



MINISTÉRIO DA EDUCAÇÃO
UTFPR – UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
GRADUAÇÃO EM ENGENHARIA DE SOFTWARE



RELATÓRIO TÉCNICO
Projeto Final da Disciplina de Programação Web 1

BIANCA MARIA BORTOLOTO CANEZIN
EDUARDO DOS SANTOS SPAGNA

CORNÉLIO PROCÓPIO
DEZEMBRO – 2019

Sumário

1 OBJETIVO	2
2 TECNOLOGIAS USADAS	2
3 PARTES DO PROJETO	2
3.1 HTML - LISTAR ELEMENTOS	2
3.2 HTML - CADASTRAR ELEMENTOS	3
3.3 HTML - EDITAR ELEMENTOS	4
3.4 BOOTSTRAP - APLICAÇÃO	5
3.5 AJAX - SALVAR ELEMENTOS	6
3.6 AJAX - LISTAR ELEMENTOS	7
3.7 AJAX - EXCLUIR ELEMENTOS	8
3.8 AJAX - ATUALIZAR ELEMENTOS	9
3.9 JAVASCRIPT - VALIDAÇÃO	10
3.10 SWEETALERT - UTILIZAÇÃO	11
3.11 PAGINAÇÃO	12
3.12 BUSCA	13

1 OBJETIVO

O presente trabalho tem como objetivo explicar, item por item, como o projeto final da disciplina de programação web 1 foi desenvolvido.

2 TECNOLOGIAS USADAS

- HTML
- JavaScript
- Biblioteca AJAX
- API SweetAlert
- Banco de Dados "food.json"
- Biblioteca JSON Server
- BootstrapCDN

3 PARTES DO PROJETO

3.1 HTML - LISTAR ELEMENTOS

A **listagem** das comidas foi construída da seguinte maneira:

```
41 <!-- Table Foods -->
42 <table class="table">
43   <thead class="thead-light">
44     <tr>
45       <th scope="col">ID</th>
46       <th scope="col">Nome</th>
47       <th scope="col">Descrição</th>
48       <th scope="col">Preço</th>
49       <th scope="col">ID da Categoria</th>
50       <th scope="col">Editar</th>
51       <th scope="col">Excluir</th>
52     </tr>
53   </thead>
54
55   <tbody>
56     <tr></tr>
57   </tbody>
58 </table>
```

Arquivo: home.html

3.2 HTML - CADASTRAR ELEMENTOS

O cadastro de novas comidas foi construído da seguinte maneira:

```
30 <!-- Form -->
31 <form>
32   <div class="form-row">
33     <div class="col-md-4 mb-3">
34       <label for="foodName1">Nome*</label>
35       <input type="text" class="form-control" id="foodName" placeholder="Ex: pizza de mussarela" required />
36     </div>
37
38     <div class="col-md-3 mb-3">
39       <label for="foodPrice1">Preço R$*</label>
40       <input type="number" min="0.00" max="10000.00" step="0.01" class="form-control" id="foodPrice"
41         placeholder="Ex: R$ 29,00" required />
42     </div>
43
44     <div class="col-md-3 mb-3">
45       <label for="foodCategory1">ID da Categoria*</label>
46       <select class="custom-select mr-sm-2" id="foodCategory" required>
47         <option selected>Escolher...</option>
48         <option value="1">1 - Lanches</option>
49         <option value="2">2 - Bebidas</option>
50         <option value="3">3 - Sucos</option>
51         <option value="4">4 - Espetinhos</option>
52         <option value="5">5 - Pizzas</option>
53         <option value="6">6 - Sobremesas</option>
54       </select>
55     </div>
56   </div>
57
58   <div class="form-row">
59     <div class="col-md-10 mb-3">
60       <label for="foodDescription1">Descrição</label>
61       <textarea class="form-control" id="foodDescription" placeholder="Ex: molho, mussarela e catupiry"
62         rows="4"></textarea>
63     </div>
64   </div>
65
66   <button class="btn btn-secondary" onClick="validateFieldsToCreate()" type="button">
67     Salvar
68   </button>
69 </form>
70
71 <div style="color: red; visibility: hidden" id="requiredFields">* Campos Obrigatórios</div>
```

Arquivo: create-food.html

3.3 HTML - EDITAR ELEMENTOS

A edição das comidas foi construída da seguinte maneira:

```
88 <!-- Modal to Update Food -->
89 <div class="modal fade" id="modalUpdateFood" tabindex="-1" role="dialog" aria-labelledby="modalUpdateFoodLabel"
90   aria-hidden="true">
91   <div class="modal-dialog" role="document">
92     <div class="modal-content">
93       <div class="modal-header">
94         <h5 class="modal-title" id="modalUpdateFoodLabel">Editar comida</h5>
95         <button type="button" class="close" data-dismiss="modal" aria-label="Fechar">
96           <span aria-hidden="true">&times;</span>
97         </button>
98       </div>
99
100       <div class="modal-body">
101         <form>
102           <div class="form-group">
103             <label for="foodID1" class="col-form-label">ID:</label>
104             <input type="text" class="form-control" id="foodID" disabled />
105           </div>
106
107           <div class="form-group">
108             <label for="foodName1" class="col-form-label">Nome:</label>
109             <input type="text" class="form-control" id="foodName" placeholder="Ex: pizza de mussarela" required />
110           </div>
111
112           <div class="form-group">
113             <label for="foodPrice1" class="col-form-label">Preço R$:</label>
114             <input type="number" min="0.00" max="10000.00" step="0.01" class="form-control" id="foodPrice"
115               placeholder="Ex: R$ 29,00" required />
116           </div>
117
118           <div class="form-group">
119             <label for="foodCategory1" class="col-form-label">ID da Categoria:</label>
120             <select class="custom-select mr-sm-2" id="foodCategory" required>
121               <option selected>Escolher...</option>
122               <option value="1">1 - Lanches</option>
123               <option value="2">2 - Bebidas</option>
124               <option value="3">3 - Sucos</option>
125               <option value="4">4 - Espetinhos</option>
126               <option value="5">5 - Pizzas</option>
127               <option value="6">6 - Sobremesas</option>
128             </select>
129           </div>
130
131           <div class="form-group">
132             <label for="foodDescription1" class="col-form-label">Descrição:</label>
133             <textarea class="form-control" id="foodDescription" placeholder="Ex: molho, mussarela e catupiry" rows="4"
134               required></textarea>
135           </div>
136         </form>
137       </div>
138
139       <div class="modal-footer">
140         <button type="button" class="btn btn-secondary" data-dismiss="modal">Fechar</button>
141         <button type="button"
142           onClick="validateFieldsToUpdate({ foodID: $('#foodID').val(), foodName: $('#foodName').val(), foodPrice: $('#foodPrice').val(), foodCate
143             class="btn btn-primary">
144           Enviar
145         </button>
146         <div style="color: red; visibility: hidden" id="requiredFields">* Campos Obrigatórios</div>
147       </div>
148     </div>
149   </div>
150 </div>
151
```

Arquivo: home.html

3.4 BOOTSTRAP - APLICAÇÃO

A **aplicação** do Bootstrap foi realizada a partir das seguintes importações:

```
10
11     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
12         integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkF0JwJ8ERdknLPMO" crossorigin="anonymous" />
13 </head>

83     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
84         integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
85         crossorigin="anonymous"></script>
```

Arquivos: home.html e create-food.html

3.5 AJAX - SALVAR ELEMENTOS

A funcionalidade **salvar (cadastrar)** elementos, utilizando Ajax, foi implementada da seguinte maneira:

```
265  /**
266   * Create a food in the database
267   */
268  function createAFood({ foodName, foodPrice, foodCategory, foodDescription }) {
269    $.ajax({
270      method: "POST",
271      url: "http://localhost:3000/foods",
272      data: {
273        nome: foodName,
274        descricao: foodDescription,
275        preco: parseFloat(foodPrice),
276        categoria_id: parseInt(foodCategory)
277      },
278
279      success: function() {
280        Swal.fire({
281          title: "Sucesso",
282          icon: "success",
283          text: "Comida cadastrada com sucesso!",
284          showCancelButton: true,
285          cancelButtonColor: "#808080",
286          buttonsStyling: true,
287          cancelButtonText: "Voltar para página inicial",
288          confirmButtonText: "Cadastrar mais"
289        }).then(result => {
290          if (result.value) {
291            reloadPage();
292          } else {
293            window.location.href = "home.html";
294          }
295        });
296      },
297
298      error: function() {
299        Swal.fire("Erro", "Ocorreu um erro ao deletar a comida!", "error");
300      }
301    });
302  }
```

Arquivo: \assets\js\script.js

A função é chamada após a validação na função “validateFieldsToCreate()”.

3.6 AJAX - LISTAR ELEMENTOS

A funcionalidade **listar** elementos, utilizando Ajax, foi implementada da seguinte maneira:

```
34  /**
35   * Search all foods in the database and fill in the table with the data returned
36   */
37  function getAllFoodsAndFillTable() {
38    $.ajax({
39      url: "http://localhost:3000/foods",
40      type: "GET",
41
42      success: function(allFoods) {
43        fillTable({ data: allFoods });
44        pageButtons({ data: allFoods });
45      },
46
47      error: function(error) {
48        console.log(error);
49      }
50    });
51  }
```

```
7  function fillTable({ data }) {
8    clearTable();
9
10   for (
11     var index = page * pageSize;
12     index < data.length && index < (page + 1) * pageSize;
13     index++
14   ) {
15     newLine =
16       "<tr>" +
17       `<th scope="row">${data[index].id}</th>` +
18       `<td>${data[index].nome}</td>` +
19       `<td>${data[index].descricao}</td>` +
20       `<td>${data[index].preco}</td>` +
21       `<td>${data[index].categoria_id}</td>` +
22       `<td><button type="button" onclick="fillAndOpenModal({ foodID: ${data[index].id} })" class="btn btn-light"><
23       `<td><button type="button" onclick="deleteAFood({ foodID: ${data[index].id} })" class="btn btn-light"><img s
24       "</tr>";
25
26     $(".table > tbody > tr:last").after(newLine);
27   }
28
29   $("#numbering").text(
30     "Página " + (page + 1) + " de " + Math.ceil(data.length / pageSize)
31   );
32 }
```

Arquivo: \assets\js\script.js

3.7 AJAX - EXCLUIR ELEMENTOS

A funcionalidade **deletar** elementos, utilizando Ajax, foi implementada da seguinte maneira:

```
195  /**
196   * Delete a food in the database
197   */
198  function deleteAFood({ foodID }) {
199    $.ajax({
200      url: `http://localhost:3000/foods/${foodID}`,
201      type: "DELETE",
202
203      success: function() {
204        Swal.fire({
205          title: "Você tem certeza que deseja excluir essa comida?",
206          text: "Não tem como reverter essa ação!",
207          icon: "warning",
208          showCancelButton: true,
209          confirmButtonColor: "#99cc00",
210          cancelButtonColor: "#d33",
211          confirmButtonText: "Sim, deletar",
212          cancelButtonText: "Cancelar"
213        }).then(result => {
214          if (result.value) {
215            Swal.fire("Sucesso", "Comida deletada com sucesso!", "success").then(
216              () => {
217                reloadPage();
218              }
219            );
220          }
221        });
222      },
223
224      error: function() {
225        Swal.fire("Erro", "Ocorreu um erro ao deletar a comida!", "error");
226      }
227    });
228  }
```

Arquivo: \assets\js\script.js

3.8 AJAX - ATUALIZAR ELEMENTOS

A funcionalidade **atualizar** elementos, utilizando Ajax, foi implementada da seguinte maneira:

```
230  /**
231   * Update a food in the database
232   */
233  function updateAFood({
234    foodID,
235    foodName,
236    foodDescription,
237    foodPrice,
238    foodCategory
239  }) {
240    $.ajax({
241      url: `http://localhost:3000/foods/${foodID}`,
242      type: "PUT",
243      data: {
244        id: foodID,
245        nome: foodName,
246        descricao: foodDescription,
247        preco: parseFloat(foodPrice),
248        categoria_id: parseInt(foodCategory)
249      },
250
251      success: function() {
252        Swal.fire("Sucesso", "Comida atualizada com sucesso!", "success").then(
253          () => {
254            reloadPage();
255          }
256        );
257      },
258
259      error: function() {
260        Swal.fire("Erro", "Ocorreu um erro ao atualizar a comida!", "error");
261      }
262    });
263  }
```

Arquivo: \assets\js\script.js

A função é chamada após a validação na função “validateFieldsToUpdate()”.

3.9 JAVASCRIPT - VALIDAÇÃO

As **validações** de formulários, utilizando javascript, foram realizadas das seguintes maneiras:

```
120 function validateFieldsToUpdate({
121     foodID,
122     foodName,
123     foodDescription,
124     foodPrice,
125     foodCategory
126 }) {
127     if (
128         foodName !== "" &&
129         foodPrice !== "" &&
130         foodCategory !== "" &&
131         foodCategory !== "Escolher..."
132     ) {
133         updateAFood({ foodID, foodName, foodDescription, foodPrice, foodCategory });
134     }
135
136     if (foodName === "") {
137         markRequiredField({ elementID: "#foodName" });
138     }
139
140     if (foodPrice === "") {
141         markRequiredField({ elementID: "#foodPrice" });
142     }
143
144     if (foodCategory === "" || foodCategory === "Escolher...") {
145         markRequiredField({ elementID: "#foodCategory" });
146     }
147 }
```

```
149 function validateFieldsToCreate() {
150     const foodName = $("#foodName").val();
151     const foodDescription = $("#foodDescription").val();
152     const foodPrice = $("#foodPrice").val();
153     const foodCategory = $("#foodCategory").val();
154
155     if (
156         foodName !== "" &&
157         foodPrice !== "" &&
158         foodCategory !== "" &&
159         foodCategory !== "Escolher..."
160     ) {
161         createAFood({ foodName, foodDescription, foodPrice, foodCategory });
162     }
163
164     if (foodName === "") {
165         markRequiredField({ elementID: "#foodName" });
166     }
167
168     if (foodPrice === "") {
169         markRequiredField({ elementID: "#foodPrice" });
170     }
171
172     if (foodCategory === "" || foodCategory === "Escolher...") {
173         markRequiredField({ elementID: "#foodCategory" });
174     }
175 }
```

Arquivo: \assets\js\script.js

3.10 SWEETALERT - UTILIZAÇÃO

A **utilização** da API SweetAlert foi realizada conforme alguns exemplos:

```
251     success: function() {
252         Swal.fire("Sucesso", "Comida atualizada com sucesso!", "success").then(
253             () => {
254                 reloadPage();
255             }
256         );
257     },
258
259     error: function() {
260         Swal.fire("Erro", "Ocorreu um erro ao atualizar a comida!", "error");
261     }
```

```
203 ✓     success: function() {
204 ✓         Swal.fire({
205             title: "Você tem certeza que deseja excluir essa comida?",
206             text: "Não tem como reverter essa ação!",
207             icon: "warning",
208             showCancelButton: true,
209             confirmButtonColor: "#99cc00",
210             cancelButtonColor: "#d33",
211             confirmButtonText: "Sim, deletar",
212             cancelButtonText: "Cancelar"
213 ✓         }).then(result => {
214 ✓             if (result.value) {
215                 Swal.fire("Sucesso", "Comida deletada com sucesso!", "success").then(
216 ✓                     () => {
217                         reloadPage();
218                     }
219                 );
220             }
221 ✓         });
222     },
223
224 ✓     error: function() {
225         Swal.fire("Erro", "Ocorreu um erro ao deletar a comida!", "error");
226     }
```

Arquivo: \assets\js\script.js

3.11 PAGINAÇÃO

A **paginação** foi construída da seguinte maneira:

```
60 <!-- Pagination -->
61 <br />
62 <nav aria-label="...">
63   <ul class="pagination justify-content-end">
64     <li class="page-item">
65       <a class="page-link" id="anterior" href="#" tabindex="-1">Anterior</a>
66     </li>
67
68     <li class="page-item">
69       <span class="page-link" href="#" id="numbering"></span>
70     </li>
71
72     <li class="page-item">
73       <a class="page-link" id="proximo" href="#">Próximo</a>
74     </li>
75   </ul>
76 </nav>
```

Arquivo: home.html

A funcionalidade de **paginação** foi implementada da seguinte maneira:

```
323 function pageButtons({ data }) {
324   $("#proximo").prop(
325     data.length <= pageSize || page >= Math.ceil(data.length / pageSize) - 1
326   );
327
328   $("#anterior").prop(data.length <= pageSize || page == 0);
329
330   $(function() {
331     $("#proximo").click(function() {
332       if (page < data.length / pageSize - 1) {
333         page++;
334         getAllFoodsAndFillTable();
335         pageButtons();
336       }
337     });
338
339     $("#anterior").click(function() {
340       if (page > 0) {
341         page--;
342         getAllFoodsAndFillTable();
343         pageButtons();
344       }
345     });
346
347     pageButtons();
348   });
349 }
```

Arquivo: \assets\js\script.js

3.12 BUSCA

A **busca** foi construída da seguinte maneira:

```
31 <!-- Search -->
32 <form class="form-inline">
33   <input class="form-control mr-sm-2" type="search" id="searchFood" placeholder="Ex: hamburguer"
34     aria-label="Pesquisar" required />
35   <button class="btn btn-outline-success my-2 my-sm-0" type="button" onclick="validateFieldSearch()">
36     Pesquisar
37   </button>
38 </form>
39 </nav>
```

Arquivo: home.html

A funcionalidade **buscar** foi implementada da seguinte maneira:

```
304 /**
305  * Search by food name in database
306  */
307 function searchByFoodName({ foodName }) {
308   $.ajax({
309     url: `http://localhost:3000/foods?nome=${foodName}`,
310     type: "GET",
311
312     success: function(food) {
313       fillTable({ data: food });
314       Swal.fire("Sucesso", "Pesquisa realizada com sucesso", "success");
315     },
316
317     error: function() {
318       Swal.fire("Erro", "Ocorreu um erro ao realizar a pesquisa", "error");
319     }
320   });
321 }
```

Arquivo: \assets\js\script.js