

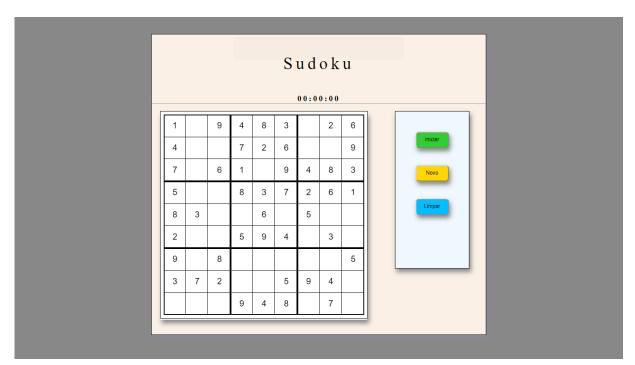


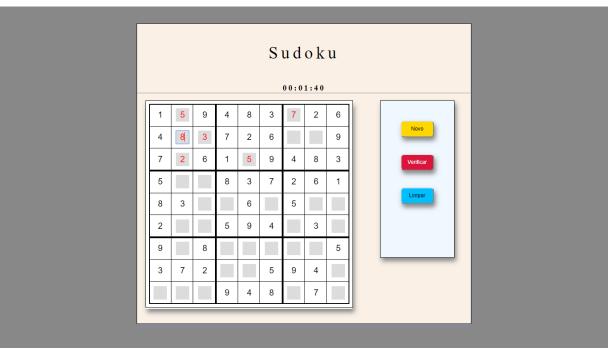


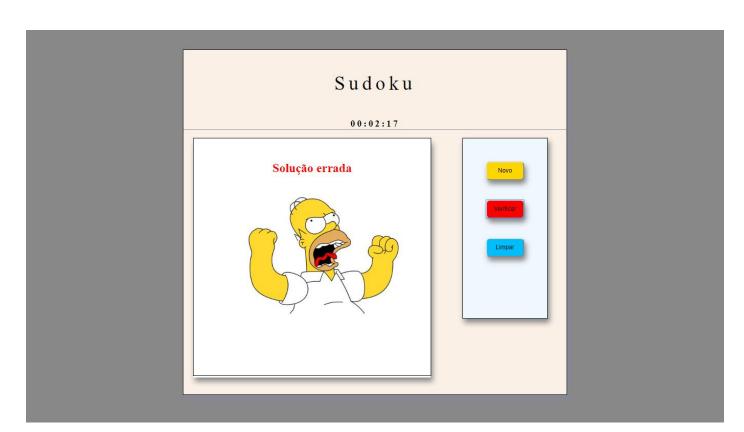
Avaliação a Distância – AD2 2018.2

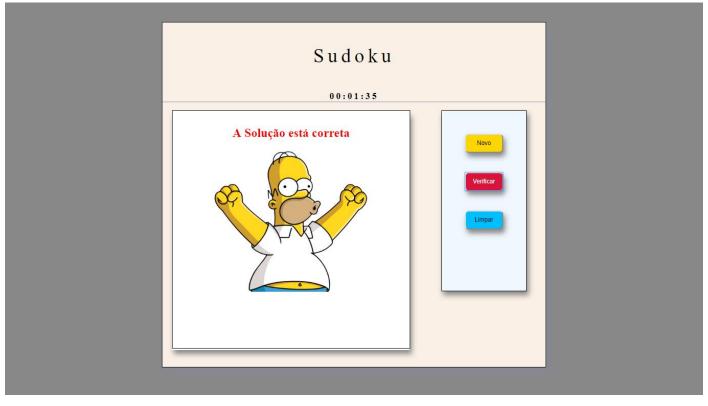
Programação de Aplicações Web

Aluno: Wellington Dias de oliveira









```
1
2
     <!DOCTYPE html>
3
     <html>
4
        <head>
5
             <title>Ad2_web</title>
 6
             <meta name="viewport" content="width=device-width, initial-scale=1.0">
 7
             <meta charset="utf-8">
8
             <script
                 src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></s</pre>
                 cript>
9
             <link rel='stylesheet' type='text/css' href='estilo.css'/>
10
        </head>
11
         <body onload="novoJogo()">
12
             <div id="container">
13
                 <header>
                     <div id='header'>Sudoku</div>
14
15
                 </header>
16
                 <div align= 'center 'id = 'counter'><h2>00:00:00</h2></div>
17
                 <hr>
18
                 <div id="teste">
19
                     <div id="area sodoku">
20
                     </div>
21
                 </div>
                 <div id="op">
22
23
                     <button id='btn start' type="button">Iniciar</button>
24
                     <button id='btn novo' type="button">Novo</button>
25
                     <button id='btn verificar' type="button">Verificar</button>
26
                     <button id='btn limpar' type="button">Limpar
27
                 </div>
             </div>
28
29
             <script>
30
    <?php
31
     include 'script.js';
32
33
             </script>
34
        </body>
35
     </html>
```

36

```
1
2
      * função que limpa todod os campos editavéis
 3
      * preenchidos pelo jogador
 4
5
     function limpar() {
 6
         var x = document.getElementsByClassName('c1');
7
         for (i = 0; i < x.length; i++) {</pre>
8
             x[i].value = x[i].value.innerHtml = '';
9
10
11
     }
12
13
      * função responsável por fazer a requisição
      * para um novo jogo.
14
15
     function novoJogo() {
16
         var page = "controle.php";
17
18
         $.ajax({
             type: 'POST',
19
20
             dataType: "html",
21
             url: page,
22
             beforeSend: function () {
23
                  $('#area sodoku').html('<img src="spinner.gif" id="spiner"/>');
24
             },
25
             data: {opt: 1},
26
             cache: false,
27
             success: function (msg) {
2.8
                  $("#area_sodoku").html(msg);
29
                  $('.c1').hide();
30
             }
31
         });
32
     }
33
34
35
      * função auxiliar que transforma um vetor em matriz,
36
      * uma vez que os elementos da classe 'campo'
37
      * estão em um array.
38
39
     function arrayParaMatriz(a) {
40
         vet = [];
41
         v = [];
         lin = 0;
42
43
         col = 0;
         for (i = 0; i <= a.length; i++) {</pre>
44
             if (i % 9 === 0 & i !== 0) {
45
46
                  vet[lin] = v;
47
                  v = [];
                  col = 0;
48
49
                  v[col] = parseInt(a[i]);
50
                  col++;
51
                  lin++;
52
              } else {
53
                  v[col] = parseInt(a[i]);
54
                  col++;
55
56
         }
57
         return vet;
58
     }
59
60
61
      * Função responsável por capturar o tabuleiro
62
      * do jogador, bem como o tempo paro o seu preenchimento
63
      * e realizar a requisição para verificação.
64
65
     function verificar() {
         var x = document.getElementsByClassName('campo');
66
67
         var vetor = [];
         for (i = 0; i < x.length; i++) {</pre>
68
             vetor[i] = x[i].value;
69
70
71
72
         vet = arrayParaMatriz(vetor);
```

```
73
          console.log(vet);
 74
          if (typeof interval !== 'undefined') {
 75
              clearInterval(interval);
 76
          }
 77
          var page = "controle.php";
 78
          tempo = document.getElementById('counter').textContent;
 79
          $.ajax({
 80
              type: 'POST',
 81
              dataType: "html",
 82
              url: page,
 83
              beforeSend: function () {
                   $('#area sodoku').html('<img src="spinner.gif"/>');
 84
 8.5
              },
 86
              data: {vet: vet, tempo: tempo, opt: 2},
 87
              cache: false,
 88
              success: function (msg) {
 89
                   $("#area sodoku").html(msg);
 90
 91
          });
 92
      }
 93
 94
 95
       * função auxiliar responsável
       * pela formatação do tempo
 96
 97
 98
      function formatatempo(hr, min, segs) {
 99
100
          if (hr < 10) {
101
              hr = '0' + hr;
102
          }
          if (min < 10) {</pre>
103
104
              min = '0' + min;
105
          }
106
          if (segs < 10) {
107
              segs = '0' + segs;
108
109
          fin = '<h2>' + hr + ':' + min + ':' + segs + '</h2>';
110
          return fin;
111
112
113
      // inicialização das variáveis de tempo
114
      var segundos = 0;
115
      var minutos = 0;
116
      var horas = 0;
117
118
       * função responsável pelo incremento
119
120
       * dos segundos, minutos e horas
121
       */
122
      function conta() {
123
          segundos++;
124
          if (segundos >= 60) {
125
              segundos = 0;
126
              minutos++;
127
128
          if (minutos \geq 60) {
129
              minutos = 0;
130
              horas++;
131
          }
132
          document.getElementById('counter').innerHTML = formatatempo(horas, minutos,
              segundos);
133
      }
134
135
136
       * Função que inicia o contador de tempo
       * chamando a função conta() em intervalos de 1 segundo
137
138
139
      function inicia() {
          //se o contador já foi inicializado será resetado
140
          if (typeof interval !== 'undefined') {
141
142
              horas = 0;
143
              minutos = 0;
```

```
144
              segundos = 0;
145
              clearInterval(interval);
146
              interval = setInterval("conta();", 1000);
147
148
              interval = setInterval("conta();", 1000);
149
150
      }
      /*
151
152
      * Função responsável por garantir
153
      * a validade da entrada passada pelo jogador
154
       * adimitindo somente números, e estes no intervalo
155
       * fechado de 1 até 9
       */
156
      function validaEntrada() {
157
158
          //função que garante a correção dos dados de entrada
159
          var x = document.getElementsByClassName("c1");
          var re = /^[1-9]$/;
160
161
          for (i = 0; i < x.length; i++) {</pre>
              if (!re.test(x[i].value)) {
162
163
                  x[i].value = x[i].value.innerHTML = '';
164
165
          }
166
      }
167
168
      /*----*/
169
170
      // botão criar novo jogo
171
172
      $("#btn novo").click(function () {
173
          $('#btn_verificar').css('display', 'none');
          if (typeof interval !== 'undefined') {
174
175
              clearInterval(interval);
176
          }
177
          novoJogo();
178
          $("#btn start").show();
179
180
      });
181
182
      // botão inicia o jogo
      $("#btn start").click(function () {
183
          $('.c1').css("background-color", '#DCDCDC');
184
          $('#btn verificar').css('display', 'block');
185
186
          $('.c1').show();
          $("#btn start").hide();
187
188
          inicia();
189
190
      });
191
192
      // Botão limpar tabuleiro
193
      $('#btn limpar').click(function () {
194
          limpar();
195
      });
196
197
      //botão avaliar
198
      $('#btn verificar').click(function () {
199
          verificar();
200
      }
201
      );
```

```
1
     <?php
2
3
4
      Classe responsável por gerar um tabuleiro de Sudoku válido,
5
      * criar as operações de inserção e consulta deste no banco de dados
      * garantir uma solução única e exibir ao jogador um tabuleiro jogável,
 6
 7
      * bem como guardar os melhores resultados
8
9
10
     class Controle {
11
12
         private $tabuleiro;
13
         private $cont = 0; //contador para o número de soluções possiveis
14
15
         public function construct() {
16
             $this->tabuleiro = $this->gerarSudoku();
17
18
19
20
          ^{\star} Este método varre a matriz e verifica se o tabuleiro
          * está completo, ou seja, sem zeros nas células
21
22
23
24
         function estaCheio($grid) {
25
             for (\$i = 0; \$i < 9; \$i++) {
                 for (\$j = 0; \$j < 9; \$j++) {
26
27
                      if ($grid[$i][$j] == 0) {
28
                          return false;
29
30
31
32
             return true;
33
34
35
         /* Verifica os candidatos à ocupar uma determinada
          * posição passada no tabuleiro. Inicialmente todos são
37
          * candidatos. Na medida que as verificações são feitas
38
          * vão sendo eliminados os que não atendem aos critérios
39
40
41
         function vetorDePossibilidades($grid, $lin, $col) {
42
             // array com todos os possiveis candidatos
43
             possibilidades = array(1, 2, 3, 4, 5, 6, 7, 8, 9);
44
             // verificação horizontal
45
             for ($i = 1; $i <= 9; $i++) {</pre>
                 for (\$c = 0; \$c < 9; \$c++) {
46
47
                      if ($grid[$lin][$c] == $i) {
48
                          possibilidades[$i - 1] = 0;
49
50
                  }
51
             // verificação vertical
52
53
             for ($i = 1; $i <= 9; $i++) {</pre>
                 for ($1 = 0; $1 < 9; $1++) {</pre>
54
55
                      if ($grid[$1][$col] == $i) {
56
                          possibilidades[$i - 1] = 0;
57
58
                  }
59
60
61
             $lr = (int) ($lin / 3);
             $cr = (int) ($col / 3);
62
63
64
             // verifica a região 3x3
65
             for (n = 1; n \le 9; n++) {
                  for ($i = ($lr * 3); $i < ($lr + 1) * 3; $i++) {</pre>
66
                      for ($j = $cr * 3; $j < ($cr + 1) * 3; $j++) {
67
68
                          if ($grid[$i][$j] == $n) {
                              possibilidades[n - 1] = 0;
69
70
```

```
71
 72
 73
              }return $possibilidades;
 74
          }
 75
 76
 77
           * O método recebe um tabuleiro parcialmente preenchido
 78
           * e o preenche com todas as soluções possíveis e
 79
           * estas vão sendo contabilizadas para ganrantir
 80
           * que o tabuleiro à ser exibido ao jogador tenha apenas
 81
           * uma solução única */
 82
 83
          function resolve($grid) {
 84
              // variaveis que quardam as coordenadas das células
 85
              // à serem preenchidas
 86
              \frac{1}{n} = 0;
 87
              column = 0;
 88
 89
              if ($this->estaCheio($grid)) {
 90
                  // Nesse ponto o tabuleiro passado foi resolvido
 91
                  // e o contador de soluções é incrementado
 92
                  $this->cont++;
 93
                  return;
 94
              } else {
 95
 96
                  //Busca a primeira célula vazia
 97
                  for (\$i = 0; \$i < 9; \$i++) {
 98
                      $achou = false;
 99
                      for (\$j = 0; \$j < 9; \$j++) {
100
                          if ($grid[$i][$j] == 0) {
101
                              $linha = $i;
102
                              column = j;
103
                              $achou = true;
104
                              break;
105
106
107
                      if ($achou) {
108
                          break;
109
110
111
                  // Após a célula ser encontrada, ela é passada para o método
                  // responsável por retornar os candidatos para aquela posição.
112
113
                  $possibilidades = $this->vetorDePossibilidades($grid, $linha, $coluna);
114
115
                  // Todas as possibilidades são verificadas recursivamente
116
                  foreach ($possibilidades as $p) {
117
                      if (!\$p == 0) {
118
                          $grid[$linha][$coluna] = $p;
119
                          $this->resolve($grid);
120
121
122
                  // Backtracking
123
              }$grid[$linha][$coluna] = 0;
124
          }
125
126
127
           * Método responsável por exibir visualmente
128
           * um grid ao jagador
129
130
131
          public function criarGrid() {
132
              for ($i = 1; $i <= 9; $i++) {
                  echo "";
133
134
                  for (\$k = 1; \$k \le 3; \$k++) {
                      echo "";
135
136
                      for ($j = 1; $j <= 3; $j++) {</pre>
                          echo" ";
137
138
139
                      echo "";
140
```

```
echo"";
141
142
              }
143
          }
144
145
146
           * Metodo responsável por armazenar o tabuleiro gerado
147
           * no banco de dados
148
149
150
          public function guardarTabela($t) {
151
              $link = $this->getConexao();
152
              $tab = json encode($t);
153
              $sql = "INSERT INTO tabuleiros(matriz) VALUES('$tab')";
154
              $result = mysqli_query($link, $sql);
155
              if (!$result) {
156
                  die('erro' . mysqli . error($link));
157
158
              mysqli close($link);
159
          }
160
161
           * Método responsável por verificar
162
163
           * se o tabuleiro preenchido pelo jogador
164
           * encontra correspondência no banco de dados
165
           * e garantir que novos tabuleiros gerados não
           * se repitam no BD
166
167
168
169
          public function existeNoBD($t) {
170
              $link = $this->getConexao();
171
              $mat = json encode($t);
              $sql = "SELECT *FROM tabuleiros WHERE matriz = '$mat' ";
172
173
              $result = mysqli_query($link, $sql);
174
175
              $n tab = mysqli num rows($result);
176
              mysqli close($link);
177
              if ($n tab > 0) {
178
                  return true;
179
              } else {
180
                  return false;
181
182
          }
183
184
185
           * Este é o método que gera os Sudokos válidos,
186
           * podendo gerar 6 tipos diferentes de tabuleiros completos
187
188
189
          private function gerarSudoku() {
              n = 3;
190
191
              sop = rand(1, 8); // opção entre um dos seis tabuleiros possíveis
              $mat = array(); // inicio da construção da matriz
192
193
              // a opção não pode ser um multiplo de 3
194
              c = pop % 3 != 0 ? pop : 1;
195
              for ($i = 0; $i < 9; $i++) {</pre>
                  $new = array();
196
197
                  for (\$j = 0; \$j < 9; \$j++) {
198
                       v = (\sin * n + (int) (\sin / n) + \sin * c % 9 + 1;
199
                       new[] = (string) v;
200
201
                  mat[] = new;
202
              }
203
204
              // Se o tabuleiro gerado não existir no banco de dados,
205
              // ele será armazenado. Como o algoritmo
206
              // gera apenas seis tipos diferentes de tabuleiros,
207
              // só existirão seis no banco.
208
              if (!$this->existeNoBD($mat)) {
209
                  $this->guardarTabela($mat);
210
```

```
211
             return $mat;
212
          }
213
214
215
          * Método responsável por exibir ao jogador uma
          * mescla de números e campos vázios editáveis
216
217
218
219
         public function exibirSudoku() {
220
             echo "<div id = 'tab'>
221
         height: 600px;'>";
222
223
              // Enquanto a solução se mantém única
224
              // uma célula é marcada aleatoriamente com zero.
225
             while ($this->cont <= 1) {</pre>
226
                 $i = rand(0, 8);
227
                 $j = rand(0, 8);
228
                 $n = $this->tabuleiro[$i][$j];
229
                 $this->tabuleiro[$i][$j] = 0;
230
                 $this->resolve($this->tabuleiro);
231
                 if ($this->cont > 1) {
232
                     $this->tabuleiro[$i][$j] = $n;
233
                  } else {
234
                     this->cont = 0;
235
236
             }
237
238
              // As células marcadas com zero serão os campos vázios editáveis
239
             for (\$i = 0; \$i < 9; \$i++) {
                 echo "";
240
241
                  for (\$j = 0; \$j < 9; \$j++) {
                     $p = $this->tabuleiro[$i][$j];
242
243
                     if (p == 0) {
                         echo '<input type="text" class ="campo c1"
244
                             onClick= "validaEntrada()">';
245
                      } else {
                         echo '<input type="text" class="campo" value=' .</pre>
246
                             $p . ' readonly>';
247
248
249
                 echo "";
250
251
             echo"</div>";
252
         }
253
254
255
           * Metodo responsável por fazer a conexão com
256
          * o Banco de Dados.
257
258
259
         private function getConexao() {
             $servidor = 'localhost';
260
             $usuario = 'root';
261
             $senha = '';
262
263
             $banco = 'db sudoku';
264
              $con = mysqli connect($servidor, $usuario, $senha, $banco)
265
                     or die('Não foi possivel conectar' . mysqli error($con));
266
             return $con;
267
          }
268
269
270
          ^{\star} Este método tem por finalidade fazer a conexao com o BD
271
          * para avaliar se o tabuleiro
272
          * preenchido pelo jogador existe no banco e,
273
          ^{\star} desta forma, verificar se sua resposta
274
           * está correta ou não e se os seu tempo é o melhor para
275
           * aquele tabuleiro em caso de acerto.
           * /
276
277
278
         public function verificaTabJogador($t, $tempo) {
```

controle.php

```
279
              $link = $this->getConexao();
280
              if ($this->existeNoBD($t)) {
281
                  $mat = json_encode($t);
282
                  $result = mysqli query($link, "SELECT tempo FROM tabuleiros WHERE matriz
                      = '$mat' ");
283
                  $res = mysqli_fetch_array($result);
                  if (!$res['tempo']) {
284
285
                      $recorde = date('H:i:s', $tempo);
                      mysqli_query($link, "UPDATE tabuleiros SET tempo ='$recorde' WHERE
286
                          matriz = '$mat'");
287
                      echo"<h1><center style='margin:20px auto;color:red'>Melhor tempo:
                          $recorde</center></h1>";
288
                  } else if ($tempo < strtotime($res['tempo'])) {</pre>
289
                      $recorde = date('H:i:s', $tempo);
                      mysqli_query($link, "UPDATE tabuleiros SET tempo ='$recorde' WHERE
290
                          matriz = '$mat'");
291
                      echo"<h2><center style='margin:20px auto;>Novo recorde :
                          $recorde</center></h2>";
292
293
                  echo"<h1><center style='margin:30px auto;color:red'>A Solução está
                      correta</center></h1>";
294
                  echo"<img style='margin-left:100px;' src='acertou.png'/>";
295
                  echo"<h1><center style='margin:50px auto;color:red'>Solução
296
                      errada</center></h1>";
297
                  echo"<img style='margin-left:100px;' src='errou.jpg'/>";
298
299
              mysqli close($link);
300
301
302
303
304
      //---- main() -----
305
306
307
      $getparam = filter input array(INPUT POST, FILTER DEFAULT);
308
309
      // O jogađor optou por avaliar seu jogo ou por começar um novo
310
      $option = $getparam['opt'];
311
      $t = new Controle();
312
      switch ($option) {
313
          case 1: // começar um novo jogo
314
              $t->criarGrid();
315
              $t->exibirSudoku();
316
              break;
317
          case 2: // submeter sua resposta a correção
              if (isset($qetparam['vet'])) {
318
                  $tabela = $getparam['vet'];
319
                  $tempo = $getparam['tempo'];
320
321
                  $time = strtotime($tempo);
322
323
                  $t->verificaTabJogador($tabela, $time);
324
              } else {
325
                  echo "<center style='margin:200px'>Inicie um novo jogo</center>";
326
327
              break;
328
```