EdsonRodrigoBA / -Sistema-Cadastro-de-Clientes Public

Code Issues Pull requests Actions Projects Security Insights

Commit

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

Refatoração em bo

Browse files

```
LoginBO - Técnicas: Extract Method, Simplification of Conditions

PessoaBO - Técnicas: Move Method, Extract Method, Encapsulate Field

UfBO - Técnicas: Extract Method, Encapsulate Field

ValidaCamposPessoasBO - Técnicas: Extract Method
```

 **moisesocosta** committed on Jun 22

1 parent 90dff36 commit 0643563

Showing **4 changed files** with **78 additions** and **94 deletions**.

Whitespace Ignore whitespace Split Unified

42 src/br/sistema/crud/jdbc/bo/LoginBO.java

```
...  ...  @@ -1,32 +1,30 @@
 1    1   package br.sistema.crud.jdbc.bo;
 2    2
 3    3   import br.sistema.crud.jdbc.dao.LoginDAO;
 4        - import br.sistema.crud.jdbc.dao.PessoaDAO;
 5    4   import br.sistema.crud.jdbc.dto.LoginDTO;
 6        - import br.sistema.crud.jdbc.dto.PessoaDTO;
 7    5   import br.sistema.crud.jdbc.exception.NegocioException;
 8    6
 9    7   public class LoginBO {
10        -
11        -       public boolean logar(LoginDTO loginDTO) throws NegocioException{
12        -               boolean resultado = false;
13        -               try{
14        -                       if(loginDTO.getNome() == null || "".equals(loginDTO)){
15        -                               throw new NegocioException("Logn Obrigatorio");
16        -                       }else if(loginDTO.getSenha() == null ||
    "".equals(loginDTO.getSenha())){
```

```
-                                throw new NegocioException("Logn Obrigatorio");
                        }else{
-                                LoginDAO loginDAO = new LoginDAO();
-                                resultado= loginDAO.logar(loginDTO);
-
-                                }
-
-                }catch(Exception e){
-                        e.printStackTrace();
-                        throw new NegocioException(e.getMessage(),e);
-                }
-                return resultado;
-        }
+    private static final String MENSAGEM_LOGIN_OBRIGATORIO = "Login obrigatório";

+    public boolean logar(LoginDTO loginDTO) throws NegocioException {
+        validarLogin(loginDTO);
+
+        try {
+            LoginDAO loginDAO = new LoginDAO();
+            return loginDAO.logar(loginDTO);
+        } catch (Exception e) {
+            e.printStackTrace();
+            throw new NegocioException(e.getMessage(), e);
+        }
+    }
+
+    private void validarLogin(LoginDTO loginDTO) throws NegocioException {
+        if (loginDTO.getNome() == null || loginDTO.getNome().isEmpty()) {
+            throw new NegocioException(MENSAGEM_LOGIN_OBRIGATORIO);
+        } else if (loginDTO.getSenha() == null || loginDTO.getSenha().isEmpty()) {
+            throw new NegocioException(MENSAGEM_LOGIN_OBRIGATORIO);
+        }
+    }
 }
```

78 src/br/sistema/crud/jdbc/bo/PessoaBO.java

```
 public class PessoaBO {

        private DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
+       private PessoaDAO pessoaDAO;
+
+       public PessoaBO() {
+               this.pessoaDAO = new PessoaDAO();
+       }

        public void cadastrar(PessoaDTO pessoaDTO) throws NegocioException {
                try {
-                       PessoaDAO pessoaDAO = new PessoaDAO();
                        pessoaDAO.inserir(pessoaDTO);
```

```
22  26          } catch(Exception exception) {
23  27                  exception.printStackTrace();
26  30          }
27  31
28  32          public String[][] listagem(List<Integer> idsPessoas) throws
                NegocioException {
29      -               int numCols = 10;
30      -               String[][] listaRetorno = null;
    33  +               final int numCols = 10;
31  34              try {
32      -                   PessoaDAO pessoaDAO = new PessoaDAO();
33      -
34  35                  List<PessoaDTO> lista = pessoaDAO.listarTodos();
35      -                   listaRetorno = new String[lista.size()][numCols];
    36  +                   String[][] listaRetorno = new String[lista.size()]
                        [numCols];
36  37
37  38                  for (int i = 0; i < lista.size(); i++) {
38  39                      PessoaDTO pessoa = lista.get(i);
49  50                      listaRetorno[i][8] = "ALTER";
50  51                      listaRetorno[i][9] = "DEL";
51  52                  }
    53  +
    54  +               return listaRetorno;
52  55              } catch(Exception exception) {
53  56                  throw new NegocioException(exception.getMessage());
54  57              }
55      -           return listaRetorno;
56  58          }
57  59
58  60          public boolean validaNome(String nome) throws ValidacaoException {
59  61              boolean ehValido = true;
60      -           if (nome == null || nome.equals("")) {
    62  +           if (nome == null || nome.isEmpty()) {
61  63                  ehValido = false;
62      -                   throw new ValidacaoException("Campo nome é obrigatório!");
    64  +                   throw new ValidacaoException("Campo nome é obrigatório!");
63  65              } else if (nome.length() > 30) {
64  66                  ehValido = false;
65      -                   throw new ValidacaoException("Campo nome comporta no máximo
                        30 chars!");
    67  +                   throw new ValidacaoException("Campo nome comporta no máximo
                        30 chars!");
66  68              }
67  69              return ehValido;
68  70          }
69  71
70  72          public boolean validaCpf(String cpf) throws ValidacaoException {
71  73              boolean ehValido = true;
72      -           if (cpf == null || cpf.equals("")) {
    74  +           if (cpf == null || cpf.isEmpty()) {
73  75                  ehValido = false;
```

```
74      -                       throw new ValidacaoException("Campo CPF é obrigatório!");
        76  +                   throw new ValidacaoException("Campo CPF é obrigatório!");
75      77                  } else if (cpf.length() != 11) {
76      78                          ehValido = false;
77      -                       throw new ValidacaoException("Campo CPF deve ter 11
        dígitos!");
        79  +                   throw new ValidacaoException("Campo CPF deve ter 11
        dígitos!");
78      80                  } else {
79      81                          char[] digitos = cpf.toCharArray();
80      82                          for (char digito : digitos) {
81      83                                  if (!Character.isDigit(digito)) {
82      84                                          ehValido = false;
83      -                                       throw new ValidacaoException("Campo CPF é
        somente numérico!");
        85  +                                   throw new ValidacaoException("Campo CPF é
        somente numérico!");
84      86                                  }
85      87                          }
86      88                  }
89      91
90      92          public boolean validaEndereco(EnderecoDTO enderecoDTO) throws
        ValidacaoException {
91      93                  boolean ehValido = true;
92      -                   if (enderecoDTO.getLogadouro() == null ||
        enderecoDTO.getLogadouro().equals("")) {
        94  +               if (enderecoDTO.getLogadouro() == null ||
        enderecoDTO.getLogadouro().isEmpty()) {
93      95                          ehValido = false;
94      -                       throw new ValidacaoException("Campo Logradouro é
        obrigatório!");
95      -                   } else if (enderecoDTO.getBairro() == null ||
        enderecoDTO.getBairro().equals("")) {
        96  +                   throw new ValidacaoException("Campo Logradouro é
        obrigatório!");
        97  +               } else if (enderecoDTO.getBairro() == null ||
        enderecoDTO.getBairro().isEmpty()) {
96      98                          ehValido = false;
97      -                       throw new ValidacaoException("Bairro Obrigatorio");
98      -                   } else if (enderecoDTO.getNumero() == null ||
        enderecoDTO.getNumero().equals(0)) {
        99  +                   throw new ValidacaoException("Bairro Obrigatório");
        100 +               } else if (enderecoDTO.getNumero() == null ||
        enderecoDTO.getNumero() == 0) {
99      101                         ehValido = false;
100     -                       throw new ValidacaoException("Numero Obrigatorio");
101     -                   }else if (enderecoDTO.getCep() == null ||
        enderecoDTO.getCep().equals(0)) {
        102 +                   throw new ValidacaoException("Número Obrigatório");
        103 +               } else if (enderecoDTO.getCep() == null || enderecoDTO.getCep() ==
        0) {
102     104                         ehValido = false;
```

```
103     -                       throw new ValidacaoException("CEP Obrigatorio");
        105 +                   throw new ValidacaoException("CEP Obrigatório");
104     106                 }
105     -
106     -
        107 +
107     108                 return ehValido;
108     109         }
109     110
110     111         public boolean validaDtNasc(String dtNasc) throws ValidacaoException {
111     112                 boolean ehValido = true;
112     -               if (dtNasc == null || dtNasc.equals("")) {
        113 +           if (dtNasc == null || dtNasc.isEmpty()) {
113     114                         ehValido = false;
114     -                       throw new ValidacaoException("Campo Dt. Nasc. é
                                obrigatório!");
        115 +                   throw new ValidacaoException("Campo Dt. Nasc. é
                                obrigatório!");
115     116                 } else {
116     -                       ehValido = false;
117     117                         try {
118     118                                 dateFormat.parse(dtNasc);
119     119                         } catch (ParseException e) {
120     -                               throw new ValidacaoException("Formato inválido de
                                data!");
        120 +                           ehValido = false;
        121 +                           throw new ValidacaoException("Formato inválido de
                                data!");
121     122                         }
122     123                 }
123     124                 return ehValido;
124     125         }
125     126
126     127         public String[][] listaConsulta(String nome, Long cpf, char sexo, String
                    orderBy) throws NegocioException {
127     -               int numCols = 6;
128     -               String[][] listaRetorno = null;
        128 +           final int numCols = 6;
129     129                 try {
130     -                       PessoaDAO pessoaDAO = new PessoaDAO();
131     -
132     130                         List<PessoaDTO> lista = pessoaDAO.filtraPessoa(nome, cpf,
                                String.valueOf(sexo), orderBy);
133     -                       listaRetorno = new String[lista.size()][numCols];
        131 +                   String[][] listaRetorno = new String[lista.size()]
                                [numCols];
134     132
135     133                         for (int i = 0; i < lista.size(); i++) {
136     134                                 PessoaDTO pessoa = lista.get(i);
141     139                                 listaRetorno[i][4] = pessoa.getSexo() == 'M' ?
                                "Masculino" : "Feminino";
```

```
142  140                                listaRetorno[i][5] =
     dateFormat.format(pessoa.getDtNascimento());
143  141                        }
     142 +
     143 +                      return listaRetorno;
144  144                } catch(Exception exception) {
145  145                        throw new NegocioException(exception.getMessage());
146  146                }
147       -                return listaRetorno;
148  147        }
149  148
150  149        public void removePessoa(Integer idPessoa, Integer idEndereco) throws
     NegocioException {
151  150                try {
152       -                        PessoaDAO pessoaDAO = new PessoaDAO();
153  151                        pessoaDAO.deletar(idPessoa);
154  152                        pessoaDAO.deletarEndereco(idEndereco);
155  153                } catch(Exception exception) {
159  157
160  158        public void removeTudo() throws NegocioException {
161  159                try {
162       -                        PessoaDAO pessoaDAO = new PessoaDAO();
163  160                        pessoaDAO.deletarTudo();
164       -
165  161                } catch(Exception e) {
166  162                        throw new NegocioException(e.getMessage());
167  163                }
168  164        }
169  165
170  166        public PessoaDTO buscaPorId(Integer idPessoa) throws NegocioException{
171       -                PessoaDTO pessoaDTO = null;
172       -
173  167                try {
174       -                        PessoaDAO pessoaDAO = new PessoaDAO();
175       -                        pessoaDTO = pessoaDAO.buscarPorId(idPessoa);
     168 +                        return pessoaDAO.buscarPorId(idPessoa);
176  169                } catch (Exception e) {
177  170                        throw new NegocioException(e.getMessage(), e);
178       -
179  171                }
180       -                return pessoaDTO;
181  172        }
182       -
183  173  }
```

28 src/br/sistema/crud/jdbc/bo/UfBO.java

```
...  ...  @@ -1,28 +1,24 @@
1    1    package br.sistema.crud.jdbc.bo;
2    2
3         - import java.util.ArrayList;
```

```
4   3   import java.util.List;
5   4
6   5   import br.sistema.crud.jdbc.dao.UfDAO;
7       - import br.sistema.crud.jdbc.dto.UfDTO;
8   6   import br.sistema.crud.jdbc.exception.NegocioException;
    7   + import br.sistema.crud.jdbc.model.Uf;
9   8
10  9   public class UfBO {
11  10
12      -       public List<UfDTO> listaUfs() throws NegocioException{
13      -               List<UfDTO> lista = null;
14      -
15      -               try{
16      -                       UfDAO ufDAO = new UfDAO();
17      -                       lista = ufDAO.listaEstado();
18      -
19      -               }catch (Exception e){
20      -                       throw new NegocioException(e.getMessage(),e);
    11  +       private UfDAO ufDAO;
    12  +
    13  +       public UfBO() {
    14  +               this.ufDAO = new UfDAO();
    15  +       }
    16  +
    17  +       public List<Uf> listarUfs() throws NegocioException {
    18  +               try {
    19  +                       return ufDAO.listarUfs();
    20  +               } catch (Exception e) {
    21  +                       throw new NegocioException(e.getMessage(), e);
21  22              }
22      -               return lista;
23      -
24      -
25      -
26  23          }
27      -
28  24  }
```

24 src/br/sistema/crud/jdbc/bo/ValidaCamposPessoasBO.java

```
...     ...   @@ -1,23 +1,23 @@
1   1     package br.sistema.crud.jdbc.bo;
2   2
3       -
4   3     import javax.swing.text.AttributeSet;
5   4     import javax.swing.text.BadLocationException;
6   5     import javax.swing.text.PlainDocument;
7   6
8       -
9   7     public class ValidaCamposPessoasBO extends PlainDocument {
10      -
```

```
11  8
12      -       private static final long serialVersionUID = 5649489410014769260L;
13      -
14      -       public void insertString(int offSet, String str, AttributeSet attr)throws
          BadLocationException{
15      -               super.insertString(offSet, str.replaceAll("[^a-z|^A-Z |^ ]",""),
          attr);
16      -       }
17      -
18      -       public void nome(int offSet, String str,javax.swing.text.AttributeSet
          attr)throws BadLocationException{
19      -               super.insertString(offSet, str.replaceAll("[^a-z|^A-Z]",""),
          attr);
        9   +   private static final long serialVersionUID = 5649489410014769260L;
        10  +
        11  +   @Override
        12  +   public void insertString(int offset, String str, AttributeSet attr) throws
          BadLocationException {
        13  +       super.insertString(offset, removeNonAlphabeticCharacters(str), attr);
        14  +   }
20      15
21      -       }
        16  +   public void nome(int offset, String str, AttributeSet attr) throws
          BadLocationException {
        17  +       super.insertString(offset, removeNonAlphabeticCharacters(str), attr);
        18  +   }
22      19
        20  +   private String removeNonAlphabeticCharacters(String str) {
        21  +       return str.replaceAll("[^a-zA-Z ]", "");
        22  +   }
23      23  }
```

0 comments on commit `0643563`

Please sign in to comment.