UNIVERSIDADE FEDERAL DE SÃO CARLOS
CAMPUS SOROCABA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO WEB

PROF ª DRA. LUCIANA APARECIDA MARTINEZ ZAINA

MINI PROJETO 1

Desenvolvimento da versão inicial, com o front-end e comunicação AJAX sem uso de banco de dados, de uma aplicação web para verificar rotas de ônibus

Fundamentos envolvidos: HTML, CSS, JS, Ajax e Servlet

INTEGRANTES DO GRUPO

Bruna Scarpelli - 800435

Felipe Ottoni Pereira - 804317

Gustavo Sanches Martins Kis - 801319

Letícia Almeida Paulino de Alencar Ferreira - 800480

Ricardo Yugo Suzuki - 802003

Índice

Descrição da Aplicação web

O mini projeto 1 consiste no desenvolvimento da versão inicial de uma aplicação web para verificar rotas de ônibus. Desse modo ele é composto pelo front-end e com comunicação AJAX, sem uso de banco de dados.

A princípio, criamos os arquivos ***html*** para construção das páginas **"**`index. html`**"** e **"**`cadastro.html`**"** e estruturamos cada uma, os ícones utilizados estão na pasta **"**`Assets`**"**. Com isso, desenvolvemos os arquivos ***css*** na pasta **"**`styles`**"** para a estilização das páginas. Nesse sentido, em **"**`cadastro.html`**"** criamos os arquivos ***javascript*** "`camposForm.js`" e `inputValidacao.js`" presentes na pasta **"**`script`**"** para deixar os campos "Pontos de parada" e "Horários de saída" dinâmicos, uma vez que eles podem receber uma sequência de entradas, e para validar as entradas dos campos do formulário, respectivamente.

Além disso, o arquivo javascript "`itinerario.js`**",** também na pasta **"**`script`**",** foi utilizado para realizar uma requisição assíncrona usando ***ajax***. Assim, criamos também os arquivos "`itinerarios.jsp`" para criar uma página da web dinâmica para itinerário, onde há a busca de itinerários, com código java embutido que é executado no servidor antes de ser enviado para o navegador do usuário e **"**`buscaRegiaoServlet.java`", que é um ***servlet*** para receber a solicitação feita pelo ajax, realizar a pesquisa e gerar uma resposta a ser enviada ao cliente.

Portanto, para melhor explicação das etapas de desenvolvimento, cada subseção abaixo descreve o que foi realizado nas partes que compõem a aplicação apontando a função de cada arquivo da aplicação.

HTML

Temos duas páginas em html nesta aplicação, a "index.html" e "cadastro.html". Além disso, é importante destacar que todas possuem os elementos `<header>`, com o menu de navegação com os elementos "home" e "rotas" obrigatórios mais "história", "cartão" e "transparência" que foram determinados pelo grupo, e o `<footer>` com uma seção de links rápidos e informações de identificação sobre trabalho e grupo.

Desse modo, a "index. html", responsável pelo conteúdo da Home da aplicação onde estão os cards para as linhas de ônibus mais usadas, compostos pelos campos de nome e número da linha de ônibus, a região da cidade pela qual ela atravessa, os pontos de início e

final da rota que ele faz e todos os horários que o ônibus sai de seu ponto de início, começando assim mais uma rota. Tendo em vista que os horários de saída são múltiplos, foi necessário o uso de `<table>` para uma melhor estética da visualização das horas de partida.

Já "cadastro.html" é a página em que há a implementação da tela de “cadastro de itens” a qual é carregada ao clicar no submenu de mesmo nome, assim, ela permite cadastrar um itinerário por meio de um formulário composto pelos campos definidos nos cards presentes na Home da aplicação. Portanto, as entradas a serem recebidas são os dados: "Nome da linha", "Região da cidade", "Pontos de parada" e "Horário de Saída", sobre um determinado itinerário. Além disso, observa-se que como a página possui o menu de navegação é possível sair da tela de cadastro e voltar a home da aplicação.

CSS

Existem cinco arquivos *css* dentro da pasta “`styles`”, sendo os “`home.css`”, “`cadastro.css`” e “`itinerário.css`” para a estilização das páginas “index. html”, “cadastro.html” e “`itinerario.jsp`”, respectivamente. E “`header.css`” e “`footer.css`” para a estilização dos elementos de mesmo nome presentes em todas as páginas.

JavaScript

Os arquivos javascript estão localizados na pasta “script”. Então, como eles são externos, adicionamos eles no arquivo “`cadastro.html`” e “`itinerario.jsp`”, pois é onde eles são necessários, com a tag: “`<script src=""></script>`” ao final do arquivo html, antes de: `</body></html>`.

O arquivo “`camposForm.js`” possui as funções que permite os campos “Pontos de parada” e “Horários de saída” serem dinâmicos, logo, o usuário pode adicionar ou remover campos. Isso porque na descrição do trabalho, ao cadastrar os pontos de parada foi definido que o cadastro deve ter uma lista com os pontos de parada do ônibus, assim, cada itinerário tem diversos pontos e cada um com uma quantidade diferente. E com a definição feita pelo grupo sobre o campo “horários de saída”, também achamos necessário que este campo receba uma lista com os horários.

No arquivo “inputValidacao.js” estão as funções responsáveis por verificar o formato de cada entrada do formulário de cadastro de itinerários, os *Event Listeners* e as mensagens de erro para o evento de *input* no formato incorreto. Desse modo, utilizamos eventos para acionar a validação conforme o usuário interagir com os campos de entrada, assim, usamos o evento *input*, que é acionado sempre que o conteúdo de um campo de entrada é alterado.

Já o arquivo “itinerarios.js” relaciona a operação de busca de itinerários à nossa aplicação. Então, ele possui uma função que realiza uma requisição assíncrona, usando *AJAX,* ao arquivo que contém o código java (servlet) para buscar os itinerários com base em uma região fornecida como entrada pelo usuário, e atualiza dinamicamente a página com os resultados da busca, sem recarregar a página inteira.

JSP

O arquivo “itinerarios.jsp” segue a abordagem clássica de dividir a lógica do lado do servidor (Java) e a apresentação do lado do cliente (HTML, CSS, JavaScript), gerando uma página da web dinâmica para itinerário. Isso permite ao usuário inserir uma região específica para que ocorra uma busca de itinerários desta região e receber a resposta. Então, na busca é especificado que os dados serão enviados ao *servlet*, em “action="buscaRegiaoServlet"”, que retorna os itinerários de acordo com a região colocada, e é definido o evento a ser acionado quando o formulário for enviado, pois em “onsubmit="return buscarItinerarios();"” a função buscarItinerarios(), que está no arquivo *javascript* “itinerarios.js”, é chamada.

JAVA (Servlet)

O arquivo “buscaRegiaoServlet.java”, localizado na pasta “java”, é responsável por receber a solicitação “POST” do AJAX, vinda do arquivo “itinerarios.js”, com a região inserida pelo usuário para saber os itinerários dela. Desse modo, ele realiza uma busca simulada de itinerários, que criamos para cada região, e então, com base na região passada pelo usuário ele gera uma resposta HTML que é enviada de volta ao cliente.

Fontes e inspirações

Essa seção reúne os links das fontes externas e de outros recursos utilizados para construir a aplicação desenvolvida e onde eles foram usados.

Ícones

Os ícones utilizados no *html* para a logo da aplicação e imagem de um ônibus nos *cards* das linhas de ônibus, guardados na pasta “Assets” do projeto, são derivados do SVG Repo - Free SVG Vectors and Icons. Disponível em: <https://www.svgrepo.com/>.

Sites de linhas de ônibus

Para nos inspirarmos, saber que outros elementos do menu poderíamos ter e, assim, termos uma base de exemplo, consultamos os sites de linhas de ônibus listados abaixo:

- URBES - TRÂNSITO E TRANSPORTE. Disponível em: <https://www.urbes.com.br/>. Acesso em: 10 dez. 2023.
- ADMINISTRAÇÃO), D. DE I. (SECRETARIA M. DE. Horários de Ônibus. Disponível em: <https://www.indaiatuba.sp.gov.br/horarios-de-onibus/>. Acesso em: 10 dez. 2023.
- IPLANRIO, P. -. Prefeitura do Rio. Disponível em: <https://onibus.rio/>. Acesso em: 10 dez. 2023.

Códigos disponíveis na web que consultamos

Para conseguirmos fazer o **submenu** dentro do elemento “`rotas`” do menu de navegação presente em:

```
<header>
        <div id="logo">
            <div>
             <img src="./Assets/bee-origami-paper-svgrepo-com.svg" alt="abelha
de origami" height="42">
            </div>

            <div>
                <h1>Busbus</h1>
```

```
            </div>
        </div>

        <nav class="menu">
            <ul> <!-- Menu de navegação -->
                <li><a href="./index.html">Home</a></li>
                <div class="dropdown">
                   <button class="dropbtn">Rotas</button>
                   <div class="dropdown-content">
                     <a href="./itinerarios.html">Itinerários</a>
                     <a href="#">Cadastro de itinerários</a>
                   </div>
                 </div>
                <li><a href="#">História</a></li>
                <li><a href="#">Cartão</a></li>
                <li><a href="#">Transparência</a></li>
            </ul>
        </nav>
    </header>
```

de todos arquivos *html* e a estilização em "`header.css`", seguimos o exemplo disponibilizado em: How To Create a Hoverable Dropdown Menu. Disponível em: <https://www.w3schools.com/howto/howto_css_dropdown.asp>.

Na etapa de **construção dos *cards*** que aparecem na tela da Home com as linhas de ônibus mais utilizadas, primeiro os criamos apenas por meio dos recursos que conhecíamos, porém, eles estavam muito simples e queríamos algo mais visualmente agradável e organizado. Por isso, procuramos adicionar mais elementos e melhorar a estilização deles, assim, consultamos o código encontrado em: Card hover effect experiments (codepen.io). Essa parte está no conteúdo da home em "`index.html`":

```
<div class="container">
            <div class="card">
                  <img class="align-center" src="./Assets/bus-svgrepo-com.svg"
alt="Ícone de Linha" height="32">
                <h3>06 - Nova Miami</h3>
                <p><b>Região da Cidade:</b> Sul</p>
                <p><b>Ponto inicial:</b> Terminal Grená</p>
                <p><b>Ponto final:</b> Ocean Drive</p>
                <h4><b>Horários de Saída:</b></h4>
                <table class="tabela">
                    <thead>
                      <tr>
                        <th>Manhã</th>
                        <th>Tarde</th>
                        <th>Noite</th>
                      </tr>
                    </thead>
```

```
                    <tbody>
                      <tr>
                        <td>07:15</td>
                        <td>12:00</td>
                        <td>18:00</td>
                      </tr>
                      <tr>
                        <td>09:00</td>
                        <td>14:30</td>
                        <td>20:30</td>
                      </tr>
                      <tr>
                        <td>10:00</td>
                        <td>16:00</td>
                        <td>22:00</td>
                      </tr>
                    </tbody>
                  </table>
                  <div class="animacao" href="#">
                    <div class="seta">
                      →
                    </div>
                  </div>
              </div>
...
</div>
```

e a estilização em “`home.css`” dentro da pasta “`styles`”.

Na tela de cadastro no arquivo “`cadastro.html`” (linha 90: `<button id="enviar" type="submit">Enviar</button>`) o **botão de enviar**, estilizado em “`cadastro.css`”, possui uma animação, que foi feita a partir da referência disponível em: [Showing icon on hover in button (codepen.io)].

```
#enviar{
    max-width: 150px;
    background: #FFD30F;
    color: #ffffff;
    cursor: pointer;
    font-size: 1.275rem;
    padding: 1rem;
    border: 0;
    transition: all 0.5s;
    position: relative;
    margin: 0 auto;
    text-align: center;
    display: block;
    font-weight: bolder;

    &::after {
        content: ">";
        position: absolute;
```

```
        left: 75%;
        top: 30%;
        right: 20%;
        bottom: 0;
        opacity: 0;

    }

    &:hover {
        background: #fad842;
        transition: all 0.5s;
        padding: 1rem 3rem 1rem 1rem;

        &::after {
            opacity: 1;
            transition: all 0.5s;

        }
    }
}
```

Para a parte de **validação dos dados de entrada** no formulário em "`cadastro.html`" construímos o arquivo javascript "`inputValidacao.js`" presente na pasta "`script`". Então, para saber como determinar os formatos de cada entrada, escrever as funções de verificação e determinar as mensagens de erros consultamos as fontes abaixo:

- Exemplo desenvolvido em sala de aula em: 14/11:: Front-End - Javascript - Parte 2 > [Exemplo 3];
- Form data validation - Aprendendo desenvolvimento web | MDN. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Forms/Form_validation#validando_formul%C3%A1rios_usando_javascript>. Acesso em: 07 dez. 2023.
- Validação de formulários HTML com JavaScript. Disponível em: <https://www.mhavila.com.br/topicos/web/valform.html>. Acesso em: 07 dez. 2023.

Foi descrito que "o cadastro deve ter uma lista com os pontos de parada do ônibus;" por isso decidimos que seria interessante ter um **campo dinâmico** em nosso formulário, em "`cadastro.html`" e "`cadastro.css`", para as paradas de ônibus e também para os horários de saída. Nesse sentido, para isso nos inspiramos nos códigos encontrados nas referências abaixo:

- Dynamically Add/Remove Table Rows Using jQuery - rowfy | Free jQuery Plugins. Disponível em:

<https://www.jqueryscript.net/table/add-remove-table-rows-rowfy.html>. Acesso em: 07 dez. 2023.

- Dynamically Create HTML Form Fields - jQuery Dynamic Forms | Free jQuery Plugins. Disponível em: <https://www.jqueryscript.net/form/dynamic-forms-fields.html>. Acesso em: 06 dez. 2023.

Bibliotecas

- Fonte de texto: GOOGLE. Google Fonts. Disponível em: <https://fonts.google.com/>.

Na `head` de todos os *html*s:

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=DM+Sans:opsz,wght@9..40,200;9..
40,300;9..40,500&display=swap" rel="stylesheet">
```

Em "`header.css`":

```
* {
    margin: 0;
    padding: 0;
    border: 0;
    box-sizing: border-box;
    list-style: none;
    font-family: 'DM Sans', sans-serif;
}
```

As bibliotecas importadas em "`buscaRegiaoServlet.java`" foram copiadas do exemplo disponibilizado para a prática da aula sobre Servlet. Portanto, para consultar sobre os recursos delas usamos as seguintes fontes:

- Java.io.IOException: IOException (Java Platform SE 8) (oracle.com)
- java.io.PrintWriter: PrintWriter (Java Platform SE 8) (oracle.com)
- javax.servlet.ServletException: ServletException (Java(TM) EE 7 Specification APIs) (oracle.com)

- javax.servlet.annotation.WebServlet: WebServlet (Java(TM) EE 7 Specification APIs) (oracle.com)
- javax.servlet.http.HttpServlet: HttpServlet (Java(TM) EE 7 Specification APIs) (oracle.com)
- javax.servlet.http.HttpServletRequest: HttpServletRequest (Java(TM) EE 7 Specification APIs) (oracle.com)
- javax.servlet.http.HttpServletResponse: HttpServletResponse (Java(TM) EE 7 Specification APIs) (oracle.com)