



Avaliação a Distância – AD2 2018.2

Programação de Aplicações Web

Aluno: Wellington Dias de oliveira

Sudoku

00:00:00

1		9	4	8	3		2	6
4			7	2	6			9
7		6	1		9	4	8	3
5			8	3	7	2	6	1
8	3			6		5		
2			5	9	4		3	
9		8						5
3	7	2			5	9	4	
			9	4	8		7	

Iniciar

Novo

Limpar

Sudoku

00:01:40

1	5	9	4	8	3	7	2	6
4	8	3	7	2	6			9
7	2	6	1	5	9	4	8	3
5			8	3	7	2	6	1
8	3			6		5		
2			5	9	4		3	
9		8						5
3	7	2			5	9	4	
			9	4	8		7	

Novo

Verificar

Limpar

Sudoku

00:02:17

Solução errada



Novo

Verificar

Limpar

Sudoku

00:01:35

A Solução está correta



Novo

Verificar

Limpar

```
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
9       src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></s
10     cript>
11   </head>
12   <body onload="novoJogo()">
13     <div id="container">
14       <header>
15         <div id='header'><p>Sudoku</p></div>
16       </header>
17       <div align= 'center' id = 'counter'><h2>00:00:00</h2></div>
18       <hr>
19       <div id="teste">
20         <div id="area_sudoku">
21         </div>
22       </div>
23       <div id="op">
24         <button id='btn_start' type="button">Iniciar</button>
25         <button id='btn_novo' type="button">Novo</button>
26         <button id='btn_verificar' type="button">Verificar</button>
27         <button id='btn_limpar' type="button">Limpar</button>
28       </div>
29     </div>
30     <script>
31     <?php
32     include 'script.js';
33     ?>
34   </script>
35 </body>
36 </html>
```

```
1  /*
2  * função que limpa todos os campos editáveis
3  * preenchidos pelo jogador
4  */
5  function limpar() {
6      var x = document.getElementsByClassName('c1');
7      for (i = 0; i < x.length; i++) {
8          x[i].value = x[i].value.innerHTML = '';
9      }
10
11 }
12 /*
13 * função responsável por fazer a requisição
14 * para um novo jogo.
15 */
16 function novoJogo() {
17     var page = "controle.php";
18     $.ajax({
19         type: 'POST',
20         dataType: "html",
21         url: page,
22         beforeSend: function () {
23             $('#area_sudoku').html('');
24         },
25         data: {opt: 1},
26         cache: false,
27         success: function (msg) {
28             $('#area_sudoku').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 = [];
42     lin = 0;
43     col = 0;
44     for (i = 0; i <= a.length; i++) {
45         if (i % 9 === 0 & i !== 0) {
46             vet[lin] = v;
47             v = [];
48             col = 0;
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() {
66     var x = document.getElementsByClassName('campo');
67     var vetor = [];
68     for (i = 0; i < x.length; i++) {
69         vetor[i] = x[i].value;
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 () {
84             $('#area_sudoku').html('');
85         },
86         data: {vet: vet, tempo: tempo, opt: 2},
87         cache: false,
88         success: function (msg) {
89             $('#area_sudoku').html(msg);
90         }
91     });
92 }
93
94 /*
95  * função auxiliar responsável
96  * pela formatação do tempo
97  */
98 function formatatempo(hr, min, segs) {
99
100     if (hr < 10) {
101         hr = '0' + hr;
102     }
103     if (min < 10) {
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 // inicialização das variáveis de tempo
113 var segundos = 0;
114 var minutos = 0;
115 var horas = 0;
116
117 /*
118  * função responsável pelo incremento
119  * dos segundos, minutos e horas
120  */
121
122 function conta() {
123     segundos++;
124     if (segundos >= 60) {
125         segundos = 0;
126         minutos++;
127     }
128     if (minutos >= 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
137  * chamando a função conta() em intervalos de 1 segundo
138  */
139 function inicia() {
140     //se o contador já foi inicializado será resetado
141     if (typeof interval !== 'undefined') {
142         horas = 0;
143         minutos = 0;
```

```
144     segundos = 0;
145     clearInterval(interval);
146     interval = setInterval("conta();", 1000);
147 } else {
148     interval = setInterval("conta();", 1000);
149 }
150 }
151 /*
152  * Função responsável por garantir
153  * a validade da entrada passada pelo jogador
154  * admitindo somente números, e estes no intervalo
155  * fechado de 1 até 9
156  */
157 function validaEntrada() {
158     //função que garante a correção dos dados de entrada
159     var x = document.getElementsByClassName("c1");
160     var re = /^[1-9]$/;
161     for (i = 0; i < x.length; i++) {
162         if (!re.test(x[i].value)) {
163             x[i].value = x[i].value.innerHTML = '';
164         }
165     }
166 }
167
168 /*----- eventos dos botões -----*/
169
170
171 // botão criar novo jogo
172 $("#btn_novo").click(function () {
173     $('#btn_verificar').css('display', 'none');
174     if (typeof interval !== 'undefined') {
175         clearInterval(interval);
176     }
177     novoJogo();
178     $("#btn_start").show();
179
180 });
181
182 // botão inicia o jogo
183 $("#btn_start").click(function () {
184     $('.c1').css("background-color", '#DCDCDC');
185     $('#btn_verificar').css('display', 'block');
186     $('.c1').show();
187     $("#btn_start").hide();
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
6   * garantir uma solução única e exibir ao jogador um tabuleiro jogável,
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 possíveis
14
15     public function __construct() {
16         $this->tabuleiro = $this->gerarSudoku();
17     }
18
19     /*
20     * Este método varre a matriz e verifica se o tabuleiro
21     * está completo, ou seja, sem zeros nas células
22     */
23
24     function estaCheio($grid) {
25         for ($i = 0; $i < 9; $i++) {
26             for ($j = 0; $j < 9; $j++) {
27                 if ($grid[$i][$j] == 0) {
28                     return false;
29                 }
30             }
31         }
32         return true;
33     }
34
35     /* Verifica os candidatos à ocupar uma determinada
36     * 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 possíveis candidatos
43         $possibilidades = array(1, 2, 3, 4, 5, 6, 7, 8, 9);
44         // verificação horizontal
45         for ($i = 1; $i <= 9; $i++) {
46             for ($c = 0; $c < 9; $c++) {
47                 if ($grid[$lin][$c] == $i) {
48                     $possibilidades[$i - 1] = 0;
49                 }
50             }
51         }
52         // verificação vertical
53         for ($i = 1; $i <= 9; $i++) {
54             for ($l = 0; $l < 9; $l++) {
55                 if ($grid[$l][$col] == $i) {
56                     $possibilidades[$i - 1] = 0;
57                 }
58             }
59         }
60
61         $lir = (int) ($lin / 3);
62         $scr = (int) ($col / 3);
63
64         // verifica a região 3x3
65         for ($n = 1; $n <= 9; $n++) {
66             for ($i = ($lir * 3); $i < ($lir + 1) * 3; $i++) {
67                 for ($j = $scr * 3; $j < ($scr + 1) * 3; $j++) {
68                     if ($grid[$i][$j] == $n) {
69                         $possibilidades[$n - 1] = 0;
70                     }
71                 }
72             }
73         }
74     }
75 }
```

```

71         }
72     }
73     }return $possibilidades;
74 }
75
76 /*
77  * O método recebe um tabuleiro parcialmente preenchido
78  * e o preenche com todas as soluções possíveis e
79  * estas vão sendo contabilizadas para garantir
80  * que o tabuleiro à ser exibido ao jogador tenha apenas
81  * uma solução única */
82
83 function resolve($grid) {
84     // variaveis que guardam as coordenadas das células
85     // à serem preenchidas
86     $linha = 0;
87     $coluna = 0;
88
89     if ($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                     $coluna = $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
112         // responsável por retornar os candidatos para aquela posição.
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][$coluna] = 0;
124     }
125 }
126
127 /*
128  * Método responsável por exibir visualmente
129  * um grid ao jogador
130  */
131
132 public function criarGrid() {
133     for ($i = 1; $i <= 9; $i++) {
134         echo "<table border=3 class='tabela_fixa'>";
135         for ($k = 1; $k <= 3; $k++) {
136             echo "<tr class='linha'>";
137             for ($j = 1; $j <= 3; $j++) {
138                 echo "<td class='cel'></td>";
139             }
140             echo "</tr>";
141         }
142     }
143 }

```



```
141         echo"</table>";
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 /*
162  * Método responsável por verificar
163  * se o tabuleiro preenchido pelo jogador
164  * encontra correspondência no banco de dados
165  * e garantir que novos tabuleiros gerados não
166  * se repitam no BD
167  */
168
169 public function existeNoBD($t) {
170     $link = $this->getConexao();
171     $mat = json_encode($t);
172     $sql = "SELECT *FROM tabuleiros WHERE matriz = '$mat' ";
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() {
190     $n = 3;
191     $op = rand(1, 8); // opção entre um dos seis tabuleiros possíveis
192     $mat = array(); // início da construção da matriz
193     // a opção não pode ser um multiplo de 3
194     $c = $op % 3 != 0 ? $op : 1;
195     for ($i = 0; $i < 9; $i++) {
196         $new = array();
197         for ($j = 0; $j < 9; $j++) {
198             $v = ($i * $n + (int) ($i / $n) + $j) * $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 }
```

```

211     return $mat;
212 }
213
214 /*
215  * Método responsável por exibir ao jogador uma
216  * mescla de números e campos vazios editáveis
217  */
218
219 public function exibirSudoku() {
220     echo "<div id = 'tab'><table class='tabela' style='width: 600px;
221 height: 600px;'>";
222
223     // Enquanto a solução se mantém única
224     // uma célula é marcada aleatoriamente com zero.
225     while ($this->cont <= 1) {
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             $this->cont = 0;
235         }
236     }
237
238     // As células marcadas com zero serão os campos vazios editáveis
239     for ($i = 0; $i < 9; $i++) {
240         echo "<tr class='linha'>";
241         for ($j = 0; $j < 9; $j++) {
242             $p = $this->tabuleiro[$i][$j];
243             if ($p == 0) {
244                 echo '<td class="cel"><input type="text" class = "campo c1"
245                     onClick= "validaEntrada()"></td>';
246             } else {
247                 echo '<td class="cel"><input type="text" class="campo" value=' .
248                     $p . ' readonly></td>';
249             }
250         }
251         echo "</tr>";
252     }
253     echo "</table></div>";
254 }
255
256 /*
257  * Metodo responsável por fazer a conexão com
258  * o Banco de Dados.
259  */
260
261 private function getConexao() {
262     $servidor = 'localhost';
263     $usuario = 'root';
264     $senha = '';
265     $banco = 'db_sudoku';
266     $con = mysqli_connect($servidor, $usuario, $senha, $banco)
267         or die('Não foi possível conectar' . mysqli_error($con));
268     return $con;
269 }
270
271 /*
272  * Este método tem por finalidade fazer a conexao com o BD
273  * para avaliar se o tabuleiro
274  * preenchido pelo jogador existe no banco e,
275  * desta forma, verificar se sua resposta
276  * está correta ou não e se os seu tempo é o melhor para
277  * aquele tabuleiro em caso de acerto.
278  */
279
280 public function verificaTabJogador($t, $tempo) {

```

```
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);
284         if (!$res['tempo']) {
285             $recorde = date('H:i:s', $tempo);
286             mysqli_query($link, "UPDATE tabuleiros SET tempo = '$recorde' WHERE
                matriz = '$mat'");
287             echo "<h1><center style='margin:20px auto;color:red'>Melhor tempo:
                $recorde</center></h1>";
288         } else if ($tempo < strtotime($res['tempo'])) {
289             $recorde = date('H:i:s', $tempo);
290             mysqli_query($link, "UPDATE tabuleiros SET tempo = '$recorde' WHERE
                matriz = '$mat'");
291             echo "<h2><center style='margin:20px auto;'>Novo recorde :
                $recorde</center></h2>";
292         }
293         echo "<h1><center style='margin:30px auto;color:red'>A Solução está
            correta</center></h1>";
294         echo "<img style='margin-left:100px;' src='acertou.png' />";
295     } else {
296         echo "<h1><center style='margin:50px auto;color:red'>Solução
            errada</center></h1>";
297         echo "<img style='margin-left:100px;' src='errou.jpg' />";
298     }
299     mysqli_close($link);
300 }
301
302 }
303
304 //----- main() -----
305
306
307 $getparam = filter_input_array(INPUT_POST, FILTER_DEFAULT);
308
309 // O jogador 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         $t->exibirSudoku();
316         break;
317     case 2: // submeter sua resposta a correção
318         if (isset($getparam['vet'])) {
319             $tabela = $getparam['vet'];
320             $tempo = $getparam['tempo'];
321             $time = strtotime($tempo);
322
323             $t->verificaTabJogador($tabela, $time);
324         } else {
325             echo "<p><center style='margin:200px'>Inicie um novo jogo</center></p>";
326         }
327         break;
328 }
```