

 [steffmartin](#) / [condominio](#) Public[Code](#) [Issues 1](#) [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.

Extract Method - 11, Rename Method - 2, Comments - 3, Hide Method - 8[Browse files](#)**EvertonDev2002** committed on Jun 281 parent [d995286](#) commit [f454295](#)

Showing 21 changed files with 982 additions and 1,039 deletions.

[Whitespace](#)[Ignore whitespace](#)[Split](#)[Unified](#)▼  16  [src/main/java/app/condominio/domain/Bloco.java](#) 

```
2      2
3      3      import java.io.Serializable;
4      4      import java.util.ArrayList;
5      5      + import java.util.Collections;
6      6      import java.util.List;
7      7
8      8      import javax.persistence.CascadeType;
19     20      import javax.validation.constraints.NotBlank;
20     21      import javax.validation.constraints.Size;
21     22
22     23      - @SuppressWarnings("serial")
23     24      +
24     25      @Entity
25     26      @Table(name = "blocos")
26     27      public class Bloco implements Serializable, Comparable<Bloco> {
42     43
43     44          @OneToMany(mappedBy = "bloco", fetch = FetchType.LAZY, cascade =
44     45          CascadeType.REMOVE, orphanRemoval = true)
45     46          @OrderBy(value = "sigla")
46     47          - private List<Moradia> moradias = new ArrayList<>();
47     48          + private final List<Moradia> moradias = new ArrayList<>();
48     49
49     50      public Long getIdBloco() {
50     51          return idBloco;
51     52      }
52     53
53     54      }
```

```

79      80      public List<Moradia> getMoradias() {
80      -      return moradias;
      81      +      return Collections.unmodifiableList(moradias);
      82      +      }
      83      +
      84      +      public void setMoradias(List<Moradia> novaMoradias) {
      85      +          this.moradias.clear();
      86      +          this.moradias.addAll(novaMoradias);
81      87      }
82      88
83      -      public void setMoradias(List<Moradia> moradias) {
84      -          this.moradias = moradias;
      89      +      public void removeMoradia(Moradia moradia) {
      90      +          this.moradias.remove(moradia);
85      91      }
86      92
87      93      @Override

```

51 src/main/java/app/condominio/domain/Categoria.java

```

2      2
3      3      import java.io.Serializable;
4      4      import java.util.ArrayList;
      5      + import java.util.Collections;
5      6      import java.util.List;
      7      + import java.util.Objects;
6      8
7      9      import javax.persistence.CascadeType;
8      10     import javax.persistence.Column;
30     32     @Table(name = "categorias")
31     33     public class Categoria implements Serializable, Comparable<Categoria> {
32     34
33     -      public static final int NIVEL_MAX = 4;
      35     +      private static final int MAX_NIVEL = 4;
34     36
35     37     @Id
36     38     @GeneratedValue(strategy = GenerationType.IDENTITY)
45     47     @NotBlank
46     48     private String descricao;
47     49
48     -      @Max(NIVEL_MAX)
      50     +      @Max(MAX_NIVEL)
49     51     private Integer nivel;
50     52
51     -      // LATER criar método para ordenação automática, lembrar que edição exige
52     -      // reescrever ordem nas categorias filhas
53     53     @Size(min = 1, max = 255)
54     54     @NotBlank
55     55     private String ordem;
66     66     private List<Categoria> categoriasFilhas = new ArrayList<>();
67     67

```

```

68      68      @OneToMany(mappedBy = "categoriaPai", fetch = FetchType.LAZY, cascade =
CascadeType.REMOVE, orphanRemoval = true)
69      -      @OrderBy(value = "descricao")
70      -      private List<Subcategoria> Subcategorias = new ArrayList<>();
      69      +      @OrderBy("descricao")
      70      +      private List<Subcategoria> subcategorias = new ArrayList<>();
71      71
72      -      public static Integer getNivelMax() {
73      -          return NIVEL_MAX;
      72      +      public static final int getMaxNivel() {
      73      +          return MAX_NIVEL;
74      74      }
75      75
76      76      public Long getIdCategoria() {
130     130      }
131     131
132     132      public List<Categoria> getCategoriasFilhas() {
133     -          return categoriasFilhas;
      133     +          return Collections.unmodifiableList(categoriasFilhas);
134     134      }
135     135
136     136      public void setCategoriasFilhas(List<Categoria> categoriasFilhas) {
137     -          this.categoriasFilhas = categoriasFilhas;
      137     +          this.categoriasFilhas = new ArrayList<>(categoriasFilhas);
      138     +      }
      139     +
      140     +      public void removeCategoriaFilha(Categoria categoriaFilha) {
      141     +          this.categoriasFilhas.remove(categoriaFilha);
138     142      }
139     143
140     144      public List<Subcategoria> getSubcategorias() {
141     -          return Subcategorias;
      145     +          return Collections.unmodifiableList(subcategorias);
142     146      }
143     147
144     148      public void setSubcategorias(List<Subcategoria> subcategorias) {
145     -          Subcategorias = subcategorias;
      149     +          this.subcategorias = new ArrayList<>(subcategorias);
      150     +      }
      151     +
      152     +      public void removeSubcategoria(Subcategoria subcategoria) {
      153     +          this.subcategorias.remove(subcategoria);
146     154      }
147     155
148     156      @Override
152     160
153     161      @Override
154     162      public int hashCode() {
155     -          final int prime = 31;
156     -          int result = 1;
157     -          result = prime * result + ((idCategoria == null) ? 0 :
idCategoria.hashCode());

```

```

158      -         return result;
      163      +         return Objects.hash(idCategoria);
159      164      }
160      165
161      166      @Override
162      167      public boolean equals(Object obj) {
163      168          if (this == obj) {
164      169              return true;
165      170          }
166      171      -         if (obj == null) {
167      172      -             return false;
168      173      -         }
169      174      -         if (getClass() != obj.getClass()) {
      171      +         if (obj == null || getClass() != obj.getClass()) {
      172      +             return false;
      173      +         }
170      173      }
171      174      Categoria other = (Categoria) obj;
172      175      -         if (idCategoria == null) {
173      176      -             if (other.idCategoria != null) {
174      177      -                 return false;
175      178      -             }
176      179      -         } else if (!idCategoria.equals(other.idCategoria)) {
177      180      -             return false;
178      181      -         }
179      182      -         return true;
      175      +         return Objects.equals(idCategoria, other.idCategoria);
180      183      }
181      184
182      185      @Override
183      186

```

▼ 279 src/main/java/app/condominio/domain/Conta.java

```

3      3      import java.io.Serializable;
4      4      import java.math.BigDecimal;
5      5      import java.util.ArrayList;
      6      + import java.util.Collections;
6      7      import java.util.List;
7      8
8      9      import javax.persistence.CascadeType;
21     22      import javax.validation.constraints.NotBlank;
22     23      import javax.validation.constraints.Size;
23     24
      25      + import org.hibernate.mapping.Collection;
      26      +
24     27      @SuppressWarnings("serial")
25     28      @Entity
26     29      @Table(name = "contas")
27     30      @Inheritance(strategy = InheritanceType.JOINED)
28     31      public class Conta implements Serializable, Comparable<Conta> {
29     32
30     33      -         @Id
31     34      -         @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

32 - @Column(name = "idconta")
33 - private Long idConta;
34 -
35 - @Size(min = 1, max = 2)
36 - @NotBlank
37 - private String sigla;
38 -
39 - @Size(max = 30)
40 - private String descricao;
41
42 33 + @Id
43 34 + @GeneratedValue(strategy = GenerationType.IDENTITY)
44 35 + @Column(name = "idconta")
45 36 + private Long idConta;
46
47 37 -
48 38 - @Column(name = "saldoinicial")
49 39 - private BigDecimal saldoInicial;
50 38 + @Size(min = 1, max = 2)
51 39 + @NotBlank
52 40 + private String sigla;
53
54 41
55 42 - // O saldo é atualizado por TRIGGER ao modificar a tabela Movimentos, e é
56 43 - // atualizado no JAVA ao modificar a tabela Contas
57 44 - @Column(name = "saldoatual")
58 45 - private BigDecimal saldoAtual;
59 42 + @Size(max = 30)
60 43 + private String descricao;
61
62 44
63 45 - @ManyToOne(fetch = FetchType.LAZY)
64 46 - @JoinColumn(name = "idcondominio")
65 47 - private Condominio condominio;
66 45 + @Column(name = "saldoinicial")
67 46 + private BigDecimal saldoInicial;
68
69 47
70 48 - @OneToMany(mappedBy = "conta", fetch = FetchType.LAZY, cascade =
71 49 - CascadeType.REMOVE, orphanRemoval = true)
72 50 - private List<Movimento> movimentos = new ArrayList<>();
73 48 + @Column(name = "saldoatual")
74 49 + private BigDecimal saldoAtual;
75
76 50
77 51 - @OneToMany(mappedBy = "contaInversa", fetch = FetchType.LAZY, cascade =
78 52 - CascadeType.REMOVE, orphanRemoval = true)
79 53 - private List<Transferencia> transferenciasRecebidas = new ArrayList<>();
80 51 + @ManyToOne(fetch = FetchType.LAZY)
81 52 + @JoinColumn(name = "idcondominio")
82 53 + private Condominio condominio;
83
84 54
85 55 - public Long getIdConta() {
86 56 - return idConta;
87 57 - }
88 55 + @OneToMany(mappedBy = "conta", fetch = FetchType.LAZY, cascade =
89 56 - CascadeType.REMOVE, orphanRemoval = true)
90 56 + private List<Movimento> movimentos = new ArrayList<>();

```

```

63 57
64 -     public void setIdConta(Long idConta) {
65 -         this.idConta = idConta;
66 -     }
58 +     @OneToMany(mappedBy = "contaInversa", fetch = FetchType.LAZY, cascade =
CascadeType.REMOVE, orphanRemoval = true)
59 +     private List<Transferencia> transferenciasRecebidas = new ArrayList<>();
67 60
68 -     public String getSigla() {
69 -         return sigla;
70 -     }
61 +     public Long getIdConta() {
62 +         return idConta;
63 +     }
71 64
72 -     public void setSigla(String sigla) {
73 -         this.sigla = sigla;
74 -     }
75 -
76 -     public String getDescricao() {
77 -         return descricao;
78 -     }
79 -
80 -     public void setDescricao(String descricao) {
81 -         this.descricao = descricao;
82 -     }
83 -
84 -     public BigDecimal getSaldoInicial() {
85 -         return saldoInicial;
86 -     }
87 -
88 -     public void setSaldoInicial(BigDecimal saldoInicial) {
89 -         this.saldoInicial = saldoInicial;
90 -     }
91 -
92 -     public BigDecimal getSaldoAtual() {
93 -         return saldoAtual;
94 -     }
95 -
96 -     public void setSaldoAtual(BigDecimal saldoAtual) {
97 -         this.saldoAtual = saldoAtual;
98 -     }
99 -
100 -     public Condominio getCondominio() {
101 -         return condominio;
102 -     }
103 -
104 -     public void setCondominio(Condominio condominio) {
105 -         this.condominio = condominio;
106 -     }
107 -
108 -     public List<Movimento> getMovimentos() {

```

```
109         return movimentos;
110     }
111
112     public void setMovimentos(List<Movimento> movimentos) {
113         this.movimentos = movimentos;
114     }
115
116     public List<Transferencia> getTransferenciasRecebidas() {
117         return transferenciasRecebidas;
118     }
119
120     public void setTransferenciasRecebidas(List<Transferencia>
transferenciasRecebidas) {
121         this.transferenciasRecebidas = transferenciasRecebidas;
122     }
123
124     @Override
125     public String toString() {
126         if (descricao != null) {
127             return sigla + " - " + descricao;
128         } else {
129             return sigla;
130         }
131     }
132
133     public String numero() {
134         return "";
135     }
136
137     @Override
138     public int hashCode() {
139         final int prime = 31;
140         int result = 1;
141         result = prime * result + ((idConta == null) ? 0 :
idConta.hashCode());
142         return result;
143     }
144
145     @Override
146     public boolean equals(Object obj) {
147         if (this == obj) {
148             return true;
149         }
150         if (obj == null) {
151             return false;
152         }
153         if (getClass() != obj.getClass()) {
154             return false;
155         }
156         Conta other = (Conta) obj;
157         if (idConta == null) {
158             if (other.idConta != null) {
```

```
159 -         return false;
160 -     }
161 -     } else if (!idConta.equals(other.idConta)) {
162 -         return false;
163 -     }
164 -     return true;
165 - }
166 -
167 - @Override
168 - public int compareTo(Conta o) {
169 -     return this.sigla.compareTo(o.getSigla());
170 - }
171 +
172 + public void setIdConta(Long idConta) {
173 +     this.idConta = idConta;
174 + }
175 +
176 + public String getSigla() {
177 +     return sigla;
178 + }
179 +
180 + public void setSigla(String sigla) {
181 +     this.sigla = sigla;
182 + }
183 +
184 + public String getDescricao() {
185 +     return descricao;
186 + }
187 +
188 + public void setDescricao(String descricao) {
189 +     this.descricao = descricao;
190 + }
191 +
192 + public BigDecimal getSaldoInicial() {
193 +     return saldoInicial;
194 + }
195 +
196 + public void setSaldoInicial(BigDecimal saldoInicial) {
197 +     this.saldoInicial = saldoInicial;
198 + }
199 +
200 + public BigDecimal getSaldoAtual() {
201 +     return saldoAtual;
202 + }
203 +
204 + public void setSaldoAtual(BigDecimal saldoAtual) {
205 +     this.saldoAtual = saldoAtual;
206 + }
207 +
208 + public Condominio getCondominio() {
209 +     return condominio;
210 + }
```



```
105 +     public void setCondominio(Condominio condominio) {
106 +         this.condominio = condominio;
107 +     }
108 +
109 +     public List<Movimento> getMovimentos() {
110 +         return Collections.unmodifiableList(movimentos);
111 +     }
112 +
113 +     public void removeMovimento(Movimento movimento){
114 +         this.movimentos.remove(movimento);
115 +     }
116 +
117 +     public void setMovimentos(List<Movimento> movimentos) {
118 +         this.movimentos.clear();
119 +         if (movimentos != null) {
120 +             this.movimentos.addAll(movimentos);
121 +         }
122 +     }
123 +
124 +     public List<Transferencia> getTransferenciasRecebidas() {
125 +         return Collections.unmodifiableList(transferenciasRecebidas);
126 +     }
127 +
128 +     public void setTransferenciasRecebidas(List<Transferencia>
transferenciasRecebidas) {
129 +         this.transferenciasRecebidas.clear();
130 +         if (transferenciasRecebidas != null) {
131 +             this.transferenciasRecebidas.addAll(transferenciasRecebidas);
132 +         }
133 +     }
134 +
135 +     public void removeTransferenciaRecebia(Transferencia transferencia){
136 +         this.transferenciasRecebidas.remove(transferencia);
137 +     }
138 +
139 +     @Override
140 +     public String toString() {
141 +         if (descricao != null) {
142 +             return sigla + " - " + descricao;
143 +         } else {
144 +             return sigla;
145 +         }
146 +     }
147 +
148 +     public String numero() {
149 +         return "";
150 +     }
151 +
152 +     @Override
153 +     public int hashCode() {
154 +         final int prime = 31;
155 +         int result = 1;
```

```

156 +         result = prime * result + ((idConta == null) ? 0 : idConta.hashCode());
157 +         return result;
158 +     }
159 +
160 +     @Override
161 +     public boolean equals(Object obj) {
162 +         if (this == obj) {
163 +             return true;
164 +         }
165 +         if (obj == null) {
166 +             return false;
167 +         }
168 +         if (getClass() != obj.getClass()) {
169 +             return false;
170 +         }
171 +         Conta other = (Conta) obj;
172 +         if (idConta == null) {
173 +             if (other.idConta != null) {
174 +                 return false;
175 +             }
176 +         } else if (!idConta.equals(other.idConta)) {
177 +             return false;
178 +         }
179 +         return true;
180 +     }
181 +
182 +     @Override
183 +     public int compareTo(Conta o) {
184 +         return this.sigla.compareTo(o.getSigla());
185 +     }
171 186 }

```

▼ 166 src/main/java/app/condominio/domain/ContaBancaria.java

```

18 18 @PrimaryKeyJoinColumn(name = "idconta")
19 19 public class ContaBancaria extends Conta {
20 20
21 - @NotNull
22 - @Enumerated(EnumType.STRING)
23 - private TipoContaBancaria tipo;
24 -
25 - @NotBlank
26 - @Size(max = 3)
27 - private String banco;
28 -
29 - @NotBlank
30 - @Size(max = 5)
31 - private String agencia;
32 -
33 - @Column(name = "agenciadv")
34 - private Character agenciaDv;
35 -

```

```
36 - @NotBlank
37 - @Size(max = 20)
38 - private String conta;
39 -
40 - @Column(name = "contadv")
41 - private Character contaDv;
42 -
43 - public TipoContaBancaria getTipo() {
44 -     return tipo;
45 - }
46 -
47 - public void setTipo(TipoContaBancaria tipo) {
48 -     this.tipo = tipo;
49 - }
50 -
51 - public String getBanco() {
52 -     return banco;
53 - }
54 -
55 - public void setBanco(String banco) {
56 -     this.banco = banco;
57 - }
58 -
59 - public String getAgencia() {
60 -     return agencia;
61 - }
62 -
63 - public void setAgencia(String agencia) {
64 -     this.agencia = agencia;
65 - }
66 -
67 - public Character getAgenciaDv() {
68 -     return agenciaDv;
69 - }
70 -
71 - public void setAgenciaDv(Character agenciaDv) {
72 -     this.agenciaDv = agenciaDv;
73 - }
74 -
75 - public String getConta() {
76 -     return conta;
77 - }
78 -
79 - public void setConta(String conta) {
80 -     this.conta = conta;
81 - }
82 -
83 - public Character getContaDv() {
84 -     return contaDv;
85 - }
86 -
87 - public void setContaDv(Character contaDv) {
```

```

88 - _____ this.contaDv = contaDv;
89 - _____}
90 -
91 - _____@Override
92 - _____public String numero() {
93 - _____String s = banco + " " + agencia;
94 - _____if (agenciaDv != null) {
95 - _____s += "-" + agenciaDv;
96 - _____}
97 - _____s += " " + conta;
98 - _____if (contaDv != null) {
99 - _____s += "-" + contaDv;
100 - _____}
101 - _____return s;
102 - _____}
103 -
21 + _____@NotNull
22 + _____@Enumerated(EnumType.STRING)
23 + _____private TipoContaBancaria tipo;
24 +
25 + _____@NotBlank
26 + _____@Size(max = 3)
27 + _____private String banco;
28 +
29 + _____@NotBlank
30 + _____@Size(max = 5)
31 + _____private String agencia;
32 +
33 + _____@Column(name = "agenciadv")
34 + _____private Character agenciaDv;
35 +
36 + _____@NotBlank
37 + _____@Size(max = 20)
38 + _____private String conta;
39 +
40 + _____@Column(name = "contadv")
41 + _____private Character contaDv;
42 +
43 + _____public TipoContaBancaria getTipo() {
44 + _____return tipo;
45 + _____}
46 +
47 + _____public void setTipo(TipoContaBancaria tipo) {
48 + _____this.tipo = tipo;
49 + _____}
50 +
51 + _____public String getBanco() {
52 + _____return banco;
53 + _____}
54 +
55 + _____public void setBanco(String banco) {
56 + _____this.banco = banco;

```

```
57 + _____}  
58 +  
59 + _____public String getAgencia() {  
60 + _____return agencia;  
61 + _____}  
62 +  
63 + _____public void setAgencia(String agencia) {  
64 + _____this.agencia = agencia;  
65 + _____}  
66 +  
67 + _____public Character getAgenciaDv() {  
68 + _____return agenciaDv;  
69 + _____}  
70 +  
71 + _____public void setAgenciaDv(Character agenciaDv) {  
72 + _____this.agenciaDv = agenciaDv;  
73 + _____}  
74 +  
75 + _____public String getConta() {  
76 + _____return conta;  
77 + _____}  
78 +  
79 + _____public void setConta(String conta) {  
80 + _____this.conta = conta;  
81 + _____}  
82 +  
83 + _____public Character getContaDv() {  
84 + _____return contaDv;  
85 + _____}  
86 +  
87 + _____public void setContaDv(Character contaDv) {  
88 + _____this.contaDv = contaDv;  
89 + _____}  
90 +  
91 + _____@Override  
92 + _____public String numero() {  
93 + _____StringBuilder sb = new StringBuilder();  
94 + _____sb.append(banco).append(" ").append(agencia);  
95 + _____if (agenciaDv != null) {  
96 + _____sb.append("-").append(agenciaDv);  
97 + _____}  
98 + _____sb.append(" ").append(conta);  
99 + _____if (contaDv != null) {  
100 + _____sb.append("-").append(contaDv);  
101 + _____}  
102 + _____return sb.toString();  
103 + _____}  
104 104 }
```

104

72 ■■■ src/main/java/app/condominio/domain/Lancamento.java

8

8

import javax.persistence.Table;

```

9      9      import javax.validation.constraints.NotNull;
10     10
11     11     - @SuppressWarnings("serial")
12     11     +
13     12     @Entity
14     13     @Table(name = "lancamentos")
15     14     @PrimaryKeyJoinColumn(name = "idmovimento")
16     15     public class Lancamento extends Movimento {
17     16
18     17     - @ManyToOne(fetch = FetchType.LAZY)
19     18     - @JoinColumn(name = "idperiodo")
20     19     - private Periodo periodo;
21     20
22     21     - @NotNull
23     22     - @ManyToOne(fetch = FetchType.LAZY)
24     23     - @JoinColumn(name = "idsubcategoria")
25     24     - private Subcategoria subcategoria;
26     25
27     26     - public Periodo getPeriodo() {
28     27     -         return periodo;
29     28     -     }
30     29
31     30     - public void setPeriodo(Periodo periodo) {
32     31     -         this.periodo = periodo;
33     32     -     }
34     33
35     34     - public Subcategoria getSubcategoria() {
36     35     -         return subcategoria;
37     36     -     }
38     37
39     38     - public void setSubcategoria(Subcategoria subcategoria) {
40     39     -         this.subcategoria = subcategoria;
41     40     -     }
42     41
43     42     - @Override
44     43     - public String detalhe() {
45     44     -         String s;
46     45     -         if (getReducao()) {
47     46     -             s = "Despesa com ";
48     47     -         } else {
49     48     -             s = "Receita de ";
50     49     -         }
51     50     -         return s + subcategoria.getDescricao().toLowerCase();
52     51     -     }
53     52
54     53     + @ManyToOne(fetch = FetchType.LAZY)
55     54     + @JoinColumn(name = "idperiodo")
56     55     + private Periodo periodo;
57     56
58     57     + @NotNull
59     58     + @ManyToOne(fetch = FetchType.LAZY)
60     59     + @JoinColumn(name = "idsubcategoria")
61     60     + private Subcategoria subcategoria;

```

```

25 +
26 + public Periodo getPeriodo() {
27 + return periodo;
28 + }
29 +
30 + public void setPeriodo(Periodo periodo) {
31 + this.periodo = periodo;
32 + }
33 +
34 + public Subcategoria getSubcategoria() {
35 + return subcategoria;
36 + }
37 +
38 + public void setSubcategoria(Subcategoria subcategoria) {
39 + this.subcategoria = subcategoria;
40 + }
41 +
42 + @Override
43 + public String detalhe() {
44 + String s;
45 + if (getReducao()) {
46 + s = "Despesa com ";
47 + } else {
48 + s = "Receita de ";
49 + }
50 + return s + subcategoria.getDescricao().toLowerCase();
51 + }
52 52
53 53 }

```

✓ 311 src/main/java/app/condominio/domain/Moradia.java

```

2 2
3 3 import java.io.Serializable;
4 4 import java.util.ArrayList;
5 5 + import java.util.Collections;
6 6 import java.util.List;
7 7
8 8 import javax.persistence.CascadeType;
32 33 @Table(name = "moradias")
33 34 public class Moradia implements Serializable, Comparable<Moradia> {
34 35
35 36 - @Id
36 37 - @GeneratedValue(strategy = GenerationType.IDENTITY)
37 38 - @Column(name = "idmoradia")
38 36 - private Long idMoradia;
39 37 + @Id
40 38 + @GeneratedValue(strategy = GenerationType.IDENTITY)
39 39 + @Column(name = "idmoradia")
40 40 + private Long idMoradia;
40 40 - @Size(min = 1, max = 10)

```

```

41 - @NotBlank
42 - private String sigla;
    41 + @Size(min = 1, max = 10)
    42 + @NotBlank
    43 + private String sigla;
43 44
44 - @NotNull
45 - @Enumerated(EnumType.STRING)
46 - private TipoMoradia tipo;
    45 + @NotNull
    46 + @Enumerated(EnumType.STRING)
    47 + private TipoMoradia tipo;
47 48
48 - private Float area;
    49 + private Float area;
49 50
50 - @Max(100)
51 - @Min(0)
52 - @Column(name = "fracaoideal")
53 - private Float fracaoIdeal;
    51 + @Max(100)
    52 + @Min(0)
    53 + @Column(name = "fracaoideal")
    54 + private Float fracaoIdeal;
54 55
55 - @Size(max = 30)
56 - private String matricula;
    56 + @Size(max = 30)
    57 + private String matricula;
57 58
58 - private Integer vagas;
    59 + private Integer vagas;
59 60
60 - @NotNull
61 - @ManyToOne(fetch = FetchType.LAZY)
62 - @JoinColumn(name = "idbloco")
63 - private Bloco bloco;
    61 + @NotNull
    62 + @ManyToOne(fetch = FetchType.LAZY)
    63 + @JoinColumn(name = "idbloco")
    64 + private Bloco bloco;
64 65
65 - @OneToMany(mappedBy = "moradia", fetch = FetchType.LAZY, cascade =
    CascadeType.ALL, orphanRemoval = true)
66 - @OrderBy(value = "dataEntrada")
67 - @Valid
68 - private List<Relacao> relacoes = new ArrayList<>();
69 -
70 - @OneToMany(mappedBy = "moradia", fetch = FetchType.LAZY, cascade =
    CascadeType.REMOVE, orphanRemoval = true)
71 - @OrderBy(value = "numero, parcela")
72 - private List<Cobranca> cobrancas = new ArrayList<>();

```



```
73 -  
74 -     public Long getIdMoradia() {  
75 -         return idMoradia;  
76 -     }  
77 -  
78 -     public void setIdMoradia(Long idMoradia) {  
79 -         this.idMoradia = idMoradia;  
80 -     }  
81 -  
82 -     public String getSigla() {  
83 -         return sigla;  
84 -     }  
85 -  
86 -     public void setSigla(String sigla) {  
87 -         this.sigla = sigla;  
88 -     }  
89 -  
90 -     public TipoMoradia getTipo() {  
91 -         return tipo;  
92 -     }  
93 -  
94 -     public void setTipo(TipoMoradia tipo) {  
95 -         this.tipo = tipo;  
96 -     }  
97 -  
98 -     public Float getArea() {  
99 -         return area;  
100 -     }  
101 -  
102 -     public void setArea(Float area) {  
103 -         this.area = area;  
104 -     }  
105 -  
106 -     public Float getFracaoIdeal() {  
107 -         return fracaoIdeal;  
108 -     }  
109 -  
110 -     public void setFracaoIdeal(Float fracaoIdeal) {  
111 -         this.fracaoIdeal = fracaoIdeal;  
112 -     }  
113 -  
114 -     public String getMatricula() {  
115 -         return matricula;  
116 -     }  
117 -  
118 -     public void setMatricula(String matricula) {  
119 -         this.matricula = matricula;  
120 -     }  
121 -  
122 -     public Integer getVagas() {  
123 -         return vagas;  
124 -     }
```

```
125 -
126 -     public void setVagas(Integer vagas) {
127 -         this.vagas = vagas;
128 -     }
129 -
130 -     public Bloco getBloco() {
131 -         return bloco;
132 -     }
133 -
134 -     public void setBloco(Bloco bloco) {
135 -         this.bloco = bloco;
136 -     }
137 -
138 -     public List<Relacao> getRelacoes() {
139 -         return relacoes;
140 -     }
141 -
142 -     public void setRelacoes(List<Relacao> relacoes) {
143 -         this.relacoes = relacoes;
144 -     }
145 -
146 -     public List<Cobranca> getCobrancas() {
147 -         return cobranças;
148 -     }
149 -
150 -     public void setCobrancas(List<Cobranca> cobranças) {
151 -         this.cobranças = cobranças;
152 -     }
153 -
154 -     @Override
155 -     public String toString() {
156 -         return bloco.toString() + " - " + sigla;
157 -     }
158 -
159 -     @Override
160 -     public int hashCode() {
161 -         final int prime = 31;
162 -         int result = 1;
163 -         result = prime * result + ((idMoradia == null) ? 0 :
idMoradia.hashCode());
164 -         return result;
165 -     }
166 -
167 -     @Override
168 -     public boolean equals(Object obj) {
169 -         if (this == obj) {
170 -             return true;
171 -         }
172 -         if (obj == null) {
173 -             return false;
174 -         }
175 -         if (getClass() != obj.getClass()) {
```

```
176 -             return false;
177 -         }
178 -         Moradia other = (Moradia) obj;
179 -         if (idMoradia == null) {
180 -             if (other.idMoradia != null) {
181 -                 return false;
182 -             }
183 -             } else if (!idMoradia.equals(other.idMoradia)) {
184 -                 return false;
185 -             }
186 -             return true;
187 -     }
188 -
189 -     @Override
190 -     public int compareTo(Moradia arg0) {
191 -         return this.toString().compareTo(arg0.toString());
192 -     }
66 +     @OneToMany(mappedBy = "moradia", fetch = FetchType.LAZY, cascade =
    CascadeType.ALL, orphanRemoval = true)
67 +     @OrderBy(value = "dataEntrada")
68 +     @Valid
69 +     private List<Relacao> relacoes = new ArrayList<>();
70 +
71 +     @OneToMany(mappedBy = "moradia", fetch = FetchType.LAZY, cascade =
    CascadeType.REMOVE, orphanRemoval = true)
72 +     @OrderBy(value = "numero, parcela")
73 +     private List<Cobranca> cobrancas = new ArrayList<>();
74 +
75 +     public Long getIdMoradia() {
76 +         return idMoradia;
77 +     }
78 +
79 +     public void setIdMoradia(Long idMoradia) {
80 +         this.idMoradia = idMoradia;
81 +     }
82 +
83 +     public String getSigla() {
84 +         return sigla;
85 +     }
86 +
87 +     public void setSigla(String sigla) {
88 +         this.sigla = sigla;
89 +     }
90 +
91 +     public TipoMoradia getTipo() {
92 +         return tipo;
93 +     }
94 +
95 +     public void setTipo(TipoMoradia tipo) {
96 +         this.tipo = tipo;
97 +     }
98 +
```

```

 99 +     public Float getArea() {
100 +         return area;
101 +     }
102 +
103 +     public void setArea(Float area) {
104 +         this.area = area;
105 +     }
106 +
107 +     public Float getFracaoIdeal() {
108 +         return fracaoIdeal;
109 +     }
110 +
111 +     public void setFracaoIdeal(Float fracaoIdeal) {
112 +         this.fracaoIdeal = fracaoIdeal;
113 +     }
114 +
115 +     public String getMatricula() {
116 +         return matricula;
117 +     }
118 +
119 +     public void setMatricula(String matricula) {
120 +         this.matricula = matricula;
121 +     }
122 +
123 +     public Integer getVagas() {
124 +         return vagas;
125 +     }
126 +
127 +     public void setVagas(Integer vagas) {
128 +         this.vagas = vagas;
129 +     }
130 +
131 +     public Bloco getBloco() {
132 +         return bloco;
133 +     }
134 +
135 +     public void setBloco(Bloco bloco) {
136 +         this.bloco = bloco;
137 +     }
138 +
139 +     public List<Relacao> getRelacoes() {
140 +         return Collections.unmodifiableList(relacoes);
141 +     }
142 +
143 +     public void setRelacoes(List<Relacao> relacoes) {
144 +         this.relacoes.clear();
145 +         this.relacoes.addAll(relacoes);
146 +     }
147 +
148 +     public void removeRelacao(Relacao relacao) {
149 +         this.relacoes.remove(relacao);
150 +     }
```

```
151 +
152 +     public List<Cobranca> getCobrancas() {
153 +         return Collections.unmodifiableList(cobrancas);
154 +     }
155 +
156 +     public void setCobrancas(List<Cobranca> cobranças) {
157 +         this.cobrancas.clear();
158 +         this.cobrancas.addAll(cobrancas);
159 +     }
160 +
161 +     public void removeCobranca(Cobranca cobranca) {
162 +         this.cobrancas.remove(cobranca);
163 +     }
164 +
165 +     @Override
166 +     public String toString() {
167 +         return bloco.toString() + " - " + sigla;
168 +     }
169 +
170 +     @Override
171 +     public int hashCode() {
172 +         final int prime = 31;
173 +         int result = 1;
174 +         result = prime * result + ((idMoradia == null) ? 0 : idMoradia.hashCode());
175 +         return result;
176 +     }
177 +
178 +     @Override
179 +     public boolean equals(Object obj) {
180 +         if (this == obj) {
181 +             return true;
182 +         }
183 +         if (obj == null) {
184 +             return false;
185 +         }
186 +         if (getClass() != obj.getClass()) {
187 +             return false;
188 +         }
189 +         Moradia other = (Moradia) obj;
190 +         if (idMoradia == null) {
191 +             if (other.idMoradia != null) {
192 +                 return false;
193 +             }
194 +         } else if (!idMoradia.equals(other.idMoradia)) {
195 +             return false;
196 +         }
197 +         return true;
198 +     }
199 +
200 +     @Override
201 +     public int compareTo(Moradia other) {
202 +         return this.toString().compareTo(other.toString());
```

```

    203 +     }
193    204 }

```

70 src/main/java/app/condominio/domain/Movimento.java

```

28    28    @Inheritance(strategy = InheritanceType.JOINED)
29    29    public class Movimento implements Serializable {
30    30
31    31    -     @Id
32    32    -     @GeneratedValue(strategy = GenerationType.IDENTITY)
33    33    -     @Column(name = "idmovimento")
34    34    -     private Long idMovimento;
35    35
36    36    +     @Id
37    37    +     @GeneratedValue(strategy = GenerationType.IDENTITY)
38    38    +     @Column(name = "idmovimento")
39    39    +     private Long idMovimento;
40    40
41    41    -     @NotNull
42    42    -     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
43    43    -     private LocalDate data;
44    44    +     @NotNull
45    45    +     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
46    46    +     private LocalDate data;
47    47
48    48    -     @NotNull
49    49    -     @Min(0)
50    50    -     private BigDecimal valor;
51    51    +     @NotNull
52    52    +     @Min(0)
53    53    +     private BigDecimal valor;
54    54
55    55    -     @Size(max = 20)
56    56    -     private String documento;
57    57    +     @Size(max = 20)
58    58    +     private String documento;
59    59
60    60    -     @Size(max = 255)
61    61    -     private String descricao;
62    62    +     @Size(max = 255)
63    63    +     private String descricao;
64    64
65    65    -     private Boolean reducao;
66    66    +     private Boolean reducao;
67    67
68    68    -     @NotNull
69    69    -     @ManyToOne(fetch = FetchType.LAZY)
70    70    -     @JoinColumn(name = "idconta")
71    71    -     private Conta conta;
72    72    +     @NotNull
73    73    +     @ManyToOne(fetch = FetchType.LAZY)
74    74    +     @JoinColumn(name = "idconta")
75    75    +     private Conta conta;

```

```

56      56
57      57      public Long getIdMovimento() {
58      58          return idMovimento;
110     110          this.conta = conta;
111     111      }
112     112
113     -      public String detalhe() {
114     -          if (reducao) {
115     -              return "Saída";
116     -          } else {
117     -              return "Entrada";
118     -          }
119     -      }
120     -
121     -      @Override
122     -      public String toString() {
123     -          DateTimeFormatter formato =
124     -              DateTimeFormatter.ofPattern("dd/MM/yyyy");
125     -          String s = data.format(formato) + " - ";
126     -          if (documento != null) {
127     -              s += documento + " - ";
128     -          }
129     -          s += "R$ " + valor;
130     -          return s;
131     -      }
132     +      public String detalhe() {
133     +          return reducao ? "Saída" : "Entrada";
134     +      }
135     +
136     +      @Override
137     +      public String toString() {
138     +          DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy");
139     +          String s = data.format(formato) + " - ";
140     +          if (documento != null) {
141     +              s += documento + " - ";
142     +          }
143     +          s += "R$ " + valor;
144     +          return s;
145     +      }
146     +
147     +      @Override
148     +      public int hashCode() {

```

47 src/main/java/app/condominio/domain/Orcamento.java

```

4      4      import java.math.BigDecimal;
5      5
6      6      import javax.persistence.Column;
7      7      + import java.util.Objects;
8      8      import javax.persistence.Entity;
9      9      import javax.persistence.FetchType;
10     10     import javax.persistence.GeneratedValue;

```

```

15 16 import javax.validation.constraints.Min;
16 17 import javax.validation.constraints.NotNull;
17 18
18 19 - @SuppressWarnings("serial")
19 20 +
20 21 @Entity
21 22 @Table(name = "orcamentos")
22 23 public class Orcamento implements Serializable {
23 24
24 25 - @Id
25 26 - @GeneratedValue(strategy = GenerationType.IDENTITY)
26 27 - @Column(name = "idorcamento")
27 28 - private Long idOrcamento;
28 29 + @Id
29 30 + @GeneratedValue(strategy = GenerationType.IDENTITY)
30 31 + @Column(name = "idorcamento")
31 32 + private Long idOrcamento;
32 33
33 34 - @NotNull
34 35 - @ManyToOne(fetch = FetchType.LAZY)
35 36 - @JoinColumn(name = "idperiodo")
36 37 - private Periodo periodo;
37 38 + @NotNull
38 39 + @ManyToOne(fetch = FetchType.LAZY)
39 40 + @JoinColumn(name = "idperiodo")
40 41 + private Periodo periodo;
41 42
42 43 - @NotNull
43 44 - @ManyToOne(fetch = FetchType.LAZY)
44 45 - @JoinColumn(name = "idsubcategoria")
45 46 - private Subcategoria subcategoria;
46 47 + @NotNull
47 48 + @ManyToOne(fetch = FetchType.LAZY)
48 49 + @JoinColumn(name = "idsubcategoria")
49 50 + private Subcategoria subcategoria;
50 51
51 52 - @NotNull
52 53 - @Min(0)
53 54 - private BigDecimal orcado;
54 55 + @NotNull
55 56 + @Min(0)
56 57 + private BigDecimal orcado;
57 58
58 59 public Long getIdOrcamento() {
59 60     return idOrcamento;
60 61     if (this == obj) {
61 62         return true;
62 63     }
63 64     if (obj == null) {
64 65         return false;
65 66     }
66 67     if (getClass() != obj.getClass()) {

```



```

93 +         if (obj == null || getClass() != obj.getClass()) {
96 94             return false;
97 95         }
98 96         Orcamento other = (Orcamento) obj;
99 -         if (idOrcamento == null) {
100 -             if (other.idOrcamento != null) {
101 -                 return false;
102 -             }
103 -         } else if (!idOrcamento.equals(other.idOrcamento)) {
104 -             return false;
105 -         }
106 -         return true;
107 97 +         return Objects.equals(idOrcamento, other.idOrcamento);
108 98     }
109 99 }
110 100 }

```

✓ ↕ 16 ■■■■ src/main/java/app/condominio/domain/Periodo.java

```

5 5 import java.time.format.DateTimeFormatter;
6 6 import java.util.ArrayList;
7 7 import java.util.List;
8 8 + import java.util.Objects;
9 9
10 10 import javax.persistence.CascadeType;
11 11 import javax.persistence.Column;
22 23
23 24 import org.springframework.format.annotation.DateTimeFormat;
24 25
25 - @SuppressWarnings("serial")
26 26 @Entity
27 27 @Table(name = "periodos")
28 28 public class Periodo implements Serializable, Comparable<Periodo> {
128 128     if (this == obj) {
129 129         return true;
130 130     }
131 -     if (obj == null) {
132 -         return false;
133 -     }
134 -     if (getClass() != obj.getClass()) {
131 131 +     if (obj == null || getClass() != obj.getClass()) {
135 132         return false;
136 133     }
137 134     Periodo other = (Periodo) obj;
138 -     if (idPeriodo == null) {
139 -         if (other.idPeriodo != null) {
140 -             return false;
141 -         }
142 -     } else if (!idPeriodo.equals(other.idPeriodo)) {
143 -         return false;
144 -     }
145 -     return true;

```

```

135 + return Objects.equals(idPeriodo, other.idPeriodo);
146 }
147
148 @Override

```

27 src/main/java/app/condominio/domain/Pessoa.java

```

2 2
3 3 import java.io.Serializable;
4 4 import java.util.ArrayList;
5 5 + import java.util.Collections;
6 6 import java.util.List;
7 7 + import java.util.Objects;
8 8
9 9 import javax.persistence.CascadeType;
10 10 import javax.persistence.Column;
27 29
28 30 import app.condominio.domain.enums.Estado;
29 31
30 32 - @SuppressWarnings("serial")
31 33 +
32 34 @Entity
33 35 @Table(name = "pessoas")
34 36 @Inheritance(strategy = InheritanceType.JOINED)
189 191 }
190 192
191 193 public List<Relacao> getRelacoes() {
192 194 - return relacoes;
193 195 + return Collections.unmodifiableList(relacoes);
194 196 }
195 197
196 198 public void setRelacoes(List<Relacao> relacoes) {
197 199 - this.relacoes = relacoes;
198 200 + this.relacoes.clear();
199 201 + this.relacoes.addAll(relacoes);
200 202 }
201 203
202 204 public void removeRelacao(Relacao relacao){
203 205 + this.relacoes.remove(relacao);
204 206 }
205 207
206 208 @Override
218 225 if (this == obj) {
219 226 return true;
220 227 }
221 228 - if (obj == null) {
222 229 - return false;
223 230 }
224 231 - if (getClass() != obj.getClass()) {
225 232 + if (obj == null || getClass() != obj.getClass()) {
226 233 return false;
227 234 }

```

```

227      231      Pessoa other = (Pessoa) obj;
228      -          if (idPessoa == null) {
229      -              if (other.idPessoa != null) {
230      -                  return false;
231      -              }
232      -          } else if (!idPessoa.equals(other.idPessoa)) {
233      -              return false;
234      -          }
235      -          return true;
236      232      +          return Objects.equals(idPessoa, other.idPessoa);
237      233      }
238      234      @Override
239      235

```

162 src/main/java/app/condominio/domain/PessoaFisica.java

```

17      17      import app.condominio.domain.enums.Genero;
18      18      import app.condominio.domain.validators.CPF;
19      19
20      - @SuppressWarnings("serial")
21      20      @Entity
22      21      @Table(name = "pessoasfisicas")
23      22      @PrimaryKeyJoinColumn(name = "idpessoa")
24      23      public class PessoaFisica extends Pessoa {
25      24
26      -         @NotBlank
27      -         @Size(min = 1, max = 100)
28      -         private String sobrenome;
29      -
30      -         @CPF
31      -         private String cpf;
32      -
33      -         @Size(max = 14)
34      -         private String rg;
35      -
36      -         @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
37      -         private LocalDate nascimento;
38      -
39      -         @Enumerated(EnumType.STRING)
40      -         private Genero genero;
41      -
42      -         public String getSobrenome() {
43      -             return sobrenome;
44      -         }
45      -
46      -         public void setSobrenome(String sobrenome) {
47      -             this.sobrenome = sobrenome;
48      -         }
49      -
50      -         public String getCpf() {
51      -             return cpf;
52      -         }

```

```
53 -  
54 -     public void setCpf(String cpf) {  
55 -         this.cpf = cpf;  
56 -     }  
57 -  
58 -     public String getRg() {  
59 -         return rg;  
60 -     }  
61 -  
62 -     public void setRg(String rg) {  
63 -         this.rg = rg;  
64 -     }  
65 -  
66 -     public LocalDate getNascimento() {  
67 -         return nascimento;  
68 -     }  
69 -  
70 -     public void setNascimento(LocalDate nascimento) {  
71 -         this.nascimento = nascimento;  
72 -     }  
73 -  
74 -     public Genero getGenero() {  
75 -         return genero;  
76 -     }  
77 -  
78 -     public void setGenero(Genero genero) {  
79 -         this.genero = genero;  
80 -     }  
81 -  
82 -     @Override  
83 -     public String cpfCnpj() {  
84 -         if (cpf == null) {  
85 -             return null;  
86 -         } else {  
87 -             try {  
88 -                 MaskFormatter formatador = new  
MaskFormatter("###.###.###-##");  
89 -  
formatador.setValueContainsLiteralCharacters(false);  
90 -                 return formatador.valueToString(cpf);  
91 -             } catch (ParseException e) {  
92 -                 return cpf;  
93 -             }  
94 -         }  
95 -     }  
96 -  
97 -     @Override  
98 -     public String toString() {  
99 -         if (sobrenome != null) {  
100 -             return getNome() + " " + sobrenome;  
101 -         } else {  
102 -             return super.toString();
```

```
103     -         }
104     -
105     -     }
106     -
25     +     @NotBlank
26     +     @Size(min = 1, max = 100)
27     +     private String sobrenome;
28     +
29     +     @CPF
30     +     private String cpf;
31     +
32     +     @Size(max = 14)
33     +     private String rg;
34     +
35     +     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
36     +     private LocalDate nascimento;
37     +
38     +     @Enumerated(EnumType.STRING)
39     +     private Genero genero;
40     +
41     +     public String getSobrenome() {
42     +         return sobrenome;
43     +     }
44     +
45     +     public void setSobrenome(String sobrenome) {
46     +         this.sobrenome = sobrenome;
47     +     }
48     +
49     +     public String getCpf() {
50     +         return cpf;
51     +     }
52     +
53     +     public void setCpf(String cpf) {
54     +         this.cpf = cpf;
55     +     }
56     +
57     +     public String getRg() {
58     +         return rg;
59     +     }
60     +
61     +     public void setRg(String rg) {
62     +         this.rg = rg;
63     +     }
64     +
65     +     public LocalDate getNascimento() {
66     +         return nascimento;
67     +     }
68     +
69     +     public void setNascimento(LocalDate nascimento) {
70     +         this.nascimento = nascimento;
71     +     }
72     +
```

```

73 +     public Genero getGenero() {
74 +         return genero;
75 +     }
76 +
77 +     public void setGenero(Genero genero) {
78 +         this.genero = genero;
79 +     }
80 +
81 +     @Override
82 +     public String cpfCnpj() {
83 +         if (cpf == null) {
84 +             return null;
85 +         } else {
86 +             try {
87 +                 MaskFormatter formatador = new MaskFormatter("###.###.###-##");
88 +                 formatador.setValueContainsLiteralCharacters(false);
89 +                 return formatador.valueToString(cpf);
90 +             } catch (ParseException e) {
91 +                 return cpf;
92 +             }
93 +         }
94 +     }
95 +
96 +     @Override
97 +     public String toString() {
98 +         String nomeCompleto = super.toString();
99 +         if (sobrenome != null) {
100 +             return nomeCompleto + " " + sobrenome;
101 +         } else {
102 +             return nomeCompleto;
103 +         }
104 +     }
105 + }

```

107

62  src/main/java/app/condominio/domain/PessoaJuridica.java 

```

12 12
13 13 import app.condominio.domain.validators.CNPJ;
14 14
15 15 - @SuppressWarnings("serial")
16 16 +
17 16 @Entity
18 17 @Table(name = "pessoasjuridicas")
19 18 @PrimaryKeyJoinColumn(name = "idpessoa")
20 19 public class PessoaJuridica extends Pessoa {
21 20
22 21 - @NotBlank
23 22 - @Size(min = 1, max = 100)
24 23 - @Column(name = "razaosocial")
25 24 - private String razaoSocial;
26 21 + @NotBlank
27 22 + @Size(min = 1, max = 100)

```

```

23 + @Column(name = "razaosocial")
24 + private String razaoSocial;
25
26 - @CNPJ
27 - private String cnpj;
26 + @CNPJ
27 + private String cnpj;
28
29 - @Size(max = 14)
30 - private String ie;
29 + @Size(max = 14)
30 + private String ie;
31
32 - @Size(max = 30)
33 - private String im;
32 + @Size(max = 30)
33 + private String im;
34
35 - @NotBlank
36 - @Size(min = 1, max = 50)
37 - @Column(name = "nomerepresentante")
38 - private String nomeRepresentante;
35 + @NotBlank
36 + @Size(min = 1, max = 50)
37 + @Column(name = "nomerepresentante")
38 + private String nomeRepresentante;
39
40 - @NotBlank
41 - @Size(min = 1, max = 100)
42 - @Column(name = "sobrenomerepresentante")
43 - private String sobrenomeRepresentante;
40 + @NotBlank
41 + @Size(min = 1, max = 100)
42 + @Column(name = "sobrenomerepresentante")
43 + private String sobrenomeRepresentante;
44
45     public String getRazaoSocial() {
46         return razaoSocial;
90         this.sobrenomeRepresentante = sobrenomeRepresentante;
91     }
92
93 - @Override
94 - public String cpfCnpj() {
95 -     if (cnpj == null) {
96 -         return cnpj;
97 -     } else {
98 -         try {
99 -             MaskFormatter formatador = new
MaskFormatter("##.###.###/####-##");
100 -
formatador.setValueContainsLiteralCharacters(false);
101 -         return formatador.valueToString(cnpj);

```

```

102         } catch (ParseException e) {
103             return cnpj;
104         }
105     }
106 }
93  + @Override
94  + public String cpfCnpj() {
95  +     try {
96  +         MaskFormatter formatador = new MaskFormatter("###.###.###/####-##");
97  +         formatador.setValueContainsLiteralCharacters(false);
98  +         return formatador.valueToString(cnpj);
99  +     } catch (ParseException e) {
100  +         return cnpj != null ? cnpj : "";
101  +     }
102  + }
107 103
108 104 @Override
109 105 public String toString() {

```

342 src/main/java/app/condominio/domain/Relacao.java

```

29 29 @Table(name = "pessoa_moradia")
30 30 public class Relacao implements Serializable {
31 31
32     // Tutoriais:
33     // https://vladmihalcea.com/the-best-way-to-map-a-many-to-many-association-
    with-extra-columns-when-using-jpa-and-hibernate/
34     // https://www.thoughts-on-java.org/many-relationships-additional-
    properties/
35
36     @EmbeddedId
37     private IdRelacao idRelacao = new IdRelacao();
38
39     @ManyToOne(fetch = FetchType.LAZY)
40     @MapsId("idPessoa")
41     @JoinColumn(name = "idpessoa")
42     @Fetch(FetchMode.JOIN)
43     private Pessoa pessoa;
44
45     @ManyToOne(fetch = FetchType.LAZY)
46     @MapsId("idMoradia")
47     @JoinColumn(name = "idmoradia")
48     @Fetch(FetchMode.JOIN)
49     private Moradia moradia;
50
51     @NotNull
52     @Enumerated(EnumType.STRING)
53     private TipoRelacao tipo;
54
55     @NotNull
56     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
57     @Column(name = "dataentrada")

```



```
58     private LocalDate dataEntrada;
59
60     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
61     @Column(name = "datasaida")
62     private LocalDate dataSaida;
63
64     @Column(name = "participacaodono")
65     @Max(100)
66     @Min(0)
67     private Float participacaoDono;
68
69     public IdRelacao getIdRelacao() {
70         return idRelacao;
71     }
72
73     public void setIdRelacao(IdRelacao idRelacao) {
74         this.idRelacao = idRelacao;
75     }
76
77     public Pessoa getPessoa() {
78         return pessoa;
79     }
80
81     public void setPessoa(Pessoa pessoa) {
82         this.pessoa = pessoa;
83         if (pessoa != null) {
84             this.idRelacao.setIdPessoa(pessoa.getIdPessoa());
85         }
86     }
87
88     public Moradia getMoradia() {
89         return moradia;
90     }
91
92     public void setMoradia(Moradia moradia) {
93         this.moradia = moradia;
94         if (moradia != null) {
95             this.idRelacao.setIdMoradia(moradia.getIdMoradia());
96         }
97     }
98
99     public TipoRelacao getTipo() {
100         return tipo;
101     }
102
103     public void setTipo(TipoRelacao tipo) {
104         this.tipo = tipo;
105     }
106
107     public LocalDate getDataEntrada() {
108         return dataEntrada;
109     }
```

```
110 -
111 -     public void setDataEntrada(LocalDate dataEntrada) {
112 -         this.dataEntrada = dataEntrada;
113 -     }
114 -
115 -     public LocalDate getDataSaida() {
116 -         return dataSaida;
117 -     }
118 -
119 -     public void setDataSaida(LocalDate dataSaida) {
120 -         this.dataSaida = dataSaida;
121 -     }
122 -
123 -     public Float getParticipacaoDono() {
124 -         return participacaoDono;
125 -     }
126 -
127 -     public void setParticipacaoDono(Float participacaoDono) {
128 -         this.participacaoDono = participacaoDono;
129 -     }
130 -
131 -     @Override
132 -     public String toString() {
133 -         return pessoa.toString() + ", " + tipo.name().toLowerCase() + " de
134 -         " + moradia.toString();
135 -     }
136 -
137 -     @Override
138 -     public int hashCode() {
139 -         final int prime = 31;
140 -         int result = 1;
141 -         result = prime * result + ((idRelacao == null) ? 0 :
142 -         idRelacao.hashCode());
143 -         return result;
144 -     }
145 -
146 -     @Override
147 -     public boolean equals(Object obj) {
148 -         if (this == obj) {
149 -             return true;
150 -         }
151 -         if (obj == null) {
152 -             return false;
153 -         }
154 -         if (getClass() != obj.getClass()) {
155 -             return false;
156 -         }
157 -         Relacao other = (Relacao) obj;
158 -         if (idRelacao == null) {
159 -             if (other.idRelacao != null) {
160 -                 return false;
161 -             }
162 -         }
163 -         return true;
164 -     }
```

```
160         } else if (!idRelacao.equals(other.idRelacao)) {
161             return false;
162         }
163         return true;
164     }
165
166     @Embeddable
167     public static class IdRelacao implements Serializable {
168
169         @Column(name = "idpessoa")
170         private Long idPessoa;
171
172         @Column(name = "idmoradia")
173         private Long idMoradia;
174
175         public Long getIdPessoa() {
176             return idPessoa;
177         }
178
179         public void setIdPessoa(Long idPessoa) {
180             this.idPessoa = idPessoa;
181         }
182
183         public Long getIdMoradia() {
184             return idMoradia;
185         }
186
187         public void setIdMoradia(Long idMoradia) {
188             this.idMoradia = idMoradia;
189         }
190
191         @Override
192         public int hashCode() {
193             final int prime = 31;
194             int result = 1;
195             result = prime * result + ((idMoradia == null) ? 0 :
idMoradia.hashCode());
196             result = prime * result + ((idPessoa == null) ? 0 :
idPessoa.hashCode());
197             return result;
198         }
199
200         @Override
201         public boolean equals(Object obj) {
202             if (this == obj) {
203                 return true;
204             }
205             if (obj == null) {
206                 return false;
207             }
208             if (getClass() != obj.getClass()) {
209                 return false;
```

```

210         }
211         IdRelacao other = (IdRelacao) obj;
212         if (idMoradia == null) {
213             if (other.idMoradia != null) {
214                 return false;
215             }
216         } else if (!idMoradia.equals(other.idMoradia)) {
217             return false;
218         }
219         if (idPessoa == null) {
220             if (other.idPessoa != null) {
221                 return false;
222             }
223         } else if (!idPessoa.equals(other.idPessoa)) {
224             return false;
225         }
226         return true;
227     }
228
229     }
230
231     @EmbeddedId
232     private IdRelacao idRelacao = new IdRelacao();
233
234
235     @ManyToOne(fetch = FetchType.LAZY)
236     @MapsId("idPessoa")
237     @JoinColumn(name = "idpessoa")
238     @Fetch(FetchMode.JOIN)
239     private Pessoa pessoa;
240
241
242     @ManyToOne(fetch = FetchType.LAZY)
243     @MapsId("idMoradia")
244     @JoinColumn(name = "idmoradia")
245     @Fetch(FetchMode.JOIN)
246     private Moradia moradia;
247
248     @NotNull
249     @Enumerated(EnumType.STRING)
250     private TipoRelacao tipo;
251
252     @NotNull
253     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
254     @Column(name = "dataentrada")
255     private LocalDate dataEntrada;
256
257     @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
258     @Column(name = "datasaida")
259     private LocalDate dataSaida;
260
261     @Column(name = "participacaodono")
262     @Max(100)
263     @Min(0)
264     private Float participacaoDono;

```

```
64 +
65 +     public Relacao() {
66 +         // Construtor vazio
67 +     }
68 +
69 +     public Relacao(Pessoa pessoa, Moradia moradia, TipoRelacao tipo, LocalDate
        dataEntrada, LocalDate dataSaida, Float participacaoDono) {
70 +         this.pessoa = pessoa;
71 +         this.moradia = moradia;
72 +         this.tipo = tipo;
73 +         this.dataEntrada = dataEntrada;
74 +         this.dataSaida = dataSaida;
75 +         this.participacaoDono = participacaoDono;
76 +         this.idRelacao = new IdRelacao(pessoa.getIdPessoa(),
            moradia.getIdMoradia());
77 +     }
78 +
79 +     public IdRelacao getIdRelacao() {
80 +         return idRelacao;
81 +     }
82 +
83 +     public Pessoa getPessoa() {
84 +         return pessoa;
85 +     }
86 +
87 +     public void setPessoa(Pessoa pessoa) {
88 +         this.pessoa = pessoa;
89 +         if (pessoa != null) {
90 +             this.idRelacao.setIdPessoa(pessoa.getIdPessoa());
91 +         }
92 +     }
93 +
94 +     public Moradia getMoradia() {
95 +         return moradia;
96 +     }
97 +
98 +     public void setMoradia(Moradia moradia) {
99 +         this.moradia = moradia;
100 +         if (moradia != null) {
101 +             this.idRelacao.setIdMoradia(moradia.getIdMoradia());
102 +         }
103 +     }
104 +
105 +     public TipoRelacao getTipo() {
106 +         return tipo;
107 +     }
108 +
109 +     public void setTipo(TipoRelacao tipo) {
110 +         this.tipo = tipo;
111 +     }
112 +
113 +     public LocalDate getDataEntrada() {
```

```
114 +         return dataEntrada;
115 +     }
116 +
117 +     public void setDataEntrada(LocalDate dataEntrada) {
118 +         this.dataEntrada = dataEntrada;
119 +     }
120 +
121 +     public LocalDate getDataSaida() {
122 +         return dataSaida;
123 +     }
124 +
125 +     public void setDataSaida(LocalDate dataSaida) {
126 +         this.dataSaida = dataSaida;
127 +     }
128 +
129 +     public Float getParticipacaoDono() {
130 +         return participacaoDono;
131 +     }
132 +
133 +     public void setParticipacaoDono(Float participacaoDono) {
134 +         this.participacaoDono = participacaoDono;
135 +     }
136 +
137 +     @Override
138 +     public String toString() {
139 +         return pessoa.toString() + ", " + tipo.name().toLowerCase() + " de " +
moradia.toString();
140 +     }
141 +
142 +     @Embeddable
143 +     public static class IdRelacao implements Serializable {
144 +
145 +         @Column(name = "idpessoa")
146 +         private Long idPessoa;
147 +
148 +         @Column(name = "idmoradia")
149 +         private Long idMoradia;
150 +
151 +         public IdRelacao() {
152 +             // Construtor vazio
153 +         }
154 +
155 +         public IdRelacao(Long idPessoa, Long idMoradia) {
156 +             this.idPessoa = idPessoa;
157 +             this.idMoradia = idMoradia;
158 +         }
159 +
160 +         public Long getIdPessoa() {
161 +             return idPessoa;
162 +         }
163 +
164 +         public void setIdPessoa(Long idPessoa) {
```

```

165 +         this.idPessoa = idPessoa;
166 +     }
167 +
168 +     public Long getIdMoradia() {
169 +         return idMoradia;
170 +     }
171 +
172 +     public void setIdMoradia(Long idMoradia) {
173 +         this.idMoradia = idMoradia;
174 +     }
175 + }
230 176 }
```

58 src/main/java/app/condominio/domain/Transferencia.java

```

15 15 @PrimaryKeyJoinColumn(name = "idmovimento")
16 16 public class Transferencia extends Movimento {
17 17
18 - @NotNull
19 - @ManyToOne(fetch = FetchType.LAZY)
20 - @JoinColumn(name = "idcontainversa")
21 - private Conta contaInversa;
22 -
23 - @OneToOne(fetch = FetchType.LAZY)
24 - @JoinColumn(name = "idmovimentoinverso")
25 - private Movimento movimentoInverso;
26 -
27 - public Conta getContaInversa() {
28 -     return contaInversa;
29 - }
30 -
31 - public void setContaInversa(Conta contaInversa) {
32 -     this.contaInversa = contaInversa;
33 - }
34 -
35 - public Movimento getMovimentoInverso() {
36 -     return movimentoInverso;
37 - }
38 -
39 - public void setMovimentoInverso(Movimento movimentoInverso) {
40 -     this.movimentoInverso = movimentoInverso;
41 - }
42 -
43 - @Override
44 - public String detalhe() {
45 -     if (getReducao()) {
46 -         return "Transferência de saída";
47 -     } else {
48 -         return "Transferência de entrada";
49 -     }
50 - }
51 - }
```

```

18 + @NotNull
19 + @ManyToOne(fetch = FetchType.LAZY)
20 + @JoinColumn(name = "idcontadestino")
21 + private Conta contaDestino;
22 +
23 + @OneToOne(fetch = FetchType.LAZY)
24 + @JoinColumn(name = "idmovimentoinverso")
25 + private Movimento movimentoInverso;
26 +
27 + public Conta getContaDestino() {
28 +     return contaDestino;
29 + }
30 +
31 + public void setContaDestino(Conta contaDestino) {
32 +     this.contaDestino = contaDestino;
33 + }
34 +
35 + public Movimento getMovimentoInverso() {
36 +     return movimentoInverso;
37 + }
38 +
39 + public void setMovimentoInverso(Movimento movimentoInverso) {
40 +     this.movimentoInverso = movimentoInverso;
41 + }
52 42 }

```

▼ 309 src/main/java/app/condominio/domain/Usuario.java

```

...    ...    @@ -1,6 +1,7 @@
1      1      package app.condominio.domain;
2      2
3      3      import java.io.Serializable;
4      4      + import java.util.Collections;
5      5      import java.util.HashSet;
6      6      import java.util.Set;
7      7
30     31      @Table(name = "usuarios")
31     32      public class Usuario implements Serializable {
32     33
33     -      @Id
34     -      @GeneratedValue(strategy = GenerationType.IDENTITY)
35     -      private Long id;
36     -
37     -      @NotBlank
38     -      @Size(min = 1, max = 50)
39     -      private String username;
40     -
41     -      @NotBlank
42     -      @Size(min = 4, max = 100)
43     -      private String password;
44     -

```



```
45 - @NotNull
46 - private Boolean ativo;
47 -
48 - @NotBlank
49 - @Size(min = 1, max = 50)
50 - private String nome;
51 -
52 - @NotBlank
53 - @Size(min = 1, max = 100)
54 - private String sobrenome;
55 -
56 - @NotBlank
57 - @Size(min = 1, max = 100)
58 - @Email
59 - private String email;
60 - // LATER validar email ao criar conta
61 -
62 - @ElementCollection(targetClass = Autorizacao.class)
63 - @CollectionTable(name = "autorizacoes", joinColumns = @JoinColumn(name =
    "id_usuario"))
64 - @Enumerated(EnumType.STRING)
65 - @Column(name = "autorizacao")
66 - private Set<Autorizacao> autorizacoes = new HashSet<>();
67 -
68 - @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.REMOVE,
    orphanRemoval = true)
69 - @JoinColumn(name = "idcondominio")
70 - private Condominio condominio;
71 -
72 - public String nomeCompleto() {
73 -     if (sobrenome != null) {
74 -         return nome + " " + sobrenome;
75 -     } else {
76 -         return nome;
77 -     }
78 - }
79 -
80 - public Long getId() {
81 -     return id;
82 - }
83 -
84 - public void setId(Long id) {
85 -     this.id = id;
86 - }
87 -
88 - public String getUsername() {
89 -     return username;
90 - }
91 -
92 - public void setUsername(String username) {
93 -     this.username = username;
94 - }
```

```
95 -
96 -     public String getPassword() {
97 -         return password;
98 -     }
99 -
100 -     public void setPassword(String password) {
101 -         this.password = password;
102 -     }
103 -
104 -     public Boolean getAtivo() {
105 -         return ativo;
106 -     }
107 -
108 -     public void setAtivo(Boolean ativo) {
109 -         this.ativo = ativo;
110 -     }
111 -
112 -     public String getNome() {
113 -         return nome;
114 -     }
115 -
116 -     public void setNome(String nome) {
117 -         this.nome = nome;
118 -     }
119 -
120 -     public String getSobrenome() {
121 -         return sobrenome;
122 -     }
123 -
124 -     public void setSobrenome(String sobrenome) {
125 -         this.sobrenome = sobrenome;
126 -     }
127 -
128 -     public String getEmail() {
129 -         return email;
130 -     }
131 -
132 -     public void setEmail(String email) {
133 -         this.email = email;
134 -     }
135 -
136 -     public Set<Autorizacao> getAutorizacoes() {
137 -         return autorizacoes;
138 -     }
139 -
140 -     public void setAutorizacoes(Set<Autorizacao> autorizacoes) {
141 -         this.autorizacoes = autorizacoes;
142 -     }
143 -
144 -     public Condominio getCondominio() {
145 -         return condominio;
146 -     }
```

```
147 -
148 -     public void setCondominio(Condominio condominio) {
149 -         this.condominio = condominio;
150 -     }
151 -
152 -     @Override
153 -     public int hashCode() {
154 -         final int prime = 31;
155 -         int result = 1;
156 -         result = prime * result + ((id == null) ? 0 : id.hashCode());
157 -         return result;
158 -     }
159 -
160 -     @Override
161 -     public boolean equals(Object obj) {
162 -         if (this == obj) {
163 -             return true;
164 -         }
165 -         if (obj == null) {
166 -             return false;
167 -         }
168 -         if (getClass() != obj.getClass()) {
169 -             return false;
170 -         }
171 -         Usuario other = (Usuario) obj;
172 -         if (id == null) {
173 -             if (other.id != null) {
174 -                 return false;
175 -             }
176 -         } else if (!id.equals(other.id)) {
177 -             return false;
178 -         }
179 -         return true;
180 -     }
181 -
34 +     @Id
35 +     @GeneratedValue(strategy = GenerationType.IDENTITY)
36 +     private Long id;
37 +
38 +     @NotBlank
39 +     @Size(min = 1, max = 50)
40 +     private String username;
41 +
42 +     @NotBlank
43 +     @Size(min = 4, max = 100)
44 +     private String password;
45 +
46 +     @NotNull
47 +     private Boolean ativo;
48 +
49 +     @NotBlank
50 +     @Size(min = 1, max = 50)
```

```
51 +     private String nome;
52 +
53 +     @NotBlank
54 +     @Size(min = 1, max = 100)
55 +     private String sobrenome;
56 +
57 +     @NotBlank
58 +     @Size(min = 1, max = 100)
59 +     @Email
60 +     private String email;
61 +
62 +     @ElementCollection(targetClass = Autorizacao.class)
63 +     @CollectionTable(name = "autorizacoes", joinColumns = @JoinColumn(name =
        "id_usuario"))
64 +     @Enumerated(EnumType.STRING)
65 +     @Column(name = "autorizacao")
66 +     private Set<Autorizacao> autorizacoes = new HashSet<>();
67 +
68 +     @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.REMOVE, orphanRemoval =
        true)
69 +     @JoinColumn(name = "idcondominio")
70 +     private Condominio condominio;
71 +
72 +     public String nomeCompleto() {
73 +         if (sobrenome != null) {
74 +             return nome + " " + sobrenome;
75 +         } else {
76 +             return nome;
77 +         }
78 +     }
79 +
80 +     public Long getId() {
81 +         return id;
82 +     }
83 +
84 +     public void setId(Long id) {
85 +         this.id = id;
86 +     }
87 +
88 +     public String getUsername() {
89 +         return username;
90 +     }
91 +
92 +     public void setUsername(String username) {
93 +         this.username = username;
94 +     }
95 +
96 +     public String getPassword() {
97 +         return password;
98 +     }
99 +
100 +     public void setPassword(String password) {
```

```
101 +         this.password = password;
102 +     }
103 +
104 +     public Boolean getAtivo() {
105 +         return ativo;
106 +     }
107 +
108 +     public void setAtivo(Boolean ativo) {
109 +         this.ativo = ativo;
110 +     }
111 +
112 +     public String getNome() {
113 +         return nome;
114 +     }
115 +
116 +     public void setNome(String nome) {
117 +         this.nome = nome;
118 +     }
119 +
120 +     public String getSobrenome() {
121 +         return sobrenome;
122 +     }
123 +
124 +     public void setSobrenome(String sobrenome) {
125 +         this.sobrenome = sobrenome;
126 +     }
127 +
128 +     public String getEmail() {
129 +         return email;
130 +     }
131 +
132 +     public void setEmail(String email) {
133 +         this.email = email;
134 +     }
135 +
136 +     public Set<Autorizacao> getAutorizacoes() {
137 +         return Collections.unmodifiableSet(authorizacoes);
138 +     }
139 +
140 +     public void setAutorizacoes(Set<Autorizacao> autorizacoes) {
141 +         this.authorizacoes.clear();
142 +         this.authorizacoes.addAll(authorizacoes);
143 +     }
144 +     public void setAutorizacoes(Autorizacao autorizacoes) {
145 +         this.authorizacoes.add(authorizacoes);
146 +     }
147 +
148 +     public void addAutorizacao(Autorizacao autorizacao) {
149 +         this.authorizacoes.add(authorizacao);
150 +     }
151 +
152 +     public void removeAutorizacao(Autorizacao autorizacao) {
```

```

153 +         this.autorizacoes.remove(autorizacao);
154 +     }
155 +
156 +     public Condominio getCondominio() {
157 +         return condominio;
158 +     }
159 +
160 +     public void setCondominio(Condominio condominio) {
161 +         this.condominio = condominio;
162 +     }
163 +
164 +     @Override
165 +     public int hashCode() {
166 +         final int prime = 31;
167 +         int result = 1;
168 +         result = prime * result + ((id == null) ? 0 : id.hashCode());
169 +         return result;
170 +     }
171 +
172 +     @Override
173 +     public boolean equals(Object obj) {
174 +         if (this == obj) {
175 +             return true;
176 +         }
177 +         if (obj == null) {
178 +             return false;
179 +         }
180 +         if (getClass() != obj.getClass()) {
181 +             return false;
182 +         }
183 +         Usuario other = (Usuario) obj;
184 +         if (id == null) {
185 +             if (other.id != null) {
186 +                 return false;
187 +             }
188 +         } else if (!id.equals(other.id)) {
189 +             return false;
190 +         }
191 +         return true;
192 +     }
193 + }

```

4 src/main/java/app/condominio/service/CategoriaServiceImpl.java

```

120 120 // VALIDAÇÕES EM AMBOS
121 121 if (entidade.getCategoriaPai() != null) {
122 122 // Não pode criar mais níveis do que o parametrizado
123 - if (entidade.getCategoriaPai().getNivel() >=
124 - Categoria.NIVEL_MAX) {
125 -         validacao.rejectValue("categoriaPai", "Max", new
126 -         Object[] { 0, Categoria.NIVEL_MAX }, null);

```

```

123 +                                     if (entidade.getCategoriaPai().getNivel() >=
    Categoria.getMaxNivel()) {
124 +                                     validacao.rejectValue("categoriaPai", "Max", new
    Object[] { 0, Categoria.getMaxNivel() }, null);
125 125                                     }
126 126                                     // Não pode ter um tipo diferente do tipo do pai
127 127                                     if (entidade.getCategoriaPai().getTipo() !=
    entidade.getTipo()) {

```

4 src/main/java/app/condominio/service/CobrancaServiceImpl.java

```

186 186                                     entidade.setOutrosAcrescimos(BigDecimal.ZERO);
187 187                                     }
188 188                                     if (entidade.getPercentualJurosMes() == null) {
189 -                                     entidade.setPercentualJurosMes(new Float(0));
    189 +                                     entidade.setPercentualJurosMes((float) 0);
190 190                                     }
191 191                                     if (entidade.getPercentualMulta() == null) {
192 -                                     entidade.setPercentualMulta(new Float(0));
    192 +                                     entidade.setPercentualMulta((float) 0);
193 193                                     }
194 194                                     }
195 195

```

6 src/main/java/app/condominio/service/MoradiaServiceImpl.java

```

119 119                                     @Transactional(readOnly = true, propagation = Propagation.SUPPORTS)
120 120                                     public void padronizar(Moradia entidade) {
121 121                                     if (entidade.getFracaoIdeal() == null) {
122 -                                     entidade.setFracaoIdeal(new Float(0));
    122 +                                     entidade.setFracaoIdeal((float) 0);
123 123                                     }
124 124                                     if (entidade.getArea() == null) {
125 -                                     entidade.setArea(new Float(0));
    125 +                                     entidade.setArea((float) 0);
126 126                                     }
127 127                                     if (entidade.getVagas() == null) {
128 128                                     entidade.setVagas(0);
134 134                                     relacao.setMoradia(entidade);
135 135                                     }
136 136                                     if (relacao.getParticipacaoDono() == null) {
137 -                                     relacao.setParticipacaoDono(new Float(0));
    137 +                                     relacao.setParticipacaoDono((float) 0);
138 138                                     }
139 139                                     }
140 140

```

10 src/main/java/app/condominio/service/MovimentoServiceImpl.java

```

62 62                                     contrapartida.setValor(entidade.getValor());

```

```

63      63      contrapartida.setDocumento(entidade.getDocumento());
64      64
65      65      -      contrapartida.setConta(((Transferencia)
66      66      -      entidade).getContaInversa());
67      67      +      contrapartida.setContaInversa(entidade.getConta());
68      68      +      contrapartida.setConta(((Transferencia)
69      69      +      entidade).getContaDestino());
70      70      +      contrapartida.setContaDestino(entidade.getConta());
71      71      +      contrapartida.setReducao(Boolean.FALSE);
72      72      +      contrapartida.setMovimentoInverso(entidade);
73      73      +      ((Transferencia)
74      74      entidade).setMovimentoInverso(contrapartida);
75      75      +      ((Transferencia)
76      76      entidade).getMovimentoInverso().setValor(entidade.getValor());
77      77      +      ((Transferencia)
78      78      entidade).getMovimentoInverso().setDocumento(entidade.getDocumento());
79      79      +      ((Transferencia)
80      80      entidade).getMovimentoInverso().setDescricao(entidade.getDescricao());
81      81      -      ((Transferencia)
82      82      entidade).getMovimentoInverso().setConta(((Transferencia)
83      83      entidade).getContaInversa());
84      84      -      ((Transferencia) ((Transferencia)
85      85      entidade).getMovimentoInverso()).setContaInversa(entidade.getConta());
86      86      +      ((Transferencia)
87      87      entidade).getMovimentoInverso().setConta(((Transferencia)
88      88      entidade).getContaDestino());
89      89      +      ((Transferencia) ((Transferencia)
90      90      entidade).getMovimentoInverso()).setContaDestino(entidade.getConta());
91      91      +      ((Transferencia)
92      92      entidade).getMovimentoInverso().setReducao(!((Transferencia)
93      93      entidade).getReducao());
94      94      +      ((Transferencia) ((Transferencia)
95      95      entidade).getMovimentoInverso()).setMovimentoInverso(entidade);
96      96      +      listaSalvar.add(((Transferencia)
97      97      entidade).getMovimentoInverso());
98      98      +      }
99      99      +      // Não permitir transferência para conta igual
100     100     +      if (entidade.getConta() != null && entidade instanceof
101     101     +      Transferencia
102     102     +      && entidade.getConta().equals(((Transferencia)
103     103     +      entidade).getContaInversa())) {
104     104     +      && entidade.getConta().equals(((Transferencia)
105     105     +      entidade).getContaDestino())) {
106     106     +      validacao.rejectValue("contaInversa", "Conflito");
107     107     +      }
108     108     +      }
109     109     +      }

```

src/main/java/app/condominio/service/PessoaServiceImpl.java

160 160 relacao.setPessoa(entidade);

| | | | |
|-----|-----|---|--|
| 161 | 161 | | } |
| 162 | 162 | | if (relacao.getParticipacaoDono() == null) { |
| 163 | | - | relacao.setParticipacaoDono(new Float(0)); |
| | 163 | + | relacao.setParticipacaoDono((float) 0); |
| 164 | 164 | | } |
| 165 | 165 | | } |
| 166 | 166 | | |

7 src/main/java/app/condominio/service/UsuarioServiceImpl.java

| | | | |
|-----|-----|---|---|
| 83 | 83 | | |
| 84 | 84 | | @Override |
| 85 | 85 | | public void salvarSindico(Usuario usuario) { |
| 86 | | - | usuario.getAutorizacoes().add(Autorizacao.SINDICO); |
| | 86 | + | |
| | 87 | + | usuario.setAutorizacoes(Autorizacao.SINDICO); |
| 87 | 88 | | salvar(usuario); |
| 88 | 89 | | } |
| 89 | 90 | | |
| 90 | 91 | | @Override |
| 91 | 92 | | public void salvarCondmino(Usuario usuario) { |
| 92 | | - | usuario.getAutorizacoes().add(Autorizacao.CONDOMINO); |
| | 93 | + | usuario.setAutorizacoes(Autorizacao.CONDOMINO); |
| 93 | 94 | | salvar(usuario); |
| 94 | 95 | | } |
| 95 | 96 | | |
| 96 | 97 | | @Override |
| 97 | 98 | | public void salvarAdmin(Usuario usuario) { |
| 98 | | - | usuario.getAutorizacoes().add(Autorizacao.ADMIN); |
| | 99 | + | usuario.setAutorizacoes(Autorizacao.ADMIN); |
| 99 | 100 | | salvar(usuario); |
| 100 | 101 | | } |
| 101 | 102 | | |

0 comments on commit f454295

Please [sign in](#) to comment.