logradouro);
56             System.out.println("Cidade: " + cidade);
57             System.out.println("Estado: " + estado);
58             System.out.println("Telefone: " + telefone);
59             System.out.println("Email: " + email);
60
61             System.out.println("CNPJ: " + cnpj);
62             System.out.println("#####");
63         }
64         resultSet.close();
65         preparedStatement.close();
66     } catch (SQLException e) {
67         e.printStackTrace();
68     }
69 }
70 };
71
72 1 usage Jonathan Araujo *
73 public static int opcoesTerminal() {
74     System.out.println("
75     =====
76     1 - Incluir Pessoa
77     2 - Alterar Pessoa
78     3 - Excluir Pessoa
79     4 - Buscar pelo Id
80     5 - Exibir Todos
81     0 - Finalizar Programa
82     =====
83     "
84 );
85 Scanner input = new Scanner(System.in);
86 int opcao = input.nextInt();
87 return opcao;
88 };

```

```

88
    5 usages Jonathan Araujo
89 @ public static String tipoPessoa() {
90     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
91     Scanner tipo = new Scanner(System.in);
92     String tipoPessoa = tipo.nextLine().toLowerCase();
93
94     return tipoPessoa;
95 };
96
    2 usages new *
97 public static int obterUltimoID(Connection connection) {
98     int ultimoID = 0;
99     try {
100         String sql = "SELECT MAX(idpessoa) AS ultimo_id FROM pessoa";
101
102         Statement statement = connection.createStatement();
103         ResultSet resultSet = statement.executeQuery(sql);
104
105         if (resultSet.next()) {
106             ultimoID = resultSet.getInt("ultimo_id");
107         }
108         resultSet.close();
109         statement.close();
110     } catch (Exception e) {
111         e.printStackTrace();
112     }
113     return ultimoID;
114 };
115
116 };

```

Classe Main

```
1 import model.Metodos;
2 import java.sql.*;
3 import java.util.Scanner;
4
5 Jonathan Araujo *
6 public class Main {
7     Jonathan Araujo *
8     public static void main(String[] args) {
9         String jdbcUrl = "jdbc:postgresql://localhost:5432/postgres";
10        String username = "loja";
11        String password = "loja";
12
13        try {
14            Connection connection = DriverManager.getConnection(jdbcUrl, username, password);
15            System.out.println("Banco de dados conectado com sucesso!");
16
17            int opcao = -1;
18
19            while (opcao != 0) {
20                opcao = Metodos.opcoesTerminal();
21                if (opcao == 1) {
22                    String tipoPessoa = Metodos.tipoPessoa();
23
24                    Scanner nome = new Scanner(System.in);
25                    Scanner logradouro = new Scanner(System.in);
26                    Scanner cidade = new Scanner(System.in);
27                    Scanner estado = new Scanner(System.in);
28                    Scanner telefone = new Scanner(System.in);
29                    Scanner email = new Scanner(System.in);
30                    Scanner doc = new Scanner(System.in);
31
32                    System.out.println("Informe o nome:");
33                    String nomeRes = nome.nextLine();
34                    System.out.println("Informe o logradouro:");
35                    String logradouroRes = logradouro.nextLine();
36                    System.out.println("Informe a cidade:");
37                    String cidadeRes = cidade.nextLine();
38                    System.out.println("Informe o estado:");
39                    String estadoRes = estado.nextLine();
40                    System.out.println("Informe o telefone:");
41                    String telefoneRes = telefone.nextLine();
42                    System.out.println("Informe o email:");
43                    String emailRes = email.nextLine();
44
45                    String sql = "INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
46
47                    PreparedStatement preparedStatement = connection.prepareStatement(sql);
48                    preparedStatement.setString(1, nomeRes);
49                    preparedStatement.setString(2, logradouroRes);
50                    preparedStatement.setString(3, cidadeRes);
51                    preparedStatement.setString(4, estadoRes);
52                    preparedStatement.setString(5, telefoneRes);
53                    preparedStatement.setString(6, emailRes);
54                    int rowsAffected = preparedStatement.executeUpdate();
55
56                    if (tipoPessoa.equals("f")) {
57                        int idPessoa = Metodos.obterUltimoID(connection);
58
59                        System.out.println("Informe o CPF:");
60                        Long docRes = doc.nextLong();
```

```

100 String sqlFisica = "INSERT INTO fisica (nome, logradouro, cidade, estado, telefone, email, idPessoa, cpf) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
101 PreparedStatement1 preparedStatement1 = connection.prepareStatement(sqlFisica);
102 preparedStatement1.setString( parameterIndex: 1, nomeRes);
103 preparedStatement1.setString( parameterIndex: 2, logradouroRes);
104 preparedStatement1.setString( parameterIndex: 3, cidadeRes);
105 preparedStatement1.setString( parameterIndex: 4, estadoRes);
106 preparedStatement1.setString( parameterIndex: 5, telefoneRes);
107 preparedStatement1.setString( parameterIndex: 6, emailRes);
108 preparedStatement1.setInt( parameterIndex: 7, idPessoa);
109 preparedStatement1.setLong( parameterIndex: 8, docRes);
110 int rowsAffectedFisica = preparedStatement1.executeUpdate();
111
112 if (rowsAffected > 0 && rowsAffectedFisica > 0) {
113     System.out.println("Inserção realizada com sucesso!");
114 } else {
115     System.out.println("Nenhum dado foi inserido.");
116 }
117 } else if (tipoPessoa.equals("J")) {
118     int idPessoa = Metodos.obterUltimoID(connection);
119
120     System.out.println("Informe o CNPJ:");
121     long docRes = doc.nextLong();
122     String sqlJuridica = "INSERT INTO juridica (nome, logradouro, cidade, estado, telefone, email, idPessoa, cnpj) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
123     PreparedStatement1 preparedStatement1 = connection.prepareStatement(sqlJuridica);
124     preparedStatement1.setString( parameterIndex: 1, nomeRes);
125     preparedStatement1.setString( parameterIndex: 2, logradouroRes);
126     preparedStatement1.setString( parameterIndex: 3, cidadeRes);
127     preparedStatement1.setString( parameterIndex: 4, estadoRes);
128     preparedStatement1.setString( parameterIndex: 5, telefoneRes);
129     preparedStatement1.setString( parameterIndex: 6, emailRes);
130     preparedStatement1.setInt( parameterIndex: 7, idPessoa);
131     preparedStatement1.setLong( parameterIndex: 8, docRes);
132     int rowsAffectedJuridica = preparedStatement1.executeUpdate();
133
134     if (rowsAffected > 0 && rowsAffectedJuridica > 0) {
135         System.out.println("Inserção realizada com sucesso!");
136     } else {
137         System.out.println("Nenhum dado foi inserido.");
138     }
139 }
140 } else if (opcao == 2) {
141     String tipoPessoa = Metodos.tipoPessoa();
142
143     Scanner nome = new Scanner(System.in);
144     Scanner logradouro = new Scanner(System.in);
145     Scanner cidade = new Scanner(System.in);
146     Scanner estado = new Scanner(System.in);
147     Scanner telefone = new Scanner(System.in);
148     Scanner email = new Scanner(System.in);
149     Scanner doc = new Scanner(System.in);
150     Scanner id = new Scanner(System.in);
151
152     System.out.println("Insira o ID da pessoa que deseja alterar:");
153     int idP = id.nextInt();
154     System.out.println("Insira o nome da pessoa que deseja alterar:");
155     String novoNome = nome.nextLine();
156     System.out.println("Insira o logradouro da pessoa que deseja alterar:");
157     String novoLogradouro = logradouro.nextLine();
158     System.out.println("Insira a cidade da pessoa que deseja alterar:");
159     String novoCidade = cidade.nextLine();

```

```

120 System.out.println("Insira o estado da pessoa que deseja alterar:");
121 String novoEstado = estado.nextLine();
122 System.out.println("Insira o telefone da pessoa que deseja alterar:");
123 String novoTelefone = telefone.nextLine();
124 System.out.println("Insira o email da pessoa que deseja alterar:");
125 String novoEmail = email.nextLine();
126
127 String sql = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idpessoa = ?";
128 PreparedStatement preparedStatement = connection.prepareStatement(sql);
129 preparedStatement.setString( parameterIndex: 1, novoNome);
130 preparedStatement.setString( parameterIndex: 2, novoLogradouro);
131 preparedStatement.setString( parameterIndex: 3, novoCidade);
132 preparedStatement.setString( parameterIndex: 4, novoEstado);
133 preparedStatement.setString( parameterIndex: 5, novoTelefone);
134 preparedStatement.setString( parameterIndex: 6, novoEmail);
135 preparedStatement.setInt( parameterIndex: 7, idP);
136
137 int linhasAfetadas = preparedStatement.executeUpdate();
138
139 if (tipoPessoa.equals("f")) {
140     System.out.println("Insira o CPF atualizado:");
141     Long cpf = doc.nextLong();
142     String sqlPf = "UPDATE fisica SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ?, cpf = ? WHERE idpessoa = ?";
143     PreparedStatement preparedStatement1 = connection.prepareStatement(sqlPf);
144     preparedStatement1.setString( parameterIndex: 1, novoNome);
145     preparedStatement1.setString( parameterIndex: 2, novoLogradouro);
146     preparedStatement1.setString( parameterIndex: 3, novoCidade);
147     preparedStatement1.setString( parameterIndex: 4, novoEstado);
148     preparedStatement1.setString( parameterIndex: 5, novoTelefone);
149     preparedStatement1.setString( parameterIndex: 6, novoEmail);
150     preparedStatement1.setLong( parameterIndex: 7, cpf);
151     preparedStatement1.setInt( parameterIndex: 8, idP);
152
153     int linhasAfetadas1 = preparedStatement1.executeUpdate();
154     if (linhasAfetadas > 0 && linhasAfetadas1 > 0) {
155         System.out.println("Atualização bem-sucedida!");
156     } else {
157         System.out.println("Nenhum registro atualizado. Verifique o ID da pessoa.");
158     }
159 } else if (tipoPessoa.equals("j")) {
160     System.out.println("Insira o CNPJ atualizado:");
161     Long cnpj = doc.nextLong();
162     String sqlPf = "UPDATE juridica SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ?, cnpj = ? WHERE idpessoa = ?";
163     PreparedStatement preparedStatement1 = connection.prepareStatement(sqlPf);
164     preparedStatement1.setString( parameterIndex: 1, novoNome);
165     preparedStatement1.setString( parameterIndex: 2, novoLogradouro);
166     preparedStatement1.setString( parameterIndex: 3, novoCidade);
167     preparedStatement1.setString( parameterIndex: 4, novoEstado);
168     preparedStatement1.setString( parameterIndex: 5, novoTelefone);
169     preparedStatement1.setString( parameterIndex: 6, novoEmail);
170     preparedStatement1.setLong( parameterIndex: 7, cnpj);
171     preparedStatement1.setInt( parameterIndex: 8, idP);
172
173     int linhasAfetadas1 = preparedStatement1.executeUpdate();
174     if (linhasAfetadas > 0 && linhasAfetadas1 > 0) {
175         System.out.println("Atualização bem-sucedida!");
176     } else {
177         System.out.println("Nenhum registro atualizado. Verifique o ID da pessoa.");
178     }
179 }

```



```

180 } else if (opcao == 3) {
181     String tipoPessoa = Metodos.tipoPessoa();
182
183     Scanner id = new Scanner(System.in);
184     System.out.println("Digite o ID de quem deseja excluir:");
185     int idP = id.nextInt();
186     String sqlPessoa = "DELETE FROM pessoa WHERE idpessoa = ?";
187     PreparedStatement preparedStatement = connection.prepareStatement(sqlPessoa);
188     preparedStatement.setInt( parameterIndex: 1, idP);
189     int rowsDeletedPessoa = preparedStatement.executeUpdate();
190
191     if (tipoPessoa.equals("f")) {
192         try {
193             String sql = "DELETE FROM fisica WHERE idpessoa = ?";
194             PreparedStatement preparedStatement1 = connection.prepareStatement(sql);
195             preparedStatement1.setInt( parameterIndex: 1, idP);
196             int rowsDeleted = preparedStatement1.executeUpdate();
197             if (rowsDeleted > 0 && rowsDeletedPessoa > 0) {
198                 System.out.println("Registro deletado com sucesso!");
199             } else {
200                 System.out.println("Nenhum registro encontrado com o ID fornecido!");
201             }
202         } catch (SQLException e) {
203             e.printStackTrace();
204         }
205     } else if (tipoPessoa.equals("j")) {
206         try {
207             String sql = "DELETE FROM juridica WHERE idpessoajuridica = ?";
208             PreparedStatement preparedStatement1 = connection.prepareStatement(sql);
209             preparedStatement1.setInt( parameterIndex: 1, idP);
210             int rowsDeleted = preparedStatement1.executeUpdate();
211             if (rowsDeleted > 0) {
212                 System.out.println("Registro deletado com sucesso!");
213             } else {
214                 System.out.println("Nenhum registro encontrado com o ID fornecido!");
215             }
216         } catch (SQLException e) {
217             e.printStackTrace();
218         }
219     }
220 } else if (opcao == 4) {
221     String tipoPessoa = Metodos.tipoPessoa();
222     if (tipoPessoa.equals("f")) {
223         Scanner id = new Scanner(System.in);
224         System.out.println("Digite o ID para realizar a busca:");
225         int idPf = id.nextInt();
226         try {
227             PreparedStatement preparedStatement = null;
228             ResultSet resultSet = null;
229             String sql = "SELECT * FROM fisica WHERE idpessoafisica = ?";
230             preparedStatement = connection.prepareStatement(sql);
231             preparedStatement.setInt( parameterIndex: 1, idPf);
232             resultSet = preparedStatement.executeQuery();
233
234             if (resultSet.next()) {
235                 System.out.println("#####");
236                 System.out.println("Id: " + resultSet.getInt( columnLabel: "idpessoa"));
237                 System.out.println("Nome: " + resultSet.getString( columnLabel: "nome"));
238                 System.out.println("Logradouro: " + resultSet.getString( columnLabel: "logradouro"));
239                 System.out.println("Cidade: " + resultSet.getString( columnLabel: "cidade"));

```

```

240         System.out.println("Estado: " + resultSet.getString( columnLabel: "estado"));
241         System.out.println("Telefone: " + resultSet.getString( columnLabel: "telefone"));
242         System.out.println("Email: " + resultSet.getString( columnLabel: "email"));
243         System.out.println("CPF: " + resultSet.getLong( columnLabel: "cpf"));
244         System.out.println("#####");
245     }
246
247     } catch (SQLException e) {
248         e.printStackTrace();
249     }
250 } else if (tipoPessoa.equals("j")) {
251     Scanner id = new Scanner(System.in);
252     int idPj = id.nextInt();
253     try {
254         PreparedStatement preparedStatement = null;
255         ResultSet resultSet = null;
256         String sql = "SELECT * FROM juridica WHERE idpessoajuridica = ?";
257         preparedStatement = connection.prepareStatement(sql);
258         preparedStatement.setInt( parameterIndex: 1, idPj);
259         resultSet = preparedStatement.executeQuery();
260
261         if (resultSet.next()) {
262             System.out.println("#####");
263             System.out.println("Id: " + resultSet.getInt( columnLabel: "idpessoa"));
264             System.out.println("Nome: " + resultSet.getString( columnLabel: "nome"));
265             System.out.println("Logradouro: " + resultSet.getString( columnLabel: "logradouro"));
266             System.out.println("Cidade: " + resultSet.getString( columnLabel: "cidade"));
267             System.out.println("Estado: " + resultSet.getString( columnLabel: "estado"));
268             System.out.println("Telefone: " + resultSet.getString( columnLabel: "telefone"));
269             System.out.println("Email: " + resultSet.getString( columnLabel: "email"));
270             System.out.println("CNPJ: " + resultSet.getLong( columnLabel: "cnpj"));
271             System.out.println("#####");
272         }
273
274     } catch (SQLException e) {
275         e.printStackTrace();
276     }
277 }
278 } else if (opcao == 5) {
279     String tipoPessoa = Metodos.tipoPessoa();
280     if (tipoPessoa.equals("f")) {
281         System.out.println("Exibindo dados de Pessoa Fisica...");
282         Metodos.restauraObjeto( tipoPessoa: "pf", connection);
283     } else if (tipoPessoa.equals("j")) {
284         System.out.println("Exibindo dados de Pessoa Juridica...");
285         Metodos.restauraObjeto( tipoPessoa: "pj", connection);
286     }
287 }
288 }
289 connection.close();
290 } catch (SQLException e) {
291     e.printStackTrace();
292 }
293 };
294 };
295

```

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

Eles desempenham um papel fundamental na integração de sistemas de software, na simplificação do desenvolvimento, na melhoria da portabilidade e

no aumento da segurança e do desempenho dos aplicativos. Eles são essenciais para construir sistemas robustos e escaláveis em um ambiente empresarial moderno.

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

O `Statement` é adequado para consultas SQL dinâmicas simples, o `PreparedStatement` é a escolha preferida para a maioria dos casos devido à sua segurança aprimorada, desempenho melhorado e legibilidade do código.

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

O padrão DAO é uma abordagem valiosa para melhorar a manutenibilidade do software, pois promove a separação de responsabilidades, reutilização de código e flexibilidade, tornando mais fácil e seguro modificar e expandir um sistema ao longo do tempo.

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

Há duas abordagens comuns para refletir herança em um banco de dados estritamente relacional: herança por tabelas separadas e herança por tabela única.

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

Persistência em arquivo pode ser mais simples para casos simples ou quando a estrutura de dados é relativamente simples, enquanto persistência em banco de dados é mais apropriada para casos que requerem estruturação, desempenho, segurança e recursos avançados de gerenciamento de dados.

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

O uso simplificou a manipulação e impressão de valores em entidades ao permitir que os desenvolvedores escrevam código mais conciso e expressivo, melhorando a legibilidade e a manutenibilidade do código.

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

Os métodos static são usados para operações que não dependem de instâncias e são chamados no contexto da classe em si.

Conclusão

Esse projeto tem por finalidade realizar a criação de um aplicativo Java, o qual possa realizar o acesso ao banco de dados através de um terminal, possibilitando realizar as funções de incluir pessoa (cadastrar no bd), alterar pessoa (editar bd), excluir pessoa (deletar elemento do bd), buscar pelo id (retornar dados específicos do bd), exibir todos os dados (retornar dados do bd) e finalizar o programa (encerrar execução do script).