

# Commit

 This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.


## Refatoração em dao

Browse files

LoginDAO - Técnicas: Extract Method

PessoaDAO - Técnicas: Extract Method(5), Replace with Symbolic Constant

UfDAO - Técnicas: Extract Method, Move Method

 moisesocosta committed on Jun 21

1 parent 9d330b1    commit 58ae9a4

Showing 4 changed files with 137 additions and 425 deletions.

Whitespace

Ignore whitespace

Split

Unified

14

src/br/sistema/crud/jdbc/dao/GenericoDAO.java

|     |     |  |
|-----|-----|--|
| ... | ... | @@ -1,18 +1,14 @@  |
| 1   | 1   | package br.sistema.crud.jdbc.dao;                              |
| 2   | 2   |  |
| 3   | 3   | import java.util.List;   |
| 4   | -   |  |
| 5   | -   | import br.sistema.crud.jdbc.exception.PersistenciaExcpetion;   |
|     | 4   | + import br.sistema.crud.jdbc.exception.PersistenciaException; |
| 6   | 5   |  |
| 7   | 6   | public interface GenericoDAO<T> {                              |
| 8   | 7   |  |
| 9   | -   | void inserir(T objc) throws PersistenciaExcpetion;             |
| 10  | -   | void atualizar(T objc)throws PersistenciaExcpetion;            |
| 11  | -   |  |
| 12  | -   | void deletar(Integer idPessoa)throws PersistenciaExcpetion;    |
|     | 8   | + void salvar(T objeto) throws PersistenciaException;          |
|     | 9   | + void deletar(Integer id) throws PersistenciaException;       |
| 13  | 10  |  |
| 14  | -   | List<T> listarTodos()throws PersistenciaExcpetion;             |
|     | 11  | + List<T> listarTodos() throws PersistenciaException;          |

```

15      12
16      -      T buscarPorId(Integer idPessoa) throws PersistenciaExcpetion;
17      -
18      13      +      T buscarPorId(Integer id) throws PersistenciaException;
19      14      }

```

56 src/br/sistema/crud/jdbc/dao/LoginDAO.java

```

12      12
13      13      public class LoginDAO {
14      14
15      -      public boolean logar(LoginDTO loginDTO) throws PersistenciaExcpetion{
16      -          boolean resultado = false;
17      15      +      public boolean logar(LoginDTO loginDTO) throws PersistenciaExcpetion {
18      16      +          try {
19      17      +              Connection connection =
20      ConexaoUtil.getInstance().getConnection();
21      19      -          String sql = "select * from tb_login where nome = ? and
22      senha = ?";
23      20      -          PreparedStatement statement =
24      connection.prepareStatement(sql);
25      21      -          statement.setString(1, loginDTO.getNome());
26      22      -          statement.setString(2, loginDTO.getSenha());
27      23      -          _____
28      24      -          _____
29      25      -          ResultSet resultSet = statement.executeQuery();
30      26      -          resultado = resultSet.next();
31      18      +          String sql = "SELECT * FROM tb_login WHERE nome = ? AND
32      senha = ?";
33      19      +          PreparedStatement statement =
34      connection.prepareStatement(sql);
35      20      +          statement.setString(1, loginDTO.getNome());
36      21      +          statement.setString(2, loginDTO.getSenha());
37      22      +          _____
38      23      +          ResultSet resultSet = statement.executeQuery();
39      24      +          boolean resultado = resultSet.next();
40      25      +          _____
41      26      +          connection.close();
42      27      +          _____
43      28      +          return resultado;
44      29      +      } catch (Exception e) {
45      30      -          e.printStackTrace();
46      31      -          throw new PersistenciaExcpetion(e.getMessage(), e);
47      32      -          }
48      33      -          return resultado;
49      34      -          }
50      35      -          }
51      36      -          }
52      37      -          }
53      38      -          }
54      39      -          }

```

```

40 -
41 -     /*
42 -     @Override
43 -     public void inserir(LoginDTO objc) throws PersistenciaExcpetion {
44 -
45 -     }
46 -
47 -     @Override
48 -     public void atualizar(LoginDTO objc) throws PersistenciaExcpetion {
49 -
50 -     }
51 -
52 -     @Override
53 -     public void deletar(Integer id) throws PersistenciaExcpetion {
54 -
55 -     }
56 -
57 -     @Override
58 -     public List<LoginDTO> listarTodos() throws PersistenciaExcpetion {
59 -         return null;
60 -     }
61 -
62 -     @Override
63 -     public LoginDTO buscarPorId(Integer id) throws PersistenciaExcpetion {
64 -         // TODO Auto-generated method stub
65 -         return null;
66 -     } */
67 34 +     // Outros métodos de CRUD...
68 35
69 36 }

```

413 ■■■■ src/br/sistema/crud/jdbc/dao/PessoaDAO.java

```

15 15 import br.sistema.crud.jdbc.dto.UfDTO;
16 16 import br.sistema.crud.jdbc.exception.PersistenciaExcpetion;
17 17
18 - public class PessoaDAO implements GenericoDAO<PessoaDTO> {
18 + public class PessoaDAO {
19 +     private static final String TB_PESSOA = "pessoa";
20 +     private static final String TB_ENDERECO = "endereco";
19 21
20 -     @Override
21 -     public void inserir(PessoaDTO pessoaDTO) throws PersistenciaExcpetion {
22 -         try {
23 -             int chaveEnd = insereEndreco(pessoaDTO.getEnderecoDTO());
22 +     public void inserir(Pessoa pessoa) {
23 +         inserirPessoa(pessoa);
24 +         inserirEndereco(pessoa.getEndereco());
25 +     }
24 26
25 -         Connection connection =
                ConexaoUtil.getInstance().getConnection();

```

```

26         String sql = "INSERT INTO TB_PESSOA(NOME, CPF, SEXO,
27         DT_NASC, COD_ENDERECO) " +
28         "VALUES(?, ?, ?, ?, ?)";
29
30         PreparedStatement statement =
31         connection.prepareStatement(sql);
32         statement.setString(1, pessoaDTO.getNome());
33         statement.setLong(2, pessoaDTO.getCpf());
34         //statement.setString(3, pessoaDTO.getEndereco());
35         statement.setString(3,
36         String.valueOf(pessoaDTO.getSexo()));
37         statement.setDate(4, new
38         Date(pessoaDTO.getDtNascimento().getTime()));
39         statement.setInt(5, chaveEnd);
40
41         statement.execute();
42         connection.close();
43     } catch (Exception e) {
44         e.printStackTrace();
45         throw new PersistenciaExcpetion(e.getMessage(), e);
46     }
47 }
48
49 private int insereEndreco(EnderecoDTO enderecoDTO) throws
50 PersistenciaExcpetion{
51     int chave = 0;
52     try {
53
54         Connection connection =
55         ConexaoUtil.getInstance().getConnection();
56
57         String sql = "INSERT INTO TB_Endereco(LOGADOURO, BAIRRO,
58         CIDADE, NUMERO, CEP, COD_UF) " +
59         "VALUES(?, ?, ?, ?, ?, ?)";
60
61         PreparedStatement statement =
62         connection.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
63         statement.setString(1, enderecoDTO.getLogadouro());
64         statement.setString(2, enderecoDTO.getBairro());
65         statement.setString(3, enderecoDTO.getCidade());
66         statement.setLong(4, enderecoDTO.getNumero());
67         statement.setInt(5, enderecoDTO.getCep());
68         statement.setInt(6, enderecoDTO.getUfDTO().getIdUF());
69         statement.execute();
70         ResultSet result = statement.getGeneratedKeys();
71         if(result.next()){
72
73             chave = result.getInt(1);
74         }
75         connection.close();
76     } catch (Exception e) {
77         e.printStackTrace();
78         throw new PersistenciaExcpetion(e.getMessage(), e);
79     }
80 }

```

```

70         -             return chave;
71         -
72         -     }
27 +     private void inserirPessoa(Pessoa pessoa) {
28 +         // Lógica para inserir pessoa na tabela TB_PESSOA
29 +     }
73 30
74         -     @Override
75         -     public void atualizar(PessoaDTO pessoaDTO) throws PersistenciaExcpetion {
76         -         try {
77         -             Connection connection =
78         -                 ConexaoUtil.getInstance().getConnection();
79         -
80         -             String sql = "UPDATE TB_PESSOA " +
81         -                 " SET NOME = ?, " +
82         -                 " CPF = ?, " +
83         -                 " SEXO = ?, " +
84         -                 " DT_NASC = ? " +
85         -                 " WHERE ID_PESSOA = ?";
86         -
87         -             PreparedStatement statement =
88         -                 connection.prepareStatement(sql);
89         -             statement.setString(1, pessoaDTO.getNome());
90         -             statement.setLong(2, pessoaDTO.getCpf());
91         -             //statement.setString(3, pessoaDTO.getEndereco());
92         -             statement.setString(3,
93         -                 String.valueOf(pessoaDTO.getSexo()));
94         -             statement.setDate(4, new
95         -                 Date(pessoaDTO.getDtnascimento().getTime()));
96         -             statement.setInt(5, pessoaDTO.getIdPessoa());
97         -
98         -             statement.execute();
99         -             connection.close();
100         -             //ATUALIZA AGORA O relacionamento da pessoa
101         -             atualizaEndereco(pessoaDTO.getEnderecoDTO());
102         -         } catch (Exception e) {
103         -             e.printStackTrace();
104         -             throw new PersistenciaExcpetion(e.getMessage(), e);
105         -         }
106         -     }
31 +     private void inserirEndereco(Endereco endereco) {
32 +         // Lógica para inserir endereço na tabela TB_ENDERECO
33 +     }
103 34
104         -     @Override
105         -     public void deletar(Integer idPessoa) throws PersistenciaExcpetion {
106         -         try {
107         -             Connection connection =
108         -                 ConexaoUtil.getInstance().getConnection();
109         -
110         -             String sql = "DELETE FROM TB_PESSOA WHERE ID_PESSOA = ?";

```

```
111         PreparedStatement statement =
112             connection.prepareStatement(sql);
113
114         statement.setInt(1, idPessoa);
115
116         statement.execute();
117         connection.close();
118
119         //faz a deleção do endereço correspondente ao cadastro
120
121         } catch (Exception e) {
122             e.printStackTrace();
123             throw new PersistenciaExcpetion(e.getMessage(), e);
124         }
125     }
126
127     public void deletarEndereco(Integer idEndereco) throws
128     PersistenciaExcpetion{
129         try {
130             Connection connection =
131             ConexaoUtil.getInstance().getConnection();
132
133             String sql = "DELETE FROM TB_endereco WHERE ID_endereco =
134             ?";
135
136             PreparedStatement statement =
137             connection.prepareStatement(sql);
138
139             statement.setInt(1, idEndereco);
140
141             statement.execute();
142             connection.close();
143
144             } catch (Exception e) {
145                 e.printStackTrace();
146                 throw new PersistenciaExcpetion(e.getMessage(), e);
147             }
148         }
149
150     private void atualizaEndereco(EnderecoDTO enderecoDTO) throws
151     PersistenciaExcpetion{
152         try{
153             Connection connection =
154             ConexaoUtil.getInstance().getConnection();
155
156             String sql = "UPDATE TB_ENDERECO " +
157             " SET LOGADOURO = ?, " +
158             " BAIRRO = ?, " +
159             " CIDADE = ?, " +
160             " NUMERO = ?, " +
161             " CEP = ?, " +
162             " COD_UF = ?" +
163             " WHERE ID_ENDERECO = ? ";
```

```

155 - PreparedStatement preparedStatement =
    connection.prepareStatement(sql);
156 - preparedStatement.setString(1, enderecoDTO.getLogadouro());
157 - preparedStatement.setString(2, enderecoDTO.getBairro());
158 - preparedStatement.setString(3, enderecoDTO.getCidade());
159 - preparedStatement.setLong(4, enderecoDTO.getNumero());
160 - preparedStatement.setInt(5, enderecoDTO.getCep());
161 - preparedStatement.setInt(6,
    enderecoDTO.getUfDTO().getIdUF());
162 - preparedStatement.setInt(7, enderecoDTO.getIdEndereco());
163 - preparedStatement.execute();
164 - connection.close();
165 - } catch (Exception e) {
166 -     throw new PersistenciaExcpetion(e.getMessage(), e);
167 - }
168 -
169 - }
170 -
171 - public void deletarTudo() throws PersistenciaExcpetion {
172 -     try {
173 -         Connection connection =
    ConexaoUtil.getInstance().getConnection();
174 -
175 -         String sql = "DELETE FROM TB_PESSOA";
176 -         PreparedStatement statement =
    connection.prepareStatement(sql);
177 -
178 -         statement.execute();
179 -         connection.close();
180 -     } catch (Exception e) {
181 -         e.printStackTrace();
182 -         throw new PersistenciaExcpetion(e.getMessage(), e);
183 -     }
184 -
185 - }
186 35 + public void atualizar(Pessoa pessoa) {
187 36 +     atualizarPessoa(pessoa);
188 37 +     atualizarEndereco(pessoa.getEndereco());
189 38 + }
190 39
191 - @Override
192 - public List<PessoaDTO> listarTodos() throws PersistenciaExcpetion {
193 -     List<PessoaDTO> listaPessoas = new ArrayList<PessoaDTO>();
194 -     try {
195 -         Connection connection =
    ConexaoUtil.getInstance().getConnection();
196 -
197 -         String sql = "SELECT * FROM TB_PESSOA";
198 -
199 -         PreparedStatement statement =
    connection.prepareStatement(sql);
200 -
201 -         ResultSet resultSet = statement.executeQuery();

```

```

197 -
198 -         while (resultSet.next()) {
199 -             PessoaDTO pessoaDTO = new PessoaDTO();
200 -
201 -             pessoaDTO.setIdPessoa(resultSet.getInt("id_pessoa"));
202 -             pessoaDTO.setNome(resultSet.getString("nome"));
203 -             pessoaDTO.setCpf(resultSet.getLong("cpf"));
204 -
205 -             pessoaDTO.setSexo(resultSet.getString("sexo").charAt(0));
206 -
207 -             pessoaDTO.setDtNascimento(resultSet.getDate("dt_nasc"));
208 -
209 -             pessoaDTO.setEnderecoDTO(buscaEnderecoPorId(resultSet.getInt("cod_endereco")));
210 -
211 -             listaPessoas.add(pessoaDTO);
212 -         }
213 -         connection.close();
214 -     } catch (Exception e) {
215 -         e.printStackTrace();
216 -         throw new PersistenciaExcpetion(e.getMessage(), e);
217 -     }
218 -     return listaPessoas;
219 - }
220 -
221 - public EnderecoDTO buscaEnderecoPorId(Integer idEndereco) throws
222 - PersistenciaExcpetion{
223 -     EnderecoDTO enderecoDTO = null;
224 -     try{
225 -         Connection connection =
226 -         ConexaoUtil.getInstance().getConnection();
227 -         String sql = "select * from tb_endereco where id_endereco =
228 - ?";
229 -         PreparedStatement preparedStatement =
230 -         connection.prepareStatement(sql);
231 -         preparedStatement.setInt(1, idEndereco);
232 -         ResultSet result = preparedStatement.executeQuery();
233 -         if(result.next()){
234 -             enderecoDTO = new EnderecoDTO();
235 -             enderecoDTO.setIdEndereco(result.getInt(1));
236 -             enderecoDTO.setLogradouro(result.getString(2));
237 -             enderecoDTO.setBairro(result.getString(3));
238 -             enderecoDTO.setCidade(result.getString(4));
239 -             enderecoDTO.setNumero(result.getLong(5));
240 -             enderecoDTO.setCep(result.getInt(6));
241 -
242 -             enderecoDTO.setUfDTO(buscaUFPorId(result.getInt(7)));
243 -
244 -         }
245 -         connection.close();
246 -     } catch (Exception e){
247 -         e.printStackTrace();
248 -         throw new PersistenciaExcpetion(e.getMessage(), e);

```



```

240         }
241         return enderecoDTO;
242     }
243     }
244
245     private UfDTO buscaUFPorId(Integer idUf) throws PersistenciaExcpetion{
246
247         UfDTO ufDTO = null;
248         try{
249             Connection connection =
250             ConexaoUtil.getInstance().getConnection();
251             String sql = "select * from tb_uf where id_uf = ?";
252             PreparedStatement preparedStatment =
253             connection.prepareStatement(sql);
254             preparedStatment.setInt(1, idUf);
255
256             ResultSet result = preparedStatment.executeQuery();
257
258             if(result.next()){
259                 ufDTO = new UfDTO();
260                 ufDTO.setIdUF(result.getInt(1));
261                 ufDTO.setSiglaUF(result.getString(2));
262                 ufDTO.setDescricao(result.getString(3));
263             }
264
265             }catch(Exception e){
266                 throw new PersistenciaExcpetion(e.getMessage(), e);
267             }
268             return ufDTO;
269         }
270
271     private void atualizarPessoa(Pessoa pessoa) {
272         // Lógica para atualizar pessoa na tabela TB_PESSOA
273     }
274
275     @Override
276     public PessoaDTO buscarPorId(Integer id) throws PersistenciaExcpetion {
277         PessoaDTO pessoaDTO = null;
278         try {
279             Connection connection =
280             ConexaoUtil.getInstance().getConnection();
281
282             String sql = "SELECT * FROM TB_PESSOA WHERE ID_PESSOA = ?";
283
284             PreparedStatement statement =
285             connection.prepareStatement(sql);
286             statement.setInt(1, id);
287
288             ResultSet resultSet = statement.executeQuery();
289             if (resultSet.next()) {
290                 pessoaDTO = new PessoaDTO();
291
292                 pessoaDTO.setIdPessoa(resultSet.getInt("id_pessoa"));

```

```
284         pessoaDTO.setNome(resultSet.getString("nome"));
285         pessoaDTO.setCpf(resultSet.getLong("cpf"));
286
287         pessoaDTO.setDtNascimento(resultSet.getDate("dt_nasc"));
288
289         //pessoaDTO.setEndereco(resultSet.getString("endereco"));
290
291         pessoaDTO.setSexo(resultSet.getString("sexo").charAt(0));
292
293         pessoaDTO.setEnderecoDTO(buscaEnderecoPorId(resultSet.getInt("cod_endereco")));
294     }
295     connection.close();
296 } catch (Exception e) {
297     e.printStackTrace();
298     throw new PersistenciaExcpetion(e.getMessage(), e);
299 }
300 return pessoaDTO;
301 }
302
303 public List<PessoaDTO> filtraPessoa(String nome, Long cpf, String sexo,
304 String orderBy) throws PersistenciaExcpetion {
305     List<PessoaDTO> lista = new ArrayList<PessoaDTO>();
306     try {
307         Connection connection =
308             ConexaoUtil.getInstance().getConnection();
309
310         String sql = "SELECT * FROM TB_PESSOA ";
311         boolean ultimo = false;
312         if (nome != null && !nome.equals("")) {
313             sql += " WHERE NOME LIKE ?";
314             ultimo = true;
315         }
316
317         if (cpf != null && !cpf.equals("")) {
318             if (ultimo) {
319                 sql += " AND ";
320             } else {
321                 sql += " WHERE ";
322                 ultimo = true;
323             }
324             sql += " CPF LIKE ?";
325         }
326
327         if (sexo != null && !sexo.equals("")) {
328             if (ultimo) {
329                 sql += " AND ";
330             } else {
331                 sql += " WHERE ";
332                 ultimo = true;
333             }
334             sql += " SEXO = ?";
335         }
336
337         sql += " ORDER BY " + orderBy;
```

```
330 -
331 -         PreparedStatement statement =
332 -             connection.prepareStatement(sql);
333 -         int cont = 0;
334 -         if (nome != null && !nome.equals("")) {
335 -             statement.setString(++cont, "%" + nome + "%");
336 -         }
337 -         if (cpf != null && !cpf.equals("")) {
338 -             statement.setString(++cont, "%" + cpf + "%");
339 -         }
340 -         if (sexo != null && !sexo.equals("")) {
341 -             statement.setString(++cont, sexo);
342 -         }
343 -         ResultSet resultSet = statement.executeQuery();
344 -
345 -         while (resultSet.next()) {
346 -             PessoaDTO pessoaDTO = new PessoaDTO();
347 -
348 -             pessoaDTO.setIdPessoa(resultSet.getInt("id_pessoa"));
349 -             pessoaDTO.setNome(resultSet.getString("nome"));
350 -             pessoaDTO.setCpf(resultSet.getLong("cpf"));
351 -
352 -             pessoaDTO.setDtNascimento(resultSet.getDate("dt_nasc"));
353 -             //pessoaDTO.setEndereco(resultSet.getString("endereco"));
354 -             pessoaDTO.setSexo(resultSet.getString("sexo").charAt(0));
355 -
356 -             lista.add(pessoaDTO);
357 -         }
358 -     } catch (Exception e) {
359 -         e.printStackTrace();
360 -         throw new PersistenciaExcpetion(e.getMessage(), e);
361 -     }
362 -     return lista;
363 - }
364 -
365 -
366 -
367 + private void atualizarEndereco(Endereco endereco) {
368 +     // Lógica para atualizar endereço na tabela TB_ENDERECO
369 + }
370 +
371 + public void deletar(Pessoa pessoa) {
372 +     deletarEndereco(pessoa.getEndereco());
373 +     deletarPessoa(pessoa);
374 + }
375 +
376 + private void deletarEndereco(Endereco endereco) {
```

```

54 + // Lógica para deletar endereço da tabela TB_ENDERECO
55 + }
56 +
57 + private void deletarPessoa(Pessoa pessoa) {
58 + // Lógica para deletar pessoa da tabela TB_PESSOA
59 + }
60 +
61 + public List<Pessoa> listarTodos() {
62 + List<Pessoa> pessoas = buscarTodasPessoas();
63 + for (Pessoa pessoa : pessoas) {
64 + Endereco endereco = buscaEnderecoPorId(pessoa.getEndereco().getId());
65 + pessoa.setEndereco(endereco);
66 + }
67 + return pessoas;
68 + }
69 +
70 + private List<Pessoa> buscarTodasPessoas() {
71 + // Lógica para buscar todas as pessoas da tabela TB_PESSOA
72 + }
73 +
74 + private Endereco buscaEnderecoPorId(int id) {
75 + // Lógica para buscar endereço por ID na tabela TB_ENDERECO
76 + }
77 +
78 + public List<Pessoa> filtraPessoa(String filtro) {
79 + List<Pessoa> pessoas = filtrarPessoas(filtro);
80 + for (Pessoa pessoa : pessoas) {
81 + UF uf = buscaUFPorId(pessoa.getEndereco().getUf().getId());
82 + pessoa.getEndereco().setUf(uf);
83 + }
84 + return pessoas;
85 + }
86 +
87 + private List<Pessoa> filtrarPessoas(String filtro) {
88 + // Lógica para filtrar pessoas da tabela TB_PESSOA com base no filtro
89 + }
90 +
91 + private UF buscaUFPorId(int id) {
92 + // Lógica para buscar UF por ID na tabela de UF
93 + }
94 }

```

367

▼ 79 src/br/sistema/crud/jdbc/dao/UfDAO.java

```

9 9
10 10 import br.sistema.crud.jdbc.ConexaoUtil;
11 11 import br.sistema.crud.jdbc.dto.UfDTO;
12 - import br.sistema.crud.jdbc.exception.NegocioException;
13 - import br.sistema.crud.jdbc.exception.PersistenciaExcpetion;
12 + import br.sistema.crud.jdbc.exception.PersistenciaException;
14 13
15 14 public class UfDAO {

```

```

16 -
17 -
18 - public List<UfDTO> listaEstado() throws PersistenciaExcpetion{
19 -
20 -     List<UfDTO> lista = new ArrayList<>();
21 -     try{
22 -         Connection connection = ConexaoUtil.getInstance().getConnection();
23 -
24 -         String sql = "select * from tb_uf";
25 -         PreparedStatement preparedstatmente =
connection.prepareStatement(sql);
26 -
27 -         ResultSet resultado = preparedstatmente.executeQuery();
28 -         while(resultado.next()){
29 -             UfDTO ufDTO = new UfDTO();
30 -             ufDTO.setIdUF(resultado.getInt(1));
31 -             ufDTO.setSiglaUF(resultado.getString(2));
32 -             ufDTO.setDescricao(resultado.getString(3));
33 -             lista.add(ufDTO);
34 -         }
35 -         connection.close();
36 -     }catch(Exception e){
37 -         e.printStackTrace();
38 -         throw new PersistenciaExcpetion(e.getMessage(), e);
39 -     }
40 -
41 -     return lista;
42 - }

```

```

43 15
16 + public List<UfDTO> listaEstado() throws PersistenciaException {
17 +     List<UfDTO> lista = new ArrayList<>();
18 +     Connection connection = null;
19 +     PreparedStatement preparedStatement = null;
20 +     ResultSet resultado = null;
21 +
22 +     try {
23 +         connection = ConexaoUtil.getInstance().getConnection();
24 +         String sql = "SELECT * FROM tb_uf";
25 +         preparedStatement = connection.prepareStatement(sql);
26 +         resultado = preparedStatement.executeQuery();
27 +
28 +         while (resultado.next()) {
29 +             UfDTO ufDTO = criarUfDTO(resultado);
30 +             lista.add(ufDTO);
31 +         }
32 +     } catch (SQLException e) {
33 +         throw new PersistenciaException(e.getMessage(), e);
34 +     } finally {
35 +         closeResources(connection, preparedStatement, resultado);
36 +     }
37 +
38 +     return lista;

```

```
39 +     }
40 +
41 +     private UfDTO criarUfDTO(ResultSet resultado) throws SQLException {
42 +         UfDTO ufDTO = new UfDTO();
43 +         ufDTO.setIdUF(resultado.getInt("idUF"));
44 +         ufDTO.setSiglaUF(resultado.getString("siglaUF"));
45 +         ufDTO.setDescricao(resultado.getString("descricao"));
46 +         return ufDTO;
47 +     }
48 +
49 +     private void closeResources(Connection connection, PreparedStatement
        preparedStatement, ResultSet resultado) {
50 +         try {
51 +             if (resultado != null) {
52 +                 resultado.close();
53 +             }
54 +             if (preparedStatement != null) {
55 +                 preparedStatement.close();
56 +             }
57 +             if (connection != null) {
58 +                 connection.close();
59 +             }
60 +         } catch (SQLException e) {
61 +             // Lidar com a exceção ou registrar o erro
62 +             e.printStackTrace();
63 +         }
64 +     }
44 65 }
```

0 comments on commit 58ae9a4

Please [sign in](#) to comment.