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

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:

```
<!-- Form -->
<form>
  <div class="form-row">
    <div class="col-md-4 mb-3">
      <label for="foodName1">Nome*</label>
      <input type="text" class="form-control" id="foodName" placeholder="Ex: pizza de mussarela" required />
    </div>

    <div class="col-md-3 mb-3">
      <label for="foodPrice1">Preço R$*</label>
      <input type="number" min="0.00" max="10000.00" step="0.01" class="form-control" id="foodPrice"
        placeholder="Ex: R$ 29,00" required />
    </div>

    <div class="col-md-3 mb-3">
      <label for="foodCategory1">ID da Categoria*</label>
      <select class="custom-select mr-sm-2" id="foodCategory" required>
        <option selected>Escolher...</option>
        <option value="1">1 - Lanches</option>
        <option value="2">2 - Bebidas</option>
        <option value="3">3 - Sucos</option>
        <option value="4">4 - Espetinhos</option>
        <option value="5">5 - Pizzas</option>
        <option value="6">6 - Sobremesas</option>
      </select>
    </div>
  </div>

  <div class="form-row">
    <div class="col-md-10 mb-3">
      <label for="foodDescription1">Descrição</label>
      <textarea class="form-control" id="foodDescription" placeholder="Ex: molho, mussarela e catupiry"
        rows="4"></textarea>
    </div>
  </div>

  <button class="btn btn-secondary" onclick="validateFieldsToCreate()" type="button">
    Salvar
  </button>
</form>

<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:

```
<!-- Modal to Update Food -->
<div class="modal fade" id="modalUpdateFood" tabindex="-1" role="dialog" aria-labelledby="modalUpdateFoodLabel"
  aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="modalUpdateFoodLabel">Editar comida</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Fechar">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>

      <div class="modal-body">
        <form>
          <div class="form-group">
            <label for="foodID1" class="col-form-label">ID:</label>
            <input type="text" class="form-control" id="foodID" disabled />
          </div>

          <div class="form-group">
            <label for="foodName1" class="col-form-label">Nome:</label>
            <input type="text" class="form-control" id="foodName" placeholder="Ex: pizza de mussarela" required />
          </div>

          <div class="form-group">
            <label for="foodPrice1" class="col-form-label">Preço R$:</label>
            <input type="number" min="0.00" max="10000.00" step="0.01" class="form-control" id="foodPrice"
              placeholder="Ex: R$ 29,00" required />
          </div>

          <div class="form-group">
            <label for="foodCategory1" class="col-form-label">ID da Categoria:</label>
            <select class="custom-select mr-sm-2" id="foodCategory" required>
              <option selected>Escolher...</option>
              <option value="1">1 - Lanches</option>
              <option value="2">2 - Bebidas</option>
              <option value="3">3 - Sucos</option>
              <option value="4">4 - Espetinhos</option>
              <option value="5">5 - Pizzas</option>
              <option value="6">6 - Sobremesas</option>
            </select>
          </div>
```

```
          </div>

          <div class="form-group">
            <label for="foodDescription1" class="col-form-label">Descrição:</label>
            <textarea class="form-control" id="foodDescription" placeholder="Ex: molho, mussarela e catupiry" rows="4"
              required></textarea>
          </div>
        </form>
      </div>

      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Fechar</button>
        <button type="button"
          onclick="validateFieldsToUpdate({ foodID: $('#foodID').val(), foodName: $('#foodName').val(), foodPrice: $('#foodPrice').val(), foodCate
          class="btn btn-primary">
          Enviar
        </button>
        <div style="color: red; visibility: hidden" id="requiredFields">* Campos Obrigatórios</div>
      </div>
    </div>
  </div>
</div>
```

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/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" 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:

```
/**
 * Create a food in the database
 */
function createAFood({ foodName, foodPrice, foodCategory, foodDescription }) {
  $.ajax({
    method: "POST",
    url: "http://localhost:3000/foods",
    data: {
      nome: foodName,
      descricao: foodDescription,
      preco: parseFloat(foodPrice),
      categoria_id: parseInt(foodCategory)
    },

    success: function() {
      Swal.fire({
        title: "Sucesso",
        icon: "success",
        text: "Comida cadastrada com sucesso!",
        showCancelButton: true,
        cancelButtonColor: "#808080",
        buttonsStyling: true,
        cancelButtonText: "Voltar para página inicial",
        confirmButtonText: "Cadastrar mais"
      }).then(result => {
        if (result.value) {
          realoadPage();
        } else {
          window.location.href = "home.html";
        }
      });
    },

    error: function() {
      Swal.fire("Erro", "Ocorreu um erro ao deletar a comida!", "error");
    }
  });
}
```

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:

```
/**
 * Search all foods in the database and fill in the table with the data returned
 */
function getAllFoodsAndFillTable() {
  $.ajax({
    url: "http://localhost:3000/foods",
    type: "GET",

    success: function(allFoods) {
      fillTable({ data: allFoods });
      pageButtons({ data: allFoods });
    },

    error: function(error) {
      console.log(error);
    }
  });
}
```

```
function fillTable({ data }) {
  clearTable();

  for (
    var index = page * pageSize;
    index < data.length && index < (page + 1) * pageSize;
    index++
  ) {
    newLine =
      "<tr>" +
      `<th scope="row">${data[index].id}</th>` +
      `<td>${data[index].nome}</td>` +
      `<td>${data[index].descricao}</td>` +
      `<td>${data[index].preco}</td>` +
      `<td>${data[index].categoria_id}</td>` +
      `<td><button type="button" onclick="fillAndOpenModal({ foodID: ${data[index].id} })" class="btn btn-light"><
      `<td><button type="button" onclick="deleteAFood({ foodID: ${data[index].id} })" class="btn btn-light"><img s
      "</tr>";

    $(".table > tbody > tr:last").after(newLine);
  }

  $("#numbering").text(
    "Página " + (page + 1) + " de " + Math.ceil(data.length / pageSize)
  );
}
```

Arquivo: \assets\js\script.js

3.7 AJAX - EXCLUIR ELEMENTOS

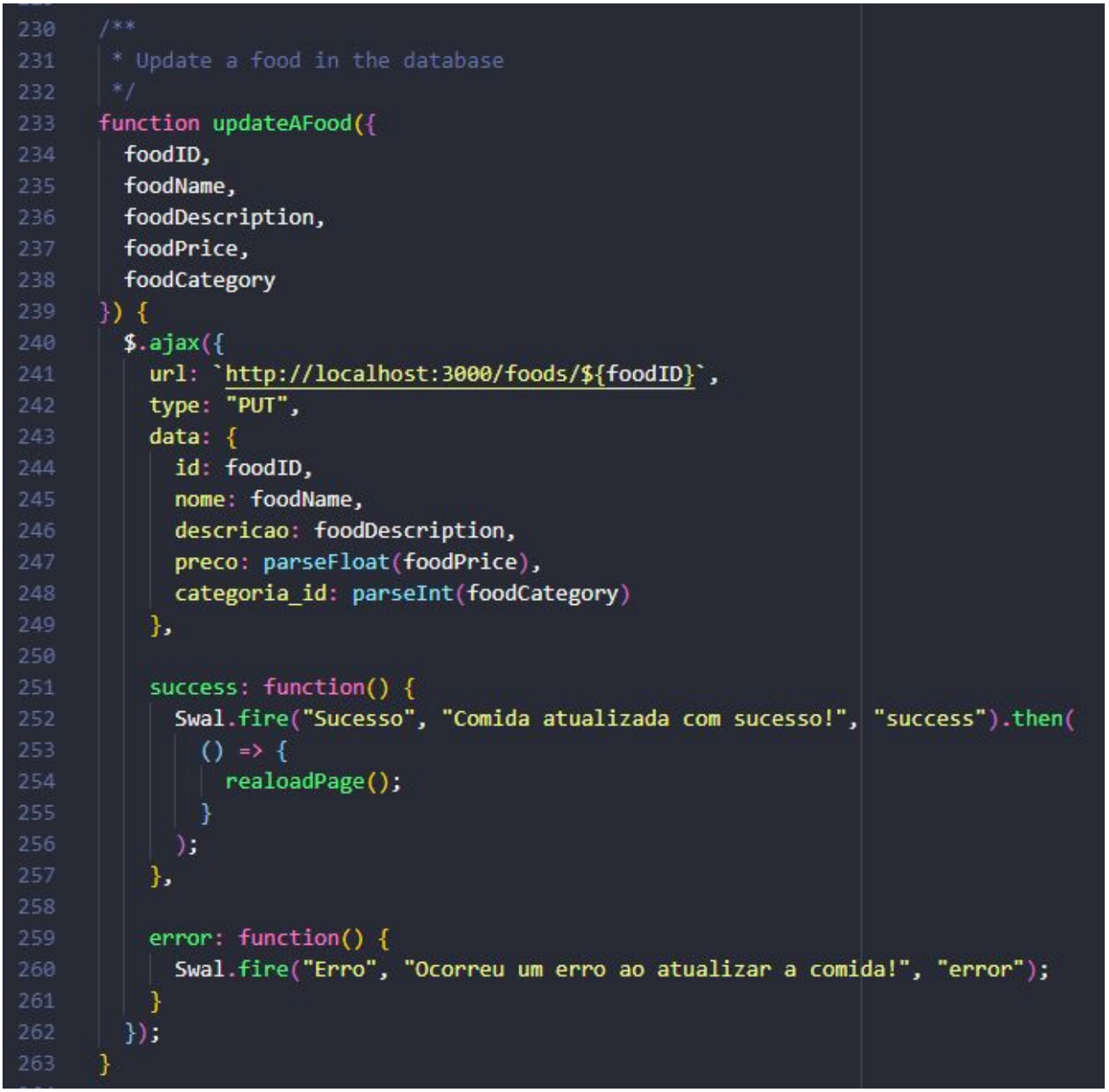
A funcionalidade **deletar** elementos, utilizando Ajax, foi implementada da seguinte maneira:

```
/**
 * Delete a food in the database
 */
function deleteAFood({ foodID }) {
  $.ajax({
    url: `http://localhost:3000/foods/${foodID}`,
    type: "DELETE",

    success: function() {
      Swal.fire({
        title: "Você tem certeza que deseja excluir essa comida?",
        text: "Não tem como reverter essa ação!",
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#99cc00",
        cancelButtonColor: "#d33",
        confirmButtonText: "Sim, deletar",
        cancelButtonText: "Cancelar"
      }).then(result => {
        if (result.value) {
          Swal.fire("Sucesso", "Comida deletada com sucesso!", "success").then(
            () => {
              realoadPage();
            }
          );
        }
      });
    },

    error: function() {
      Swal.fire("Erro", "Ocorreu um erro ao deletar a comida!", "error");
    }
  });
}
```

Arquivo: \assets\js\script.js

3.8 AJAX - ATUALIZAR ELEMENTOS

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



```
/**
 * Update a food in the database
 */
function updateAFood({
  foodID,
  foodName,
  foodDescription,
  foodPrice,
  foodCategory
}) {
  $.ajax({
    url: `http://localhost:3000/foods/${foodID}`,
    type: "PUT",
    data: {
      id: foodID,
      nome: foodName,
      descricao: foodDescription,
      preco: parseFloat(foodPrice),
      categoria_id: parseInt(foodCategory)
    },

    success: function() {
      Swal.fire("Sucesso", "Comida atualizada com sucesso!", "success").then(
        () => {
          realoadPage();
        }
      );
    },

    error: function() {
      Swal.fire("Erro", "Ocorreu um erro ao atualizar a comida!", "error");
    }
  });
}
```

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:

```
function validateFieldsToUpdate({
  foodID,
  foodName,
  foodDescription,
  foodPrice,
  foodCategory
}) {
  if (
    foodName !== "" &&
    foodPrice !== "" &&
    foodCategory !== "" &&
    foodCategory !== "Escolher..."
  ) {
    updateAFood({ foodID, foodName, foodDescription, foodPrice, foodCategory });
  }

  if (foodName === "") {
    markRequiredField({ elementID: "#foodName" });
  }

  if (foodPrice === "") {
    markRequiredField({ elementID: "#foodPrice" });
  }

  if (foodCategory === "" || foodCategory === "Escolher...") {
    markRequiredField({ elementID: "#foodCategory" });
  }
}
```

```
function validateFieldsToCreate() {
  const foodName = $("#foodName").val();
  const foodDescription = $("#foodDescription").val();
  const foodPrice = $("#foodPrice").val();
  const foodCategory = $("#foodCategory").val();

  if (
    foodName !== "" &&
    foodPrice !== "" &&
    foodCategory !== "" &&
    foodCategory !== "Escolher..."
  ) {
    createAFood({ foodName, foodDescription, foodPrice, foodCategory });
  }

  if (foodName === "") {
    markRequiredField({ elementID: "#foodName" });
  }

  if (foodPrice === "") {
    markRequiredField({ elementID: "#foodPrice" });
  }

  if (foodCategory === "" || foodCategory === "Escolher...") {
    markRequiredField({ elementID: "#foodCategory" });
  }
}
```

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           realoadPage();
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               realoadPage();
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:

```
<!-- Pagination -->
<br />
<nav aria-label="...">
  <ul class="pagination justify-content-end">
    <li class="page-item">
      <a class="page-link" id="anterior" href="#" tabindex="-1">Anterior</a>
    </li>

    <li class="page-item">
      <span class="page-link" href="#" id="numbering"></span>
    </li>

    <li class="page-item">
      <a class="page-link" id="proximo" href="#">Próximo</a>
    </li>
  </ul>
</nav>
```

Arquivo: home.html

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

```
function pageButtons({ data }) {
  $("#proximo").prop(
    data.length <= pageSize || page >= Math.ceil(data.length / pageSize) - 1
  );

  $("#anterior").prop(data.length <= pageSize || page == 0);

  $(function() {
    $("#proximo").click(function() {
      if (page < data.length / pageSize - 1) {
        page++;
        getAllFoodsAndFillTable();
        pageButtons();
      }
    });

    $("#anterior").click(function() {
      if (page > 0) {
        page--;
        getAllFoodsAndFillTable();
        pageButtons();
      }
    });

    pageButtons();
  });
}
```

Arquivo: \assets\js\script.js

3.12 BUSCA

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

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

Arquivo: home.html

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

```
/**
 * Search by food name in database
 */
function searchByFoodName({ foodName }) {
  $.ajax({
    url: `http://localhost:3000/foods?nome=${foodName}`,
    type: "GET",

    success: function(food) {
      fillTable({ data: food });
      Swal.fire("Sucesso", "Pesquisa realizada com sucesso", "success");
    },

    error: function() {
      Swal.fire("Erro", "Ocorreu um erro ao realizar a pesquisa", "error");
    }
  });
}
```

Arquivo: \assets\js\script.js