UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
TECNOLOGIA EM SISTEMAS PARA INTERNET

CAMPOS, ESTER DE ARRUDA
CAVALCANTI, MARCUS VINICIUS

TEMA: FOTOS

PROJETO DESENVOLVIMENTO PARA WEB II

GUARAPUAVA

2016

CAMPOS, ESTER DE ARRUDA
CAVALCANTI, MARCUS VINICIUS

TEMA: FOTOS

Projeto apresentado ao professor Dr. Roni Fabio Banaszewski como requisito parcial para composição de nota semestral de sua respectivas disciplina – Desenvolvimento para Web II.

GUARAPUAVA

2016

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

1.2 OBJETIVO

O presente projeto tem por objetivo apresentar o site que foi desenvolvido abrangendo os conteúdos da disciplina de Desenvolvimento para Web II. Parametrizamos a elaboração deste site nos pre-requisitos listados pelo docente da disciplina, conforme está definida na está definido no sumário deste.

1.3 ESTRUTURA DO PROJETO

O Capítulo 1 – Introdução, irá definir o que vem a ser o projeto e quais os objetivos a serem analisados. No Capítulo 2, estaremos apresentando as exigências Gerais definidas como requisito para elaboração do projeto. O Capítulo 3, apresentará respostas para as exigências específicas e seus devidos tópicos. Por fim, no Capítulo 4 apresentaremos alguns trechos de código.

Toda formatação para este texto incluindo a formatação para apresentação dos códigos elaborados foram feitos utilizando a linguagem *LaTeX*.

1.4 METODOLOGIA UTILIZADA

Utilizaremos o método de pesquisa aplicada cuja meta é contribuir para fins práticos buscando solução para problema concreto.

2 EXIGÊNCIAS GERAIS

2.1 CONTEÚDO

O conteúdo do site é de autoria pessoal. Todas as fotos são do portfólio da fotógrafa a mesma disponibiliza e cede os direitos de utilização de imagens para este projeto, com citação a devida citação. fotografias: Ester Campos.

2.2 HTML5 E CSS

Neste projeto foram utilizados *HTML5* e *CSS* um dos exemplos para utilização da *CSS* podem ser conferidos no item *Layout*.

2.3 FRAMEWORK

Foi utilizado o framework Bootstrap seguindo uma das recomendações.

2.4 LAYOUT

Conforme indicado pelo docente uma das páginas do projeto precisa ser totalmente responsiva. Desta forma apresentamos a página *index.html*.

Não utilizamos *templates* prontos. Nesta página fazemos utilização de *CSS*.

```
1 <link href="assets/libs/bootstrap/css/bootstrap.css" rel="stylesheet">
2 <link href="assets/resources/css/principal.css" rel="stylesheet">
3 <link href="assets/resources/css/home.css" rel="stylesheet">
```

Esta tela é o primeiro contato que um cliente que esteja visitando a página terá. Por este motivo pensamos em mostrar algumas imagens para que o visitante tenha a sensação de que está no lugar certo.

Desta forma deixamos na página inicial apenas um *slider* onde são apresentadas cinco (5) fotos. Este *slider* foi montado para apresentar uma transição de fotos de maneira simples. O intervalo entre uma foto e outra é feito de modo que apareça um efeito de opacidade.

O efeito de opacidade e transição foi conseguido utilizando *CSS transitions* seguindo algumas ideias de Rich Bradshaw em seu site (**??**).

2.4.0.1 HTML E CSS

Criamos um arquivo chamado *home.css* onde estão as configurações *crossfade* do *css*. São estas configurações as responsáveis pelos efeitos de transição/animação das 5 imagens. Este efeito se dá por 30*s*.

Também foi criado um arquivo chamado *principal.css* neste, estão as configurações para o menu *menuToggle* que utilizamos nesta página e que também utilizaremos nas demais páginas deste projeto.

Abaixo um trecho do arquivo *index.html* onde na *tag* $<$ *nav* $>$ adicionamos a classe *menu* e suas configurações são definidas no arquivo *principal.css*.

```
<nav class="menu" id="theMenu">
  <div class="menu-wrap">
    <h1 class="logo"><a href="index.html#home">StudioE</a></h1>
    <i class="fa fa-arrow-right menu-close"></i>
    <a href="index.html"><span>HOME</span></a>
    <a href="galeria.html"><span>AUTORAIS</span></a>
    <a href="cliente.html"><span>CLIENTES</span></a>
    <a href="contato.html"><span>ORCAMENTOS</span></a>
    <a href="contato.html"><span>CONTATO</span></a>
    <a href="about.html">INFORMACOES</a>
    <a href="https://www.facebook.com/estercamposfotografia"><i class="
       fa fa-facebook"></i></a>
    <a href="#"><i class="fa fa-twitter"></i></a>
    <a href="#"><i class="fa fa-envelope"></i></a>
  </div>
  <!-- Menu button -->
  <div id="menuToggle"><i class="fa fa-bars navicon img-responsive"></i
     ></div>
</nav>
```

Para o ícone do menu, foi configurado no *css* as seguintes linhas:

```
    #menuToggle {
```

```
        position: absolute;
        top: 3px;
        left: 0;
        right: 10;
        z-index: 11;
        opacity: 1;
        text-align: center;
        font-size: 30px;
        color: #000;
        width: 20px;
        height: 20px;
        line-height: 30px;
        padding: -10px;
        cursor: pointer;
        -webkit-transition: all .1s ease-in-out;
        transition: opacity 0.5s;
        -moz-transition: all .1s ease-in-out;
        -ms-transition: all .1s ease-in-out;
        -o-transition: all .1s ease-in-out;
        transition: all .1s ease-in-out;
    }
```

Ao passar o *mouse* em cima do ícone mudamos a cor do ícone para cinza claro:

```
    #menuToggle:hover {
        color: #ccc;
        -webkit-transition: all .1s ease-in-out;
        -moz-transition: all .1s ease-in-out;
        -ms-transition: all .1s ease-in-out;
        -o-transition: all .1s ease-in-out;
        transition: all .1s ease-in-out;
    }
```

2.4.0.2 TESTE WEBDEVELOPERS

Utilizamos a extensão para navegador web *WebDevelopers* para testar a responsividade desta página. Este teste foi feito utilizando o navegador web *Mozilla Firefox for Linux Mint* versão 49.0.2 e *Web Developer* versão 1.2.11.

Web Developer é um plugin que "adiciona um menu e uma barra com várias ferramentas de desenvolvimento para web"

```
    chrome://web-developer/content/generated/
```

```
        view-responsive-layouts.html#content
```

Abaixo listamos duas imagens que mostram um resultado do teste. Mais testes podem ser realizados durante a apresentação deste projeto.

Figura 5: 321x480

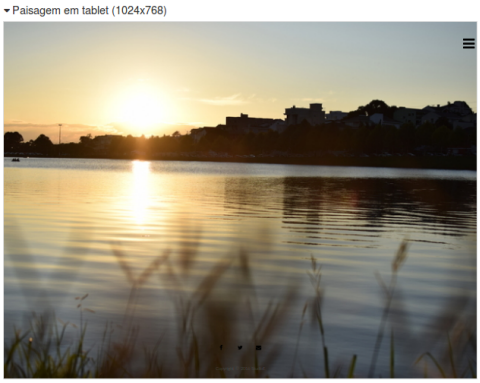

Figura 6: 1024x768

3 EXIGÊNCIAS ESPECÍFICAS

O sistema deve ser implementado contendo as funcionalidades referentes aos tópicos apresentados abaixo. Cada tópico será avaliado separadamente. Para o aluno obter nota máxima em cada tópico, ele precisa utilizar todas as estruturas listadas nos respectivos sub-tópicos. Também serão consideradas as boas práticas de programação em JavaScript, uso adequado de notações e conceitos aprendidos, organização do código e criatividade.

3.1 QUALIDADE DO CÓDIGO

Estes itens foram retirado conforme combinado em sala de aula. Style Guide Strict mode

3.1.1 HINT

Objetivo: mostrar a correção de apenas 5 problemas informados pelo lint ou hint.

Problema 1

Figura 8: Problema 1

Figura 9: Problema 1 - Correção

Problema 2

Problema 3

Figura 10: Problema 2

Figura 11: Problema 2 - Correção

Figura 12: Problema 3

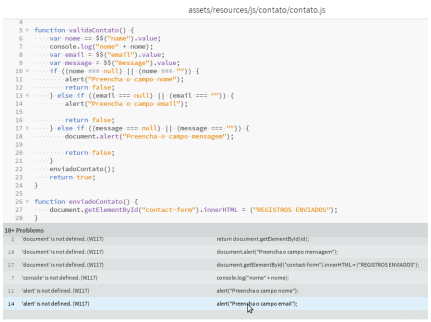

Figura 13: Problema 3 - Correção

Problema 4

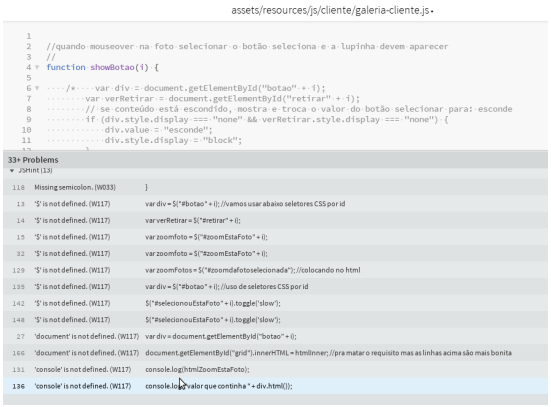

Figura 14: Problema 4

Figura 15: Problema 4 - Correção

3.2 CAIXAS DE DIÁLOGO

3.2.1 PROMPT

Ver arquivo *galeria_cliente.js*

```
1
2 function enviarFotosSelecionadas() {
```

```
3       var confirma = confirm("Confirmar o envio das fotos?!");
4       if (confirma == true) {
5           var email = prompt("Por favor coloque um email para que
                possamos lhe comunicar sobre a entrega: ");
6           //$(this).replaceWith($('<h5>' + this.innerHTML + '</h5>'));
7           var fotosenviadas = { //comecamos armazenando email do cliente
8               email: email,
9               qtdeFotos: jsonClienteX.qtdeFotos
10          };
11          for (var i = 0; i < jsonClienteX.fotos.length; i++) {
12              if (jsonClienteX.fotos[i].escolhido == true) {
13                  fotosenviadas[i] = jsonClienteX.fotos[i].arquivo; //
                        armazenando todos os nomes dos arquivos
14                  console.log("enviando fotos" + i + "estava marcada no
                        localstorage");
15              }
16          }
17          var chaveParaAdmin = "FotosSalvasDe" + sessionStorage.getItem("
                logou");
18          localStorage.setItem(chaveParaAdmin, JSON.stringify(
                fotosenviadas));
19          localStorage.removeItem(sessionStorage.getItem("logou"));
20          alert("Suas fotos foram enviadas VocÃª serÃ¡ redirecionado para
                o inicio");
21          deslogar();
22
23      } else {
24          return;
25      }
26  }
```

3.2.2 ALERT

No arquivo *JavaScript* chamado *contato.js* criamos uma função com nome *validaContato*() conforme segue abaixo para melhor visualização:

```
1   function validaContato() {
2       var nome = $$("name").value;
3       console.log("nome" + nome);
4       var email = $$("email").value;
5       var message = $$("message").value;
6       if ((nome == null) || (nome == "")) {
7           alert("Preencha o campo nome");
```

```
8           return false;
9       } else if ((email == null) || (email == "")) {
10          alert("Preencha o campo email");
11          return false;
12      } else if ((message == null) || (message == "")) {
13          alert("Preencha o campo mensagem");
14          return false;
15      }
16      return true;
17  }
```

Esta função está relacionada a validação dos campos do formulário. E dentro dela utilizamos *alert* para enviar uma mensagem ao visitante da página.

3.2.3 CONFIRM

No arquivo *JavaScript* chamado *galeria_cliente.js*. Criamos uma função com nome *enviarFotosSelecionadas*() conforme segue abaixo para melhor visualização:

```
1  function enviarFotosSelecionadas() {
2      var confirma = confirm("Confirmar o envio das fotos?!");
3      if (confirma == true) {
4          var email = prompt("Por favor coloque um email para que
               possamos lhe comunicar sobre a entrega: ");
5          var fotosenviadas = { //comecamos armazenando email do cliente
6              email: email
7          };
8          for (var i = 0; i < jsonClienteX.fotos.length; i++) {
9              if (jsonClienteX.fotos[i].escolhido == true) {
10                 fotosenviadas[i] = jsonClienteX.fotos[i].arquivo; //
                       armazenando todos os nomes dos arquivos
11                 console.log("enviando fotos" + i + "estava marcada no
                       localstorage");
12             }
13         }
14         var chaveParaAdmin = "FotosSalvasDe" + sessionStorage.getItem("
               logou"); //id do cliente sessionStorage.getItem("logou")
15
16         localStorage.setItem(chaveParaAdmin, JSON.stringify(
               fotosenviadas));
17
18         localStorage.removeItem(sessionStorage.getItem("logou"));
19
```

```
20          alert("Suas fotos foram enviadas VocÃª serÃ¡ redirecionado para
                o inicio");
21          deslogar();
22
23      } else {
24          return;
25      }
26  }
```

Observe que na segunda linha da função temos o solicitado.

3.3 FUNÇÕES

3.3.1 FUNÇÃO ANÔNIMA COM ARGUMENTO

No arquivo *JavaScript* chamado *login.js* criamos uma função anônima que receberá um argumento *i* conforme segue abaixo para melhor visualização:

```
1  var naoLogou = function (i) {
2      var logou;
3      if (i == 23) {
4          sessionStorage.setItem("logou", "1267");
5      }
6  }
```

3.3.2 FUNÇÃO ANÔNIMA SEM ARGUMENTO

3.3.3 FUNÇÃO ANÔNIMA AUTO-EXECUTÁVEL

Ver no arquivo *contato.js*

```
1  (function () {
2      tagDesnegrito();
3  });
```

3.3.4 FUNÇÃO COM NOME

No arquivo *JavaScript* chamado *contato.js* criamos uma função com nome *validaContato*() conforme segue abaixo para melhor visualização:

```
1  function validaContato() {
2      var nome = $$("name").value;
```

```
3       console.log("nome" + nome);
4       var email = $$("email").value;
5       var message = $$("message").value;
6       if ((nome == null) || (nome == "")) {
7           alert("Preencha o campo nome");
8           return false;
9       } else if ((email == null) || (email == "")) {
10          alert("Preencha o campo email");
11          return false;
12      } else if ((message == null) || (message == "")) {
13          alert("Preencha o campo mensagem");
14          return false;
15      }
16      return true;
17  }
```

Esta função está relacionada a validação dos campos do formulário.

3.3.5 FUNÇÃO ANINHADA/INTERNA

Esta função está relacionada a construção de uma *grid* de fotos do cliente, uma galeria de fotos. A função *exibirFotosAoIniciar*() é chamada quando a página é carregada, ela sempre irá verificar antes se o cliente está logado, pois do contrário não poderemos mostrar ao visitante as fotos, isto é a galeria não será carregada.

```
1   function exibirFotosAoIniciar() {
2       var logou = verSeLogado();
3       if (!logou) {
4           return;
5       }
6       var htmlInner = "";
7       for (var i = 0; i < jsonClienteX.fotos.length; i++) {
8           htmlInner += '<li>\n<div class = "caption img-wrapper"
                onmouseover="showBotao(' + i + ');" onmouseout="
                escondeBotaoSelecionar(' + i + ');" style="
                position:relative;">\n<img src="';
9           htmlInner += jsonClienteX.pasta + jsonClienteX.fotos[i].arquivo
                + jsonClienteX.extensao;
10          htmlInner += '">\n<div class="selecionouEstaFoto" id="
                selecionouEstaFoto' + i + '" style="display:none"><span
                class="glyphicon glyphicon-ok"> </span></div>\n <div class="
                zoomEstaFoto" id="zoomEstaFoto' + i + '" style="display:none
                "><a href="#" data-toggle="modal" data-target="#myModal"
```

```
            onclick=zoomFotoLinha(' + i + '); <span class="glyphicon
            glyphicon-zoom-in"> </span></a></div >\n<button id="botao' +
             i + '" style="display:none" class="btn btn-success"
            onclick="alternarEscolhaDaFoto(' + i + ');">Selecionar</b
            utton></div>\n</li>';
    }
    document.getElementById("grid").innerHTML = htmlInner;
}
```

3.3.6 PASSAGEM DE UMA FUNÇÃO COMO PARÂMETRO

Para este item consideremos no arquivo *javascript* da galeria do cliente a função *alternarEscolhaDaFoto*(i), dentro dela veja a linha 4.

```
1  function alternarEscolhaDaFoto(i) {
2      console.log("valor que continha " + div.html());
3      if (div.html() == "Selecionar") {
4          selecionaOuRetiraFoto(i, selecionarFoto);
5      } else {
6          selecionaOuRetiraFoto(i, deselecionarFoto);
7      }
8      //salvando cada as fotos selecionada
9      var chaveCliente = sessionStorage.getItem("logou");
10     localStorage.setItem(chaveCliente, JSON.stringify(jsonClienteX));
11 }
12 var selecionaOuRetiraFoto = function (i, executar) {
13     executar(i);
14 }
```

Na linha 4 temos a função solicitada que recebe por parâmetro a função *selecionarFoto* dentro dela aplica o prâmetro *i* também recebido. Abaixo temos o código da função *selecionarFoto*(i).

```
1  function selecionarFoto(i) {
2      var div = $("#botao" + i); //uso de seletores CSS por id
3      div.toggle('slow').toggle('slow');
4      div.html("Retirar");
5      jsonClienteX.fotos[i].escolhido = true;
6      div.removeClass('btn-success').addClass('btn-warning');
7      $("#selecionouEstaFoto" + i).toggle('slow');
8      div.fadeOut(500);
9      jsonClienteX.qtdeFotos++;
10 }
11
```

```
function deselecionarFoto(i) {
    var div = $("#botao" + i); //uso de seletores CSS por id
    div.html("Selecionar");
    jsonClienteX.fotos[i].escolhido = false;
    div.removeClass('btn-warning').addClass('btn-success');
    $("#selecionouEstaFoto" + i).toggle('slow');
    jsonClienteX.qtdeFotos--;
}
```

3.4 EVENTOS

3.4.1 EVENTO DE CARREGAMENTO DO DOCUMENTO

Para este item verificar no arquivo *cliente.html* as seguintes linhas:

```
    <script>
        $(document).ready(function () {
            exibirFotosAoIniciar();
            $(window).scroll(function () {
                set = $(document).scrollTop() + "px";
                jQuery('#float-banner').animate({
                    top: set
                }, {
                    duration: 1000,
                    queue: false
                });
            });
            console.log("pronto carregada a pÃ¡gina! Vamos trabalhar js
                ");
            new AnimOnScroll(document.getElementById('grid'), {
                minDuration: 0.4,
                maxDuration: 0.7,
                viewportFactor: 0.2
            });
        });
    </script>
```

3.4.2 EVENTO DE MOVIMENTO DO MOUSE

No arquivo *about.html* temos:

```
  <p onmouseover="tagNegrito()" onmouseout="tagDesnegrito()">
```

3.4.3 EVENTO DE TECLADO

Objetivo: - usar charCode ou KeyCode.

3.4.4 EVENTOS DE FORMULÁRIO

Objetivo: onfocus e onblur.

3.4.5 OBJETO EVENT

Obejtivo: Imprimir alguma propriedade do objeto event recebido como parâmetro

3.4.6 PROPAGAÇÃO DE EVENTOS NO MODELO BOLHA

3.5 ACESSO AOS ELEMENTOS DOM DO HTML

3.5.1 VIA ACESSO DIRETO

Pelo id do elemento HTML utilizamos a função *loginCliente*() no arquivo *login.js*

```
1  function loginCliente() {
2      //var email = document.getElementById("email").value;
3      var email = window.email.value;
4      var password = document.getElementById("password").value;
5      console.log("entrou aqui" + email + " e " + password);
6      if (email == "teste@gmail.com" && password == "projeto$$") {
7          naoLogou(23);
8          alert("Obrigada por fazer login voce sera redirecionado (a).");
9          window.location = "cliente.html";
10         return false;
11     } else {
12         tentativas--;
13         alert("Voce tem: " + tentativas + " tentativas;");
14         if (tentativas == 0) {
15             document.getElementById("email").disabled = true;
16             document.getElementById("password").disabled = true;
17             document.getElementById("submit").disabled = true;
18             return false;
19         }
20     }
21 }
```

3.5.2 VIA GETELEMENTBYID()

Ver arquivo *contato.html* a função *enviadoContato*(), conforme código abaixo:

```
function enviadoContato() {
    document.getElementById("contact-form").innerHTML = ("REGISTROS
        ENVIADOS");
}
```

3.5.3 VIA GETELEMENTSBYNAME()

Considerar o arquivo *admin.js*

```
  function insereSelect() {
    console.log("quququq");
    var x = document.getElementsByTagName('select')[0].length;
    document.getElementsByName("selecionandofotos")[0].innerHTML = "
        Quantide de clientes: " + x;
}
```

3.5.4 VIA GETELEMENTSBYTAGNAME()

Considerar o arquivo *admin.js*

```
  function insereSelect() {
    console.log("quququq");
    var x = document.getElementsByTagName('select')[0].length;
    document.getElementsByName("selecionandofotos")[0].innerHTML = "
        Quantide de clientes: " + x;
}
```

3.5.5 VIA SELETORES CSS

O botão enviar da página *cliente.html* é um *banner* que fica seguinto a ação de rolar a página. Para isto adicionamos no arquivo *cliente.html* algumas linhas, dentre elas temos esta:

```
  jQuery('#float-banner').animate({
```

Este *selector* irá se referir ao $id = "float - banner"$, que no arquivo $HTML$ está representado com as linhas abaixo:

```
<div id="float-banner">
```

```
2       <a class="btn btn-default btn-lg btn-sm" href="#" role="button"
            onclick="return enviarFotosSelecionadas()"><span class="
            glyphicon glyphicon-send"></span> ENVIAR </a>
3   </div>
```

Abaixo seguem todas as linhas da função do *JQuery* (*document*).*ready*(*function* () que utilizamos por ser parecida com o *window.onload*, exceto que no *JQuery* não precisamos esperar o carregamento de imagens como no *window.onload* (nesta página isto será útil pois temos muitas fotos).

```
1       <script>
2           $(document).ready(function () {
3               exibirFotosAoIniciar();
4               $(window).scroll(function () {
5                   set = $(document).scrollTop() + "px";
6                   jQuery('#float-banner').animate({
7                       top: set
8                   }, {
9                       duration: 1000,
10                      queue: false
11                  });
12              });
13              console.log("pronto carregada a pÃ¡gina! Vamos trabalhar js
                    ");
14              new AnimOnScroll(document.getElementById('grid'), {
15                  minDuration: 0.4,
16                  maxDuration: 0.7,
17                  viewportFactor: 0.2
18              });
19          });
20      </script>
```

3.6 TRATADORES DE EVENTO

3.6.1 EVENTO INLINE

Objetivo: especificar o tratador de evento inline. Tratador de eventos inline - onmouseover, onmouseout ou onclick. Neste item escolhemos utilizar *onclick*

onclick - Ver no arquivo *galeria_cliente.js* a seguinte linha:

```
1   <button id="botao' + i + '" style="display:none" class="btn btn-success
        " onclick="alternarEscolhaDaFoto(' + i + ');">Selecionar</button>
```

Esta linha irá criar o botão Selecionar. Nela é mostrado uma ação para *onclick* que chama a função $alternarEscolhaDaFoto(i)$ - I i é um parâmetro referente a foto que estará sendo mostrada.

3.6.2 MODO TRADICIONAL

Objetivo: especificar o tratador de evento no carregamento da página HTML no modo tradicional.

3.6.3 ADDEVENTLISTENER

Objetivo: especificar o tratador de evento no carregamento da página HTML com a função addEventListener.

3.6.4 OPERADOR THIS

Objetivo: usar o operador this em funções tratadoras de eventos.

3.7 FORMULÁRIO

Foi criado a página *contato.html*. Nesta página temos um formulário com 3 campos de preenchimento obrigatórios.

3.7.1 VALIDAÇÃO

Objetivo: validação de formulário com onsubmit usando os métodos tradicionais.

Para cumprir o objetivo o formulário *contato.html* faz uso da seguinte maneira:

```
<form id="contact-form" class="validate-form" method="post" onsubmit="
    return validaContato()">
```

O arquivo *JavaScript* chamado *contato.js* contém a função com nome *validaContato*():

```
function validaContato() {
    var nome = $$("name").value;
    console.log("nome" + nome);
    var email = $$("email").value;
    var message = $$("message").value;
    if ((nome == null) || (nome == "")) {
```

```
7           alert("Preencha o campo nome");
8           return false;
9       } else if ((email == null) || (email == "")) {
10          alert("Preencha o campo email");
11
12          return false;
13      } else if ((message == null) || (message == "")) {
14          alert("Preencha o campo mensagem");
15
16          return false;
17      }
18      enviadoContato();
19      return true;
20  }
```

3.7.2 PROPRIEDADE VALUE

Objetivo: ler e escrever em elementos *input* com a propriedade *value*

```
1  $(document).ready(function () {
2      $('input').val('Informacoes enviadas')
3  })
```

3.7.3 INNERHTML

Objetivo: alterar o conteúdo de elementos *div* ou *p*com a propriedade *innerHTML*

Para este item escolhemos alterar o elemento *div* do formulário de *contatos* pois assim que o visitante enviar uma mensagem de contato a estrutura do formulário será substituída por uma frase, conforme código abaixo, definido pela função.

```
1  function enviadoContato() {
2      document.getElementById("contact-form").innerHTML = ("REGISTROS
           ENVIADOS");
3  }
```

3.7.4 CHECKBOX, RADIO OU SELECT

Manipulação de elemento de listagem, como checkbox, radio ou select

3.7.5 ACESSO VIA HIERARQUIA

Acesso aos elementos de um formulário via hierarquia (caminho) de objetos, ou seja, array forms e elements

3.8 OBJETOS NATIVOS

3.8.1 MANIPULAÇÃO DE ARRAY

Usar métodos para manipulação de array - PAra este item observar o Array fotos dentro do objeto *jsonClientX*

```
1   var jsonClienteX = {
2     pasta: "assets/resources/img/clients/cliente/",
3     extensao: ".jpg",
4     qtdeFotos: 0,
5     fotos: [
6         {
7             arquivo: '2',
8             escolhido: false
9         },
10        {
11            arquivo: '20',
12            escolhido: false
13        },
14  ],
15 }
```

3.8.2 MANIPULAÇÃO DE STRING

Usar métodos para manipulação de string

3.9 OBJETOS

3.9.1 CRIAR OBJETO

Criar objeto usando função construtora ou notação literal

```
1   var jsonClienteX = {
2     pasta: "assets/resources/img/clients/cliente/",
3     extensao: ".jpg",
```

```
    fotos: [
        {
            arquivo: '2',
            escolhido: false
        },
        {
            arquivo: '20',
            escolhido: false
        },
        {
            arquivo: '3',
            escolhido: false
        },
        {
            arquivo: '4',
            escolhido: false
        },
        {
            arquivo: '5',
            escolhido: false
        },
        {
            arquivo: '20',
            escolhido: false
        },
        {
            arquivo: '6',
            escolhido: false
        },
        {
            arquivo: '8',
            escolhido: false
        },
        {
            arquivo: '9',
            escolhido: false
        },
        {
            arquivo: '10',
            escolhido: false
        },
        {
            arquivo: '11',
```

```
47              escolhido: false
48          },
49          {
50              arquivo: '12',
51              escolhido: false
52          },
53          {
54              arquivo: '13',
55              escolhido: false
56          },
57          {
58              arquivo: '14',
59              escolhido: false
60          },
61          {
62              arquivo: '15',
63              escolhido: false
64          },
65          {
66              arquivo: '16',
67              escolhido: false
68          },
69          {
70              arquivo: '17',
71              escolhido: false
72          },
73          {
74              arquivo: '18',
75              escolhido: false
76          },
77          {
78              arquivo: '19',
79              escolhido: false
80          },
81    ],
82  }
```

3.9.2 HERANÇA

Usar herança prototipal

3.10 JQUERY

3.10.1 SELETORES CSS

Uso de seletores CSS - id, classe e tag. Os subitem abaixo nos mostram como foram utilizados os seletores solicitados.

3.10.1.1 ID

Verificar a linha 6 abaixo:

```
1       <script>
2          $(document).ready(function () {
3              exibirFotosAoIniciar();
4              $(window).scroll(function () {
5                  set = $(document).scrollTop() + "px";
6                  jQuery('#float-banner').animate({
7                      top: set
8                  }, {
9                      duration: 1000,
10                     queue: false
11                 });
12             });
13             console.log("pronto carregada a pÃ¡gina! Vamos trabalhar js
                   ");
14             new AnimOnScroll(document.getElementById('grid'), {
15                 minDuration: 0.4,
16                 maxDuration: 0.7,
17                 viewportFactor: 0.2
18             });
19         });
20     </script>
```

Observar também abaixo que na linha 2 do arquivo *galeria_cliente.js* temos o uso de seletor *CSS* por *id*.

```
1 function deselecionarFoto(i) {
2     var div = $("#botao" + i); //uso de seletores CSS por id
3     div.html("Selecionar");
4     jsonClienteX.fotos[i].escolhido = false;
5     div.removeClass('btn-warning').addClass('btn-success');
6     $("#selecionouEstaFoto" + i).toggle('slow');
7 }
```

3.10.1.2 CLASSE

Ver linha 6.

```
1 function selecionarFoto(i) {
2     var div = $("#botao" + i); //uso de seletores CSS por id
3     div.toggle('slow').toggle('slow');
4     div.html("Retirar");
5     jsonClienteX.fotos[i].escolhido = true;
6     div.removeClass('btn-success').addClass('btn-warning');
7     $("#selecionouEstaFoto" + i).toggle('slow');
8     div.fadeOut(500);
9 }
```

3.10.1.3 TAG

Ver no arquivo *contato.js* a função faz a troca de cor da letra das tags $< p >$.

```
1   function tagNegrito() {
2     $("p").css("color", "black");
3 }
4
5
6 function tagDesnegrito() {
7     $("p").css("color", "gray");
8
9 }
10
11 (function () { //funÃ§Ã£o auto-executÃ¡vel
12     tagDesnegrito();
13     //se houver algum css negrito remover
14 });
```

3.10.2 SELETORES HIERARQUICOS

Uso de seletores hierarquicos - ancestral/descendente, pai/filho, anterior/proximo

3.10.3 EFEITOS FADE OU SLIDE

Efeito utilizado $fadeout$. Observe na linha 8 da função $selecionarFoto(i)$ do arquivo $galeria\_cliente.js$.

```
1 function selecionarFoto(i) {
2     var div = $("#botao" + i); //uso de seletores CSS por id
3     div.toggle('slow').toggle('slow');
4     div.html("Retirar");
5     jsonClienteX.fotos[i].escolhido = true;
6     div.removeClass('btn-success').addClass('btn-warning');
7     $("#selecionouEstaFoto" + i).toggle('slow');
8     div.fadeOut(500);
9 }
```

3.10.4 TRATADOR DE ALGUM EVENTO

Especificar o tratador de algum evento via jQuery

3.10.5 MANIPULAÇÃO CSS

Objetivo: manipulação do CSS via função *css*() e *addClass*()/*removeClass*()

Utilizamos *addClass* e *removeClass* na função $alternarEscolhaDaFoto(i)$ do arquivo $galeria - cliente.js$. Descrição da função: quando um cliente realizar a seleção de uma foto o botão irá mudar de cor, ficando disponível a opção para desmarcar a foto que acabou de ser selecionada, na função, veja a linha 8. O inverso irá acontecer quando o cliente desmarcar a foto que havia sido selecionada, linha 13.

```
1  function alternarEscolhaDaFoto(i) {
2      var div = $("#botao" + i); //uso de seletores CSS por id
3      console.log("valor que continha " + div.html());
4      if (div.html() == "Selecionar") {
5          selecionaOuRetiraFoto(i, selecionarFoto); //passagem de uma
               funÃ§Ã£o por parÃ¢metro
6      } else {
7          selecionaOuRetiraFoto(i, deselecionarFoto);
8      }
9      //salvando cada as fotos selecionada
10     var chaveCliente = sessionStorage.getItem("logou");
11
12     localStorage.setItem(chaveCliente, JSON.stringify(jsonClienteX));
13 }
```

3.10.6 MANIPULAÇÃO DO CONTEÚDO

Objetivo: manipulação do conteúdo de um *input* e *div* usando *jQuery*.

3.11 WEB STORAGE

Abaixo todos os subitens solicitados.

3.11.1 LOCALSTORAGE

Fazemos uso de *localstorage* na função *alternarEscolhaDaFoto* do arquivo *galeria_cliente.js*. Conforme observamos nas linhas de código abaixo:

```
1   function alternarEscolhaDaFoto(i) {
2     console.log("valor que continha " + div.html());
3     if (div.html() == "Selecionar") {
4         selecionaOuRetiraFoto(i, selecionarFoto); //passagem de uma
              funÃ§Ã£o por parÃ¢metro
5     } else {
6         selecionaOuRetiraFoto(i, deselecionarFoto);
7     }
8     //salvando cada as fotos selecionada
9     var chaveCliente = sessionStorage.getItem("logou");
10
11    localStorage.setItem(chaveCliente, JSON.stringify(jsonClienteX));
12  }
```

Uma chave do cliente, *chaveCliente* foi armazenada juntamente com as fotos que o cliente selecionou.

A função *exibirComFotosMarcadasAnteriormente* faz uso do localstorage também. Desta forma com a finalidade de recuperar os dados armazenados:

```
1
2  function exibirComFotosMarcadasAnteriormente() {
3     //pra carregar com eventuais fotos selecionadass anteriormente
4     var chaveCliente = sessionStorage.getItem("logou");
5     if (!localStorage.getItem(chaveCliente)) {
6         //nÃ£o existe nada no localstorage deste cliente
7         return;
8     }
9     var jsonClienteXTmp = JSON.parse(localStorage.getItem(chaveCliente)
          );
```

```
10      for (var i = 0; i < jsonClienteXTmp.fotos.length; i++) {
11          if (jsonClienteXTmp.fotos[i].escolhido == true) {
12              console.log("foto" + i + "estava marcada no localstorage");
13              var div = $("#botao" + i); //uso de seletores CSS por id
14              div.html("Retirar");
15              jsonClienteX.fotos[i].escolhido = true;
16              div.removeClass('btn-success').addClass('btn-warning');
17              $("#selecionouEstaFoto" + i).toggle();
18              jsonClienteX.qtdeFotos++;
19          }
20      }
21      console.log("exibirComFotosMarcadasAnteriormente" + i + "fim");
22  }
```

Já na função podemos observar a remoção dos itens salvos, isto é resumidamente quando o cliente *clicar* em enviar os dados selecionados vão passar para outra chave e os dados anteriores serão removidos:

```
1   function enviarFotosSelecionadas() {
2     var confirma = confirm("Confirmar o envio das fotos?!");
3     if (confirma == true) {
4         var email = prompt("Por favor coloque um email para que
              possamos lhe comunicar sobre a entrega: ");
5         //$(this).replaceWith($('<h5>' + this.innerHTML + '</h5>'));
6         var fotosenviadas = { //comecamos armazenando email do cliente
7             email: email
8         };
9         for (var i = 0; i < jsonClienteX.fotos.length; i++) {
10            if (jsonClienteX.fotos[i].escolhido == true) {
11                fotosenviadas[i] = jsonClienteX.fotos[i].arquivo; //
                      armazenando todos os nomes dos arquivos
12                console.log("enviando fotos" + i + "estava marcada no
                      localstorage");
13            }
14        }
15        var chaveParaAdmin = "FotosSalvasDe" + sessionStorage.getItem("
              logou"); //id do cliente sessionStorage.getItem("logou")
16
17        localStorage.setItem(chaveParaAdmin, JSON.stringify(
              fotosenviadas));
18        localStorage.removeItem(sessionStorage.getItem("logou"));
19        alert("Suas fotos foram enviadas Voce sera redirecionado para o
              inicio");
```

```
20          deslogar();
21      } else {
22          return;
23      }
24  }
```

3.11.2 SESSIONSTORAGE

Utilizamos no arquivo *login.js* para armazenar a chave *logou*, está chave será testada posteriormente no arquivo *galeria_clientes.js* com a finalidade de verificar se o cliente está logado. Pois somente um cliente só terá acesso a página *cliente.html* se ele tiver feito *login* anteriormente.

```
1  var naoLogou = function (i) {
2      var logou;
3      if (i == 23) {
4          sessionStorage.setItem("logou", "1267");
5      }
6  }
```

arquivo $galeria_clientes.js$

```
1  function verSeLogado() {
2      var estaLogado = sessionStorage.getItem('logou');
3      console.log("esta logado " + estaLogado);
4      if (estaLogado == '1267') {
5
6          return true;
7      } else {
8          alert("VocÃª precisa fazer o login.");
9          // deslogar(); //pra este caso limpar Pra dar certo qdo logar
10         window.location = "index.html";
11         return false;
12     }
13     return false;
14 }
```

A função *verSeLogado*() será chamada dentro da função *exibirFotosAoIniciar*();. Note que esta função é chamada ao carregar o *html*.

```
1    function deslogar() {
2      sessionStorage.removeItem("logou");
3      window.location = "index.html";
4  }
```

3.11.3 LEITURA E ESCRITA DE DADOS SIMPLES

Leitura:

```
function verSeLogado() {
    var estaLogado = sessionStorage.getItem('logou');
    console.log("esta logado " + estaLogado);
    if (estaLogado == '1267') {
        return true;
    } else {
        alert("Voca precisa fazer o login.");
        window.location = "index.html";
        return false;
    }
    return false;
}
```

Escrita:

```
  var naoLogou = function (i) {
    var logou;
    if (i == 23) {
        sessionStorage.setItem("logou", "1267");
    }
}
```

3.11.4 LEITURA E ESCRITA DE JSON

Observe (linha 9) que na função *enviarFotosSelecionadas* temos a escrita e leitura de JSON:

```
1  function enviarFotosSelecionadas() {
2    var confirma = confirm("Confirmar o envio das fotos?!");
3    if (confirma == true) {
4        var email = prompt("Por favor coloque um email para que
             possamos lhe comunicar sobre a entrega: ");
5        var fotosenviadas = {
6            email: email
7        };
8        for (var i = 0; i < jsonClienteX.fotos.length; i++) {
9            if (jsonClienteX.fotos[i].escolhido == true) {
10               fotosenviadas[i] = jsonClienteX.fotos[i].arquivo; //
                     armazenando todos os nomes dos arquivos
```

```
11                  console.log("enviando fotos" + i + "estava marcada no
                        localstorage");
12              }
13          }
14          var chaveParaAdmin = "FotosSalvasDe" + sessionStorage.getItem("
                logou"); //id do cliente sessionStorage.getItem("logou")
15
16          localStorage.setItem(chaveParaAdmin, JSON.stringify(
                fotosenviadas));
17          localStorage.removeItem(sessionStorage.getItem("logou"));
18          alert("Suas fotos foram enviadas Voce sera redirecionado para o
                inicio");
19          deslogar();
20      } else {
21          return;
22      }
23  }
```

4 CÓDIGO

4.1 ABORDAGEM

Todos os arquivos criado estão disponíveis na pasta do projeto que foi entregue ao professor via ambiente *moodle*, conforme solicitado.

5 REFERÊNCIAS

Rich Bradshaw. Cross fading images, November de 2016.

Rich Bradshaw. Cross fading images, Nov. 2016.