Brazilian Journal of Development

# Recomendações para adoção de padrões de usabilidade e responsividade no desenvolvimento de aplicações web

# Recommendations for adopting usability and responsiveness standards in the web applications development

DOI:10.34117/bjdv6n5-075

Recebimento dos originais: 06/05/2020
Aceitação para publicação: 06/05/2020

**Enio de Jesus Pontes Monteiro**
Mestrando em Ciência da Computação pela Universidade Estadual de Campinas
Instituição: Universidade Estadual de Campinas
Endereço: Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP, Brasil
E-mail: e262945@dac.unicamp.br

**Carlos dos Santos Portela**
Doutor em Ciência da Computação pela Universidade Federal de Pernambuco
Instituição: Universidade Federal do Pará, Campus do Tocantins/Cametá
Endereço: Trav. Pe. Antônio Franco, 2617 - Bairro da Matinha, Cametá - PA, Brasil
E-mail: csp@ufpa.br

**Fabricio de Souza Farias**
Doutor em Engenharia Elétrica pela Universidade Federal do Pará
Instituição: Universidade Federal do Pará, Campus do Tocantins/Cametá
Endereço: Trav. Pe. Antônio Franco, 2617 - Bairro da Matinha, Cametá - PA, Brasil
E-mail: fabriciosf@ufpa.br

**Allan Barbosa Costa**
Mestre em Engenharia Elétrica pela Universidade Federal do Pará
Instituição: Universidade Federal do Pará, Campus do Tocantins/Cametá
Endereço: Trav. Pe. Antônio Franco, 2617 - Bairro da Matinha, Cametá - PA, Brasil
E-mail: allancosta@ufpa.br

**Diovanni Moraes de Araújo**
Mestre em Ciência da Computação pela Universidade Federal do Pará
Instituição: Universidade Federal do Pará, Campus do Tocantins/Cametá
Endereço: Trav. Pe. Antônio Franco, 2617 - Bairro da Matinha, Cametá - PA, Brasil
E-mail: diovanni@ufpa.br

Brazilian Journal of Development

**RESUMO**

Atualmente, os dispositivos móveis constituem o principal meio para acessar conteúdo web, pois através de seus *smartphones e tablets* as pessoas estão conectadas quase que 24 horas. No entanto, nem todo site está preparado para a diversidade de dispositivos, tornando a experiência do usuário negativa. Mediante o contexto exposto, esse artigo apresenta um estudo de caso do uso de um guia prático que permite implementar, através do *framework* Bootstrap, as principais características de usabilidade e responsividade. As recomendações deste guia foram aplicadas no desenvolvimento do site de uma Incubadora de Empresas. Espera-se, a partir do seguimento dessas recomendações, que desenvolvedores possam criar aplicações compatíveis com os diversos dispositivos e navegadores disponíveis.

**Palavras-chave:** Aplicações Web, Usabilidade, Responsividade.

**ABSTRACT**

Currently, mobile devices are the main ways to access web content, because people are connected almost 24 hours through their smartphones and tablets. However, not every website is prepared for the diversity of devices, making the user experience negative. In this context, this paper presents a practical guide that allows us to implement the main characteristics of usability and responsiveness through the Bootstrap framework. The guide's recommendations have been applied in the development of a Business Incubator website. We expect, from the follow-up of these recommendations, that developers can create applications compatible with the various devices and browsers available.

**Keywords:** Web Applications, Usability, Responsiveness.

## 1 INTRODUÇÃO

A *web* teve início em 1990, com a invenção da linguagem HTML (*Hypertxt Markup Language*) por Tim Berners Lee, na Suíça (BERNERS-LEE, HENDLER e LASSILA, 2001). A partir de então, tornou-se rapidamente um poderoso meio de compartilhamento de informações. Com a evolução das linguagens, métodos, técnicas e ferramentas de programação, em paralelo, com o crescimento sem precedentes da população usuária dos computadores, foi surgindo a necessidade de se desenvolver sistemas adequados às pessoas com características heterogêneas tanto em relação ao ambiente de acesso que evoluiu de *desktop* para dispositivos móveis, quanto pela acelerada evolução técnica da apresentação das informações para os usuários (FARROCO et al., 2020).

Neste cenário, observou-se a migração de sistemas com muita informação e baixa eficiência para sistemas com a quantidade adequada de conteúdo e elevado poder de

organização do que realmente precisa ser e como deve ser apresentado aos usuários. Mas, nem toda aplicação *web* está preparado para a diversidade de usuários e para o acesso de dispositivos móveis, podendo tornar a experiência do usuário negativa (VALENTIM, OLIVEIRA e CONTE, 2012).

Para evitar as frustrações dos usuários, os *softwares* devem ser preparados para seguir as recomendações de usabilidade, de forma que sejam compreendidos, aprendidos, operados, e atraentes ao usuário (ISO/IEC 9126, 2003). Dentre as recomendações, as aplicações devem ser responsivas, ou seja, precisam se adequar aos diversos formatos de exibição dos dispositivos que acessam seu conteúdo. Neste sentido, existem soluções de mercado que buscam facilitar o processo de desenvolvimento de páginas *web* responsivas, destacando-se o *framework* Bootstrap (2020), que utiliza *HypeTtext Markup Language* (HTML), *Cascading Style Sheets* (CSS) e Javascript para criação dos *layouts* responsivos.

Diante do contexto exposto, este artigo apresenta um guia prático para desenvolvedores que visa auxiliar na implementação de sistemas *web* seguindo as principais características de usabilidade, segundo a norma ISO/IEC 9126, e de responsividade utilizando o Bootstrap. Este guia destina-se tanto a *web* designers (*front-end*), quanto programadores *back-end* e *full-stack* que pretendem criar interfaces usáveis, acessíveis e adaptáveis. A fim de validar a proposta, foi realizado um estudo de caso do desenvolvimento de um sistema *web* de uma Incubadora de Empresas, o qual foi baseado nas recomendações presentes no guia proposto.

O restante desse trabalho está organizado da seguinte forma. Na Seção 2 são apresentados os principais conceitos de usabilidade e responsividade. Na Seção 3 é apresentado o guia de recomendações para adoção de padrões de usabilidade e responsividade no desenvolvimento de aplicações web. Esse guia foi avaliado através de um estudo de caso da sua aplicação no desenvolvimento de um sistema web, conforme relata a Seção 4. Por fim, na Seção 5, são apresentadas as principais contribuições desse trabalho e as próximas etapas a serem desenvolvidas.

## 2 USABILIDADE E RESPONSIVIDADE

### 2.1 USABILIDADE NA ISO/IEC 9126

A ISO/IEC 9126 (2003) define usabilidade como a capacidade do *software* de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificas. Dessa forma, define como atributos de qualidade as seguintes características:

- **Inteligibilidade:** Capacidade do *software* de possibilitar ao usuário compreender se o sistema é apropriado e como ele pode ser usado para tarefas e condições de uso específicas. Essa inteligibilidade dependerá da documentação e das impressões inicias oferecidas pelo *software*;
- **Apreensibilidade:** Capacidade do produto de *software* de possibilitar ao usuário aprender sua aplicação;
- **Operacionalidade:** Capacidade do *software* de possibilitar ao usuário operá-lo e controlá-lo. Corresponde aos atributos de controlabilidade, tolerância a erros e conformidade com as expectativas do usuário;
- **Atratividade:** Capacidade do *software* de ser atraente ao usuário. Refere-se a atributos de software que possuem a intenção de tornar o *software* mais atraente para o usuário, como o uso de cores e da aplicação do projeto gráfico.

Como possíveis usuários a norma inclui operadores, usuários finais e usuários indiretos que sejam dependentes ou estejam sob influência do uso do *software*, além de todos os ambientes que o *software* pode afetar. Deve-se considerar o ambiente onde usuários estão sendo preparados para o uso do produto e o ambiente onde será feita a avaliação de resultados do uso do *software*.

## 2.2 WEB DESIGN RESPONSIVO

A responsividade é definida como a qualidade do que é responsivo. No contexto *web*, os sistemas, portais, jogos ou qualquer tecnologia visualizada em uma tela, devem manter um visual semelhante e, mais importante, igualmente agradável de ser utilizado em qualquer tamanho e qualidade de tela em que seja exibido (SILVA, M., 2016).

No contexto do desenvolvimento *web*, Silva, A. (2014) define *design* responsivo como um princípio cujo objetivo é adaptar o *layout* das páginas a qualquer dispositivo, tela e resolução, com a finalidade de garantir uma boa experiência do usuário, possibilitando navegação e leitura confortáveis, sem comprometer o conteúdo.

De maneira geral, utilizam-se quatro tecnologias para a implementação de um *design* responsivo: *layouts* fluídos, *medias queries*, recursos flexíveis (imagens, fontes e etc.) e a *meta tag viewport*.

Brazilian Journal of Development

## 2.3 ATRIBUTOS DO DESIGN RESPONSIVO

### *2.3.1 Fundação Flexível*

Os *layouts* fluídos são aqueles que acompanham o tamanho da tela, apenas aumentando e diminuindo, não trocando a estrutura (SILVA, M., 2014). Eles adotam medidas flexíveis (%) para larguras ao invés de *pixels*. A recomendação para um *layout* com medidas flexíveis é a proporção que existe entre os elementos.

Assim, deve-se observar a relação entre os elementos do *layout* em vez de seus tamanhos fixos (SILVA, A., 2014). Para converter um *layout* que é baseado em *pixel* em um proporcional, e que seja fluído, utiliza-se a seguinte fórmula (equação 1):

| | |
|---|---|
| $$\boldsymbol{Resultado = \frac{objeto}{contexto}}$$ | (1) |

Por exemplo, cria-se uma *div* de largura 1024 *pixels*. Dentro dela, uma coluna com 240 *pixels* de largura. A coluna é, portanto, filha do *container*. Então, pega-se o valor em *pixels* da coluna (*objeto* = 240px) e divide-se pelo valor em *pixels* do elemento pai (*contexto* = 1024px). O resultado encontrado é 0,234375. Em seguida, anda-se duas casas para a direita com a vírgula e, acrescenta-se um ponto, assim obtendo-se o valor 23.4375. Portanto, esta é a correspondência dessa coluna em porcentagem: 23.4375%.

Outra abordagem adotada para a criação de *layouts* fluídos é a utilização de *grids* flexíveis. Os *grids* são um conjunto de linhas bases que fornecem uma estrutura para o *layout* de páginas *web* (SILVA, A., 2014). Esse tipo de sistema, geralmente divide o site em colunas de mesma largura, onde se dispõe o conteúdo, normalmente utilizando *grids* de até 12 colunas. *Frameworks front-end*, como o Bootstrap, utilizam essa abordagem para trabalhar com responsividade das suas páginas.

A criação de um site apenas com a ideia de *layout* fluído é possível, se todas as medidas forem definidas com valores relativos. Desse modo, o *design* se adapta a todas as resoluções. No entanto, podem existir cenários que necessitem de ajustes específicos no *layout*. Para isso, deve-se adotar a opção de *Media Queries*.

### *2.3.2 Media Queries*

As *Medias Queries* foram especificadas pelo CSS3, sendo uma evolução das *Media Types* do CSS2. Elas são utilizadas para aplicar mudanças nas especificações do CSS para uma melhor experiência do usuário.

Segundo Silva, M. (2014), *Media Queries* dispõem uma folha de estilo específica para um determinado dispositivo, por meio da consulta e identificação das características para o qual a aplicação está sendo disponibilizada. As *Media Queries* definem qual bloco do CSS é mais adequado para um determinado tamanho de tela.

O ponto ou intervalo onde é adicionado a *media query* é chamado de *breakpoint*. Os *breakpoints* são pontos nos quais o *layout* se readapta para se ajustar a uma nova janela.

### *2.3.3 Imagens Adaptáveis*

As imagens são formadas por *bitmaps*, que por sua vez são formados por *pixels*. O *pixel* representa um ponto na imagem que está no seu monitor, tela do celular e etc. A junção de blocos de *pixels* (normalmente em cores semelhantes ou próximas) formam um *bitmap* (ALURA, 2018).

Por serem formados *pixel* a *pixel*, se tiverem seu tamanho aumentado, os arquivos *bitmaps* sofrem distorções consideráveis, pois cada ponto é transformado em blocos maiores, preenchidos com *pixels* de cores intermediárias que, consequentemente, deixam a imagem deformada por conta do efeito causado pela interpolação entre os pixels. Os formatos mais comuns que utilizam esse modelo são: jpeg, png, gif, tiff, etc.

Para evitar o problema, o ideal seria utilizar uma fórmula para redimensionar proporcionalmente a mesma imagem para manter a qualidade. Neste sentido, aplica-se a vetorização que ao invés de "pintar" *pixel* a *pixel*, utiliza fórmulas matemáticas para calcular as formas criadas, e assim não perder sua proporção e qualidade.

Na vetorização são realizados cálculos para cada forma criada. No entanto, existem diversas ferramentas que podem ser utilizadas nesse processo, como o *software* livre Inskape (https://inkscape.org/pt-br). A utilização de ferramentas para a criação de formas vetorizadas faz uma grande diferença, pois recomenda-se utilizar vetores que redimensionam automaticamente, isto é, independente de quanto *zoom* for dado na imagem. Neste sentido, um ícone feito em imagem pode perder qualidade, dependendo do tamanho da tela em que for visualizado, mas com um vetor esse problema não ocorrerá.

Outra técnica utilizada para contrair e expandir imagens consiste em definir, através de uma linha CSS, uma largura máxima de 100% para todas as imagens, fazendo com que todas elas sejam redimensionadas automaticamente e proporcionalmente de acordo com a largura do *container* em que se encontram.

No entanto, a utilização dessa técnica não é a ideal, visto que, se a resolução da imagem for menor que a do dispositivo, a mesma será alongada. Por outro lado, diminuir uma imagem com alta resolução via código torna a página pesada e fornece ao usuário uma imagem maior do que ele necessita.

Diante do exposto, a solução é utilizar *media queries*, pois permitem fornecer uma imagem ideal para cada contexto, ou seja, versões diferentes do mesmo arquivo de acordo com as características do dispositivo.

### *2.3.4 Meta Tag Viewport*

O *viewport* é a área branca da janela quando se abre o *browser*. Assim, o *viewport* sempre vai ter o tamanho da janela. Mas a forma como os elementos são renderizados vai depender bastante do dispositivo (EIS, 2011).

Antes dos *tablets* e *smartphones*, as páginas *web* eram projetadas apenas para telas de computador, e normalmente, eram páginas estáticas com um tamanho fixo. Então, quando se passou a utilizar outros dispositivos para navegar na Internet, essas tornaram-se grandes demais para se ajustarem as suas janelas de visualização.

Uma solução encontrada para esse problema foi fazer com que os navegadores desses dispositivos reduzissem toda a página para que essas se ajustassem a janela em que estavam sendo visualizadas. Porém, essa não foi a solução ideal, mas sim, uma solução temporária para o problema.

Para solucionar essa problemática, o HTML5 adicionou um método para que os *web designers* assumissem o controle da *viewport*, através da *tag* <meta>. *Meta tags* servem para descrever informações sobre a página. Sendo assim, a *meta tag viewport* informa ao navegador o tamanho de tela disponível para exibir o site, possibilitando o controle de *zoom* do dispositivo. A *meta tag viewport* possui os atributos *name* e *content*.

No *content*, podem ser definidos os seguintes parâmetros e valores:

- *width/height*: definem a largura e altura da tela em que o site será exibido;
- *initial-scale*: define o zoom inicial, de 0 a 10;
- *user-scalable*: ativa ou desativa o zoom, recebendo os valores yes ou no;
- *minimum-scale/maximum-scale*: definem o limite permitido de zoom da página, de 0.25 a 10.

Brazilian Journal of Development

O parâmetro *width=device-width* informa ao navegador que o tamanho da *viewport* é igual ao tamanho da tela do dispositivo utilizado, possibilitando a renderização correta das proporções da página e, a *initial-scale*=1.0 define o nível de *zoom* quando a página é carregada pela primeira vez no navegador.

## 3 GUIA DE RECOMENDAÇÕES

### 3.1 OBJETIVO

O principal objetivo do guia proposto é apoiar a etapa de desenvolvimento de aplicações *web* que contemplem padrões de responsividade, ou seja, se adequem as variedades de dispositivos. Adicionalmente, que sigam padrões de usabilidade segundo a norma ISO/IEC 9126 (2003).

Para isso, este guia fornece uma série de recomendações que auxiliam no desenvolvimento de interfaces responsivas, abordando conceitos, técnicas de programação e a utilização de algumas ferramentas. Por não serem regras e sim recomendações, não há a obrigatoriedade em seguir integralmente o conteúdo contido neste guia, ficando a critério do desenvolvedor escolher adotar uma ou mais dessas recomendações.

Este guia, não tem como objetivo ensinar a elaboração de interfaces, marcações HTML, uso e criação de folhas de estilos com CSS desde o início. Portanto, é recomendável que o usuário já possua conhecimentos básicos no desenvolvimento de páginas *web* adotando essas tecnologias, para que possa extrair o máximo de conhecimento e aplicá-los em seus projetos. Este guia também se encontra disponível *online* em *http://lafoca.com.br/guiaweb/*.

### 3.2 BOOTSTRAP

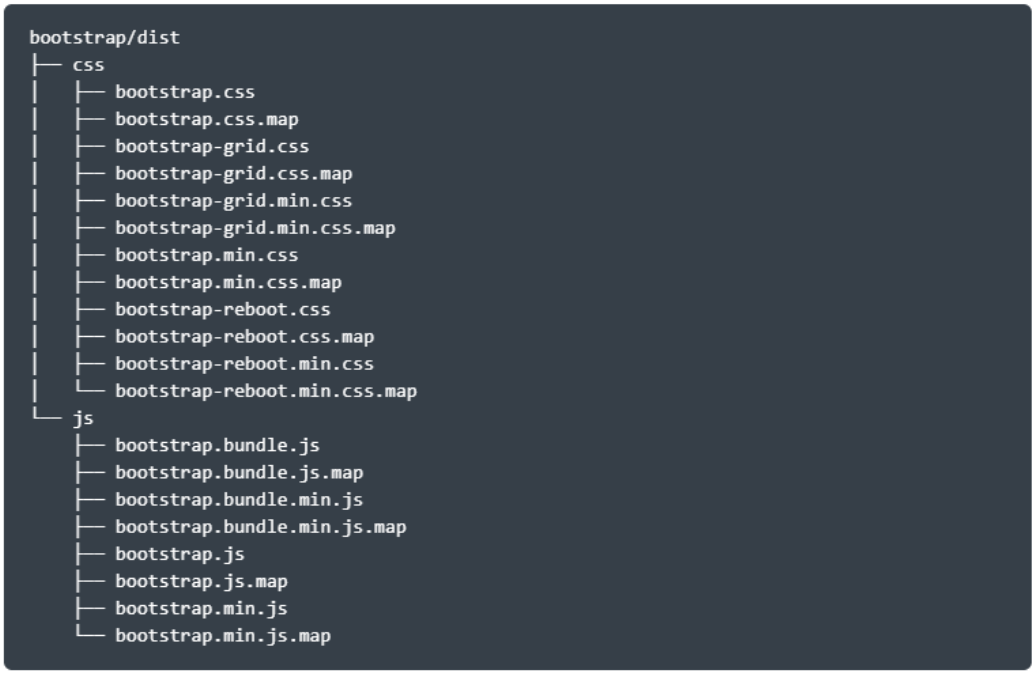
O Bootstrap é um *framework* criado por Mark Otto e Jacob Thornton, ambos engenheiros do Twitter, sendo lançado no dia 19 de agosto de 2011. Desde então, tornou-se o mais popular *framework* JavaScript, HTML e CSS para desenvolvimento de sites e aplicações *web* responsivas e alinhadas com o paradigma *mobile first*. Neste paradigma, a responsividade é focada primeiramente para dispositivos *mobile*, e depois para dispositivos com resoluções maiores.

Neste guia, utilizou-se a versão Bootstrap 4, disponível para *download* em *getbootstrap.com*. Depois de realizar o *download*, deve-se extrair os arquivos CSS e JavaScript para uma pasta onde os arquivos do projeto *web* devem ser mantidos.

Brazilian Journal of Development

Pode-se renomear a pasta que foi extraída com o nome do projeto *web* a ser desenvolvido. Se já tiver um servidor local, pode-se instalar nele. Após instalar o Bootstrap, a seguinte estrutura apresentada na Figura 1 estará disponível.

Figura 1 - Estrutura de Arquivos do Bootstrap




Fonte: Bootstrap (2020).

Na pasta CSS ficam os arquivos dos estilos visuais dos componentes do Bootstrap. Para usar tudo que o Bootstrap fornece, deve-se importar o arquivo *bootstrap.css* ou o arquivo *bootstrap.min.css*. Caso seja necessário usar apenas o *Grid System*, pode-se importar o arquivo *bootstrap-grid.css* ou o arquivo *bootstrap-grid.min.css*.

Na pasta JS ficam os arquivos com as funções dos componentes do Bootstrap como, por exemplo, os Menus. Para usar todos os componentes que o Bootstrap possui, basta importar o arquivo *bootstrap.js* ou o arquivo *bootstrap.min.js*. Cada pasta possui uma versão padrão (os arquivos *.css e *.js), e uma versão "minificada" (os arquivos *.min.css e *.min.js) de cada *script*.

A recomendação é que se utilize a versão padrão quando estiver desenvolvendo o projeto para ter acesso ao código-fonte do Bootstrap e deve-se usar a versão "minificada" (ou *minified*) quando for colocar o projeto *online* (em produção), pois o arquivo *minified* é bem menor que o padrão e economiza na utilização de banda.

Brazilian Journal of Development

## 3.3 RECOMENDAÇÕES PARA USABILIDADE

### *3.3.1 Inteligibilidade*

No que diz respeito à Inteligibilidade, o guia recomenda a definição de um Manual de Ajuda (*Help*) ou FAQs (*Frequently Asked Questions* - Perguntas Frequentes), o uso de ícones e cores dos botões (verde, amarelo e vermelho, por exemplo), como mostra a Figura 2, nomes intuitivos (padronizados) para os menus e botões, dentre outros que permitam gerar um bom entendimento aos usuários.

Figura 2 - Uso de Cores e Ícones nos Botões


Fonte: Elaborado pelo Autor (2020).

Na Figura 3 é apresentado o código HTML com as classes do Bootstrap e da *Font Awesome* para a criação e estilização destes botões e ícones.

Figura 3 - Código HTML com as Classes do Bootstrap e da *Font Awesome* para os Botões e Ícones

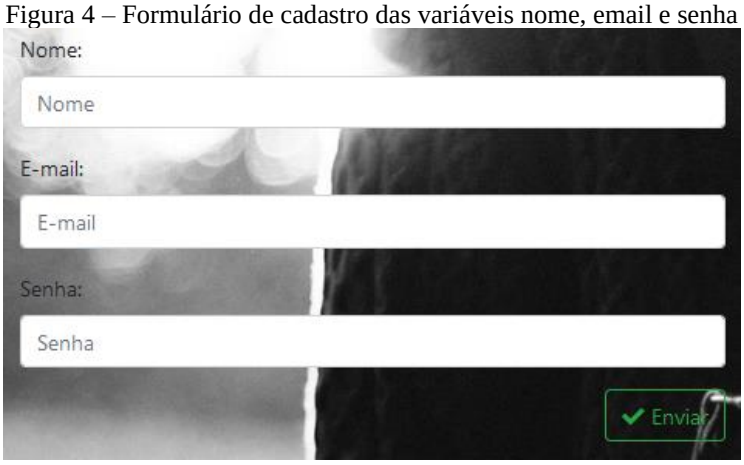
```
<div class="container">
    <div class="row">
        <div class="col">
            <a href="#" class="btn btn-success fas fa-check"> Cadastrar</a>
            <a href="#" class="btn btn-danger fas fa-trash-alt"> Excluir</a>
            <a href="#" class="btn btn-warning fas fa-pencil-alt"> Editar</a>
        </div>
    </div>
</div>
```

Fonte: Elaborado pelo Autor (2020).

### *3.3.2 Apreensibilidade*

Para que uma aplicação *web* atenda a característica de Apreensibilidade, recomenda-se o uso de mensagens de *feedback*, uso de analogias e metáforas nos ícones. Um exemplo é usar o ícone de lixeira como metáfora de exclusão e o formulário como uma analogia ao cadastro em ficha de papel, como mostra a Figura 4.

Figura 4 – Formulário de cadastro das variáveis nome, email e senha




Fonte: Elaborado pelo Autor (2020).

Na Figura 5 é apresentado o código HTML com as classes do Bootstrap e da *Font Awesome* para a criação e estilização do formulário apresentado na Figura 4.

Figura 5 - Código HTML com as Classes do Bootstrap e da *Font Awesome* para Formulários

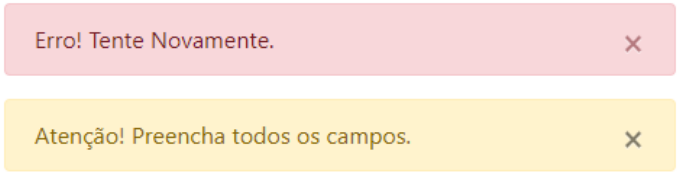
```
<form method="POST" action="cadastro.php" class="form">
    <div class="form-group">
        <label for="nome" class="control-label">Nome: </label>
        <input id="nome" type="text" name="nome" class="form-control" placeholder="Nome" autofocus required>
    </div>
    <div class="form-group">
        <label for="email" class="control-label">E-mail: </label>
        <input id="email" type="email" name="email" class="form-control" placeholder="E-mail" required>
    </div>
    <div class="form-group">
        <label for="senha" class="control-label">Senha: </label>
        <input id="senha" type="password" name="senha" class="form-control" placeholder="Senha" required>
    </div>
    <button type="submit" value="cadastrar" class="btn btn-outline-success float-right">
        <i class="fas fa fa-check"></i>
        Enviar
    </button>
</form>
    </div>
```

Fonte: Elaborado pelo Autor (2020).

### *3.3.3 Operacionalidade*

Para implementar a Operacionalidade, o guia recomenda ao desenvolvedor realizar tratamento de erros, disponibilizar caminhos alternativos para realizar a mesma operação, usar mensagens de erro, como exemplificado na Figura 6, realizar correção ortográfica, usar máscaras de entrada, dentre outras funcionalidades que auxiliem o usuário na realização das suas operações.

Figura 6 - Mensagens de Erro e Alerta




Fonte: Elaborado pelo Autor (2020).

Na Figura 7 é ilustrado o código HTML com as classes do Bootstrap para a criação e estilização de mensagens que auxiliam o usuário em suas operações de controle de erro, alerta ou mensagem de *feedback* positivo.

Figura 7 - Código HTML com as Classes do Bootstrap para a Criação e Estilização de Mensagens

```
<div class="container">
    <div class="row">
        <div class="col">
            <div class="alert alert-danger alert-dismissible fade show"
            role="alert">
                <button type="button" class="close"
                data-dismiss="alert" arial-label="Fechar">
                    <span aria-hidden="true">&times;</span>
                </button>
                Erro! Tente novamente.
            </div>
            <div class="alert alert-warning alert-dismissible fade show"
            role="alert">
                <button type="button" class="close"
                data-dismiss="alert" arial-label="Fechar">
                    <span aria-hidden="true">&times;</span>
                </button>
                Atenção! Preencha todos os campos.
            </div>
        </div>
    </div>
</div>
```

Fonte: Elaborado pelo Autor (2020).

### *3.3.4 Atratividade*

A atratividade pode ser contemplada através do uso harmonizado de cores na interface do sistema (a natureza do projeto gráfico), como exemplifica a Figura 8, até a adequação das informações prestadas para o usuário.

Brazilian Journal of Development

Figura 8 - Barra de Navegação com o Uso Harmonizado de Cores


Fonte: Elaborado pelo Autor (2020).

Na Figura 9 é apresentado o código HTML com as classes do Bootstrap e da *Font Awesome* para a criação e estilização de Barras de Navegação.

**Figura 9 - Código HTML com as Classes do Bootstrap e da Font Awesome para as Barras de Navegação**

```
<nav class="navbar navbar-dark bg-primary navbar-expand-lg">
    <a class="navbar-brand" href="#">Linguagens</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
    data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown"
    aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
        <ul class="navbar-nav justify-content-end">
          <li class="nav-item">
            <a class="nav-link" href="#"><i class="fab fa-html5"></i> HTML5</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#"><i class="fab fa-css3-alt"></i> CSS3</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#"><i class="fab fa-js"></i> JavaScript</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#"><i class="fab fa-php"></i> PHP</a>
          </li>
        </ul>
    </div>
</nav>
```

**Fonte:** Elaborado pelo Autor (2020).

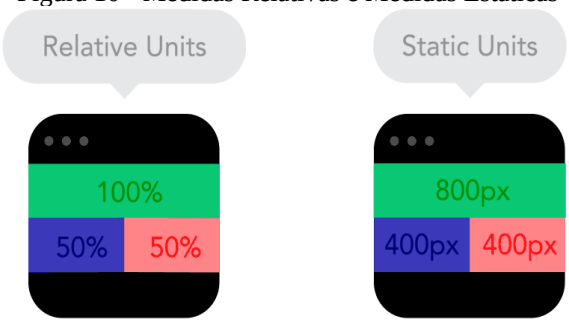
## 3.4 RECOMENDAÇÕES PARA DESIGN RESPONSIVO

### *3.4.1 Fundação Flexível*

Os *layouts* fluídos adotam medidas flexíveis (%) para larguras ao invés de *pixels* (px), conforme comparam as Figuras 10 e 11. Nos *layouts* com medidas em porcentagem as imagens se adequam de forma relativa as telas de apresentação, enquanto que os *layouts* em *pixels* não permitem o ajuste adequado por apresentarem medidas estáticas.

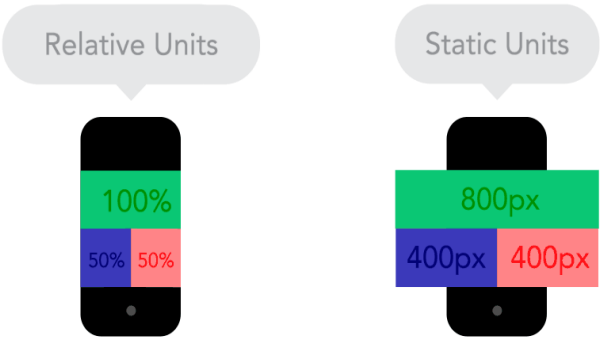
Brazilian Journal of Development

Figura 10 - Medidas Relativas e Medidas Estáticas




Fonte: FROONT (2018).

Figura 11 - Medidas Relativas e Medidas Estáticas em Telas Pequenas




Fonte: FROONT (2018).

Adotando o exemplo da Figura 10, o *layout* tem 800 *pixels* de largura, sendo esse o tamanho máximo do *layout*. Tudo contido nele será proporcional. Ao observarmos o *layout* de exemplo, percebe-se que o cabeçalho ocupa toda a largura do *layout*. Logo, o resultado da aplicação da fórmula será: 800/800 = 1 (100%).

Um exemplo de código HTML para criar o cabeçalho é ilustrado na Figura 12.

Figura 12 - Código HTML para a Criação do Cabeçalho

```
<div class="cabecalho">
    <!-- conteúdo do cabeçalho -->
</div>
```

Fonte: Elaborado pelo Autor (2020).

Para especificar que a largura do cabeçalho terá de 100% de largura, utiliza-se um código CSS como ilustrado na Figura 13.

Figura 13 - Código CSS para Definir a Largura do Cabeçalho como 100%

```css
.cabecalho {
    width: 100%;
    /*outras declarações de estilos*/
}
```

Fonte: Elaborado pelo Autor (2020).

As colunas da esquerda e da direita têm 400 *pixels* de largura dentro dos 800 *pixels* total. Para conseguir uma medida proporcional, utiliza-se a mesma fórmula, alterando apenas os valores: 400/800 = 0,5 (50%).

Um exemplo de código HTML para criar a coluna da esquerda é ilustrado na Figura 14.

Figura 14 - Código HTML para a Criação da Coluna da Esquerda

```html
<div class="left-side">
    <!-- conteúdo da coluna esquerda-->
</div>
```

Fonte:  Elaborado pelo Autor (2020).

Para especificar que a largura da coluna esquerda terá 50% de largura, utiliza-se um código CSS como ilustrado na Figura 15.

Figura 15 - Código CSS para Definir a Largura da Coluna da Esquerda como 50%

```css
.left-side {
    width: 50%;
    /*outras declarações de estilos*/
}
```

Fonte: Elaborado pelo Autor (2020).

Um exemplo de código HTML para criar a coluna da direita é ilustrado na Figura 16.

Figura 16 - Código HTML para a Criação da Coluna da Direita

```html
<div class="right-side">
    <!-- conteúdo da coluna direita -->
</div>
```

Fonte: Elaborado pelo Autor (2020).

Para especificar que a largura da coluna direita terá 50% de largura, utiliza-se um código CSS como ilustrado na Figura 17.

Brazilian Journal of Development

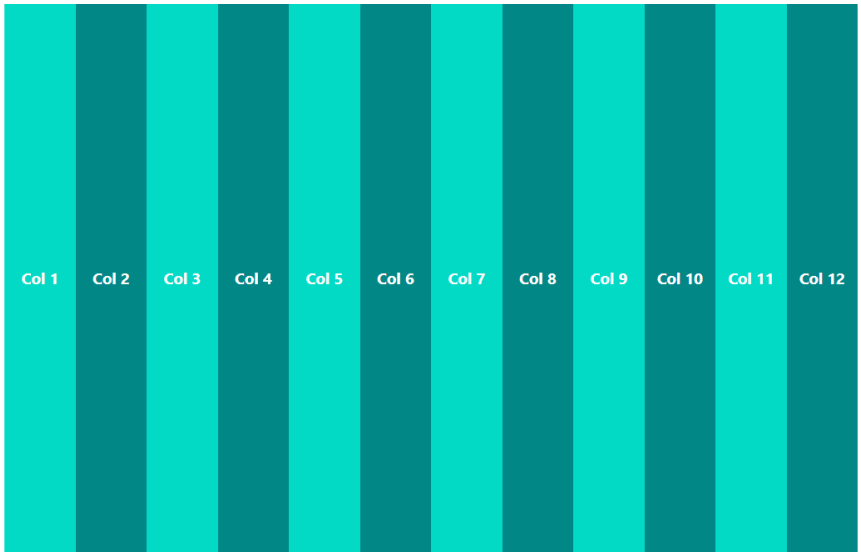
Figura 17 - CSS para Definir a Largura da Coluna da Direita como 50%

```css
.right-side {
    width: 50%;
    /*outras declarações de estilos*/
}
```

(Elaborado pelo Autor).

A utilização de *grids* flexíveis no Bootstrap adota 12 colunas, conforme ilustra a Figura 18.

Figura 18 - Representação de um Sistema de *Grid* de 12 Colunas




Fonte: Elaborado pelo Autor (2020).

A Figura 19 apresenta um exemplo de código HTML para a criação do sistema de *grid* do Bootstrap.

Brazilian Journal of Development

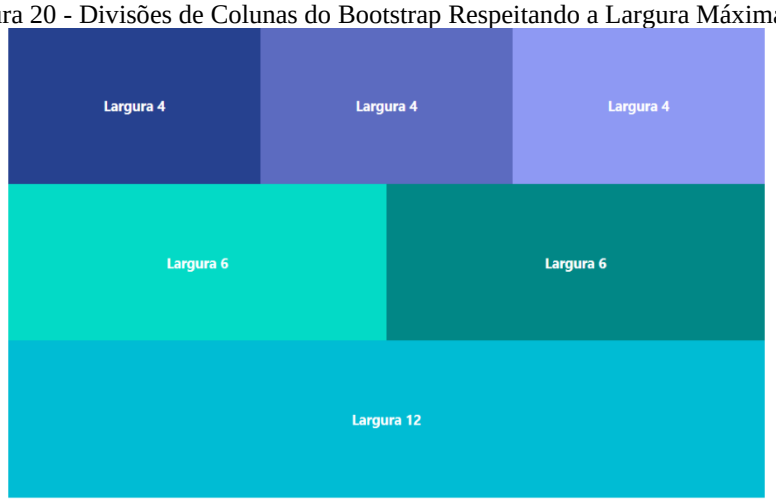
Figura 19 - Código HTML para a Criação do Sistema de *Grid* do *Bootstrap*

```
<div class="container">
    <div class="row">
        <div class="col-sm-1">Col 1</div>
        <div class="col-sm-1">Col 2</div>
        <div class="col-sm-1">Col 3</div>
        <div class="col-sm-1">Col 4</div>
        <div class="col-sm-1">Col 5</div>
        <div class="col-sm-1">Col 6</div>
        <div class="col-sm-1">Col 7</div>
        <div class="col-sm-1">Col 8</div>
        <div class="col-sm-1">Col 9</div>
        <div class="col-sm-1">Col 10</div>
        <div class="col-sm-1">Col 11</div>
        <div class="col-sm-1">Col 12</div>
    </div>
</div>
```

Fonte: Elaborado pelo Autor (2020).

A lógica para qualquer divisão é a mesma, podendo criar colunas de larguras diferentes, porém, sempre respeitando que o resultado das somas das larguras seja sempre 12, conforme exemplo da Figura 20.

Figura 20 - Divisões de Colunas do Bootstrap Respeitando a Largura Máxima 12




Fonte: Elaborado pelo Autor (2020).

A Figura 21 demonstra um exemplo de código HTML para a criação do sistema de *grid* do Bootstrap com diferentes formas de agrupar as colunas.

Figura 21 - Código HTML para a Criação de Várias Colunas com o *Bootstrap*

```
<div class="container">
    <div class="row">
        <div class="col-sm-4">Largura 4</div>
        <div class="col-sm-4">Largura 4</div>
        <div class="col-sm-4">Largura 4</div>
    </div>
    <div class="row linha">
        <div class="col-sm-6">Largura 6</div>
        <div class="col-sm-6">Largura 6</div>
    </div>
    <div class="row linha">
        <div class="col-sm-12">Largura 12</div>
    </div>
</div>
```

Fonte: Elaborado pelo Autor (2020).

### *3.4.2 Media Queries*

As *Media Queries* definem qual bloco do CSS é mais adequado para um determinado tamanho de tela de dispositivo, conforme apresenta um exemplo na Figura 22.

Figura 22 - *Media Query* Especificando em quais Tamanhos de Telas Aplicar o Estilo CSS

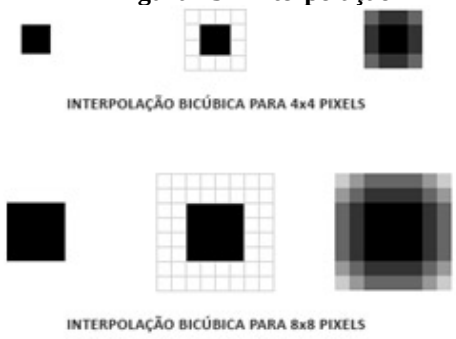
```
@media(min-width: 768px) and (max-width: 1024px) {
    /*Declaração de estilos*/
}
```

Fonte: Elaborado pelo Autor (2020).

### *3.4.3 Imagens Adaptáveis*

Arquivos *bitmaps*, se tiverem seu tamanho aumentado, sofrem distorções consideráveis, deixando a imagem deformada. Isso é chamado de interpolação, conforme exemplifica a Figura 23.

**Figura 23 - Interpolação**

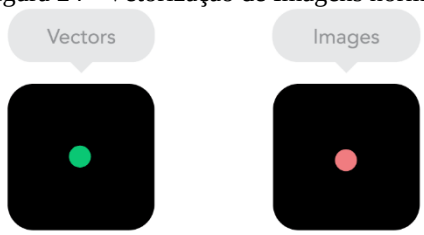



**Fonte:** ALURA (2018).

Brazilian Journal of Development

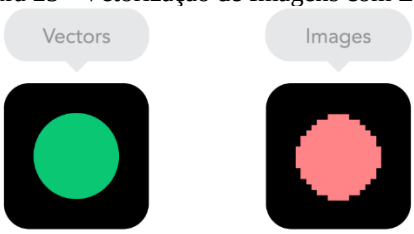
Neste sentido há o método de vetorização que utiliza fórmulas matemáticas para calcular as formas criadas, e assim não perder sua proporção e qualidade. Exemplos de comparação entre essas duas técnicas são apresentados nas Figuras 24 e 25.

Figura 24 - Vetorização de Imagens normal




Fonte: FROONT (2018).

Figura 25 - Vetorização de Imagens com *Zoom*




Fonte: FROONT (2018).

Outra opção, é utilizar a *Font Awesome*, mostrada na Figura 26. Enquanto que a Figura 27 demonstra um exemplo de código HTML para adicionar o ícone home.

Figura 26 - Ícone Home da *Font Awesome*

Fonte: Elaborado pelo Autor (2020).

Figura 27 - Código HTML para Adicionar o Ícone Home da *Font Awesome*

```
<div>
    <i class="fas fa-home fa-5x"></i>
</div>
```

Fonte: Elaborado pelo Autor (2020).

Também pode-se utilizar outra técnica para contrair e expandir imagens, que consiste em definir, através de uma linha CSS, uma largura máxima de 100% para todas as imagens, conforme mostra a Figura 28.

Figura 28 - Código CSS para Definir uma Largura Máxima para Imagens

```css
img {
    max-width: 100%;
}
```

Fonte: Elaborado pelo Autor (2020).

A Figura 29 mostra a aplicação desse CSS no código HTML, fazendo com que todas as imagens sejam redimensionadas automaticamente e proporcionalmente de acordo com a largura do *container* em que se encontra.

Figura 29 - Código HTML para Adicionar uma Imagem

```html
<figure style="width: 304px; height: 228px;">
    <img src="images/html5-topper.png" alt="logo HTML5">
</figure>
```

Fonte: Elaborado pelo Autor (2020).

O Bootstrap trabalha de forma bem simples com o redimensionamento de imagens. Adicionando a classe “img-fluid” na *tag* img, a imagem se ajusta automaticamente nas especificações (largura/altura) do elemento que esteja inserida, como apresenta a Figura 30.

Figura 30 - Código HTML com a classe img-fluid do Bootstrap

```html
<figure style="width: 350px; ">
    <img src="images/html5-topper.png" alt="HTML5" class="img-fluid">
</figure>
```

Fonte: Elaborado pelo Autor (2020).

No entanto, a recomendação principal do guia é utilizar *media queries*, sendo que elas permitem fornecer uma imagem ideal para cada contexto, versões diferentes do mesmo arquivo de acordo com as características do dispositivo, conforme mostra o CSS da Figura 31.

Figura 31 - Código CSS Definindo a Imagem Ideal para Diferentes Larguras de Telas

```css
#banner {
    background: url('images/grande.jpg');
}

@media(max-width: 768px) {
    #banner {
        background: url('images/media.jpg');
    }
}

@media(max-width: 320px) {
    #banner {
        background: url('images/pequena.jpg');
    }
}
```

Fonte: Elaborado pelo Autor (2020).

### *3.4.4 Meta Tag Viewport*

A *meta tag viewport* informa ao navegador o tamanho de tela disponível para exibir o site, assim permitindo o controle de *zoom* do dispositivo. Deste modo, a *meta tag viewport* assume o formato apresentado na Figura 32.

Figura 32 - Formato da Meta Tag Viewport

```html
<head>
    <meta name="viewport" content="">
</head>
```

Fonte: Elaborado pelo Autor (2020).

O guia recomenda que se utilize a *meta tag viewport* com os seguintes valores, apresentados na Figura 33.

Figura 331 - *Meta Tag Viewport* com os Valores Recomendados

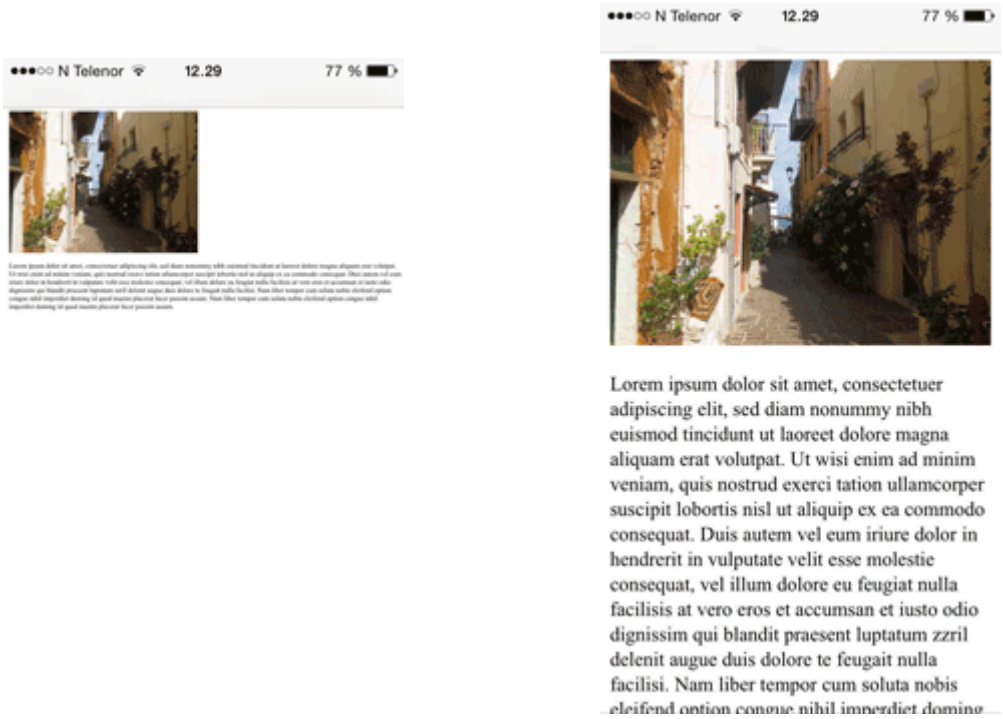
```html
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

Fonte: Elaborado pelo Autor (2020).

Por fim, a Figura 34 mostra um exemplo de uma página da *web* sem a *meta tag viewport* (a) e a mesma página utilizando a *meta tag viewport* (b).

Brazilian Journal of Development

Figura 34 - Exemplo de Uso da *Meta Tag Viewport*




a) Página sem a *Meta Tag Viewport* b) Página com a *Meta Tag Viewport*

Fonte: W3SCHOOLS (2020).

## 4 APLICAÇÃO DO GUIA

Para a avaliação do guia proposto, suas recomendações foram adotadas durante o desenvolvimento de um sistema *web* para a Incubadora de Sistemas de Informação da UFPA, Campus Cametá. Essa Incubadora é uma entidade que busca promover e estimular empreendimentos inovadores no Campus, estejam eles iniciando ou já em operação. Nesse sentido, oferece suporte material e intelectual aos empreendedores, ajudando a aprimorar ideias de negócio para transformá-las em micro ou pequenas empresas.

As principais funcionalidades solicitadas pelo coordenador da Incubadora de Sistemas de Informação para o desenvolvimento desse sistema envolveram: cadastro de notícias, cursos e empresas, bem como a edição, exclusão e exibição dessas informações.

Além dos requisitos funcionais, a aplicação deveria atender alguns requisitos não-funcionais, sendo estes:

- Disponibilizar uma interface *web* acessível através de um navegador;

Brazilian Journal of Development

- Possuir um *design* responsivo para permitir que a interface do sistema se ajuste ao dispositivo de acesso (*notebook*, *smarthphones*, *tablet*, etc.);
- Apresentar uma interface amigável para que o usuário se sinta confortável ao utilizar o sistema;
- Possuir facilidade de uso, onde o administrador realize suas tarefas (inclusão, alteração, consulta e exclusão) com menos de 30 minutos de treinamento.

Para atender a esses requisitos, adotou-se as recomendações dispostas no guia apresentado na Seção 3. Esse guia aborda os principais conceitos e especificações de Usabilidade e Responsividade para o desenvolvimento de aplicações *web*, sendo assim adequado às demandas solicitadas pelo Coordenador da Incubadora. A seguir é mostrado como as recomendações do guia foram aplicadas no desenvolvimento da aplicação.

## 4.1 CARACTERÍSTICAS DE USABILIDADE

### *4.1.1 Inteligibilidade*

As orientações do guia para aplicação de Inteligibilidade sugerem o uso de ícones e cores nos botões e nomes intuitivos que permitam um bom entendimento para o usuário. Na aplicação, essa característica está presente principalmente na área do Administrador, onde pode-se visualizar notícias, cursos e empresas adicionadas ao sistema. As informações cadastradas possuem as opções de exclusão e edição através de botões estilizados, como mostra a Figura 35.

Figura 35- Uso de ícones e cores nos botões de exclusão e edição

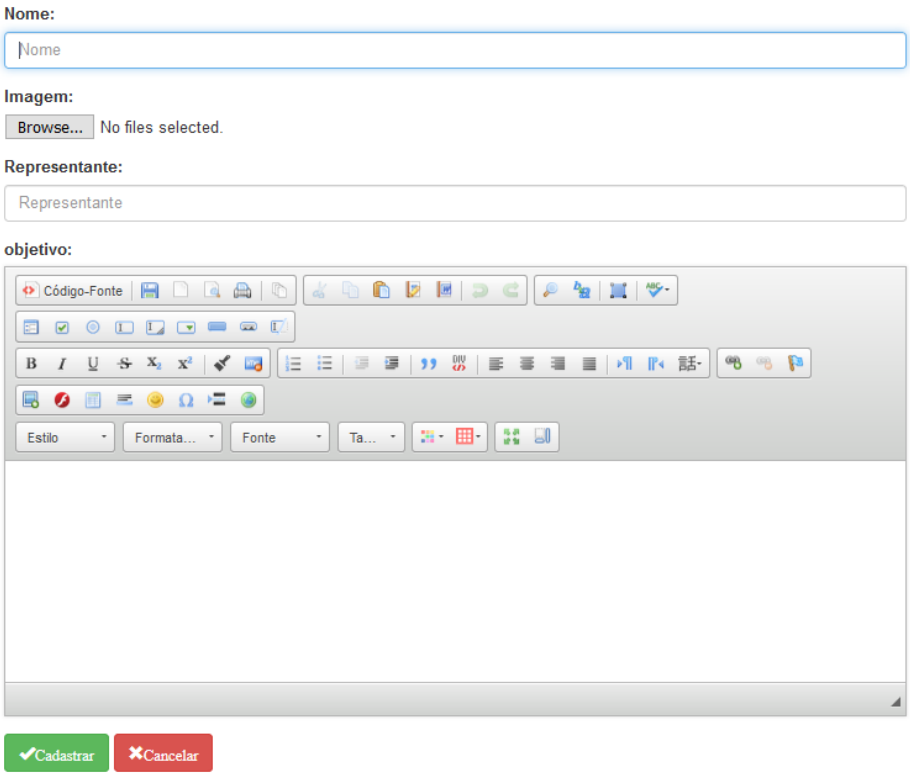

Fonte: INCUBADORA (2020).

Brazilian Journal of Development

### *4.1.2 Apreensibilidade*

Entre as recomendações do guia relacionadas à Apreensibilidade encontra-se a utilização de formulários como analogia ao cadastro em ficha de papel. Na aplicação, essa orientação está presente na área do Administrador nos cadastros de notícias, cursos e empresas. A Figura 36 apresenta o formulário de cadastro de empresas.

Figura 36 - Formulário para cadastro de empresas




Fonte: INCUBADORA (2020).

### *4.1.3 Operacionalidade*

A Operacionalidade envolve aspectos de controlabilidade, adequação, modificabilidade, adaptabilidade, capacidade para ser instalado, tolerância a erros e conformidade com as expectativas do usuário.

Dentre as orientações do guia está a utilização de máscara de entrada. Tal característica foi adicionada nos campos de datas dos formulários de cadastro de notícias e cursos, como exemplifica a Figura 37 na área do Administrador.

Brazilian Journal of Development

Figura 37 - Máscara de entrada nos campos de data

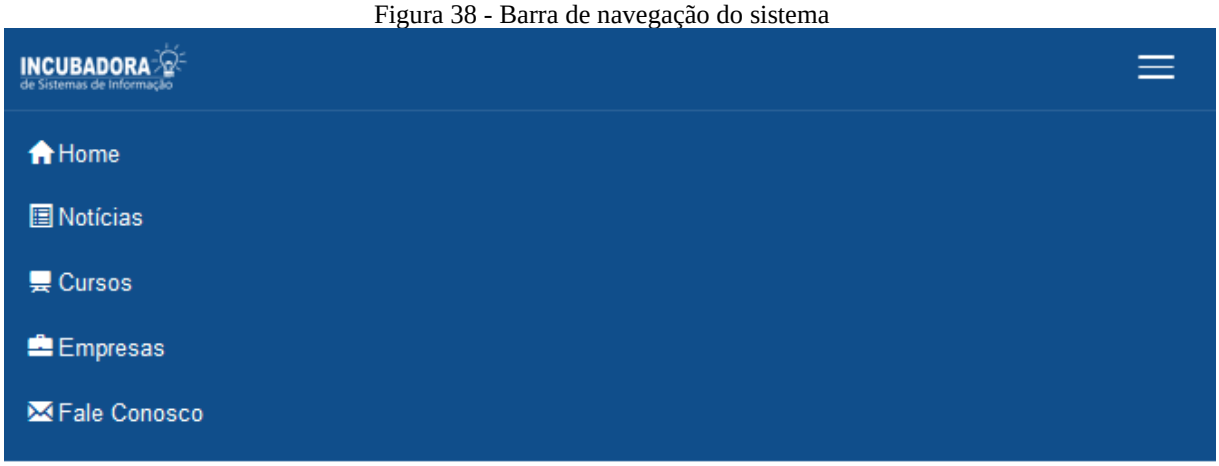

Fonte: INCUBADORA (2020).

### *4.1.4 Atratividade*

A Atratividade se refere às características que possam atrair um potencial usuário para o sistema. Neste sentido, os atributos do *software* devem possuir a intenção de torná-lo mais agradável para o usuário.

O uso harmonizado de cores está entre suas orientações. Na aplicação *web*, adotou-se as cores em tons de azul e branco para o *layout* do sistema como ilustra a Figura 38. Nota-se essa harmonia de cores na barra de navegação do sistema, onde os menus estão em tom de branco e o fundo na cor azul.

Figura 38 - Barra de navegação do sistema




Fonte: INCUBADORA (2020).

## 4.2 ATRIBUTOS DE RESPONSIVIDADE

### *4.2.1 Layouts Fluídos*

*Layouts* fluídos são tecnologias utilizadas no guia para a implementação de um *design* responsivo. Com essa técnica, a estrutura de uma página pode expandir ou diminuir de acordo com o dispositivo no qual está sendo visualizada, fazendo com que a mesma não seja desconfigurada ou perca qualidade. No desenvolvimento do sistema, adotou-se essa tecnologia para a criação do *layout* das páginas. Dessa forma, a aplicação mantém a mesma estrutura em dispositivos com diferentes configurações de tela.

Brazilian Journal of Development

A Figura 39 mostra a aplicação sendo acessada de um iPad Pro (a) com configurações de 1024px de largura e 1366px de altura e de um iPad (b) que possui as configurações de 768px de largura e 1024px de altura.

Figura 39 - Aplicação de *Layout* Fluído


A) Acesso ao sistema através de um iPad Pro B) Acesso ao sistema através de um iPad

Fonte: INCUBADORA (2020).

Observa-se que a estrutura das páginas permanece inalterada, adequando-se ao dispositivo de acesso. Dessa forma, a aplicação atende ao requisito de responsividade solicitado para o seu desenvolvimento.

### *4.2.2 Media Queries*

De acordo com o guia, a principal funcionalidade das *Media Queries* é dispor uma folha de estilo CSS específica para uma determinada mídia por meio da consulta e identificação das características para a qual a aplicação está sendo disponibilizada. Dessa forma, são utilizadas para readaptar *layouts* a uma nova janela de dispositivo.

No sistema foi adicionada essa propriedade para readaptar o *layout* quando a aplicação for acessada de um dispositivo com a largura máxima de 767px. Com isso, a área onde ficam os laboratórios parceiros da Incubadora fica disposta abaixo do conteúdo principal quando o sistema é visualizado através de um dispositivo com esse tamanho máximo de janela, como mostra Figura 40.

Brazilian Journal of Development

Figura 40 - Aplicação de *media queries*


a) Laboratórios parceiros ao lado do conteúdo b) Laboratórios parceiros abaixo do conteúdo

Fonte: INCUBADORA (2020).

Na Figura 40, vê-se o sistema acessado de um dispositivo (a) com largura mínima superior a 767px, como em um *notebook*, e a mesma aplicação visualizada através de um outro dispositivo (b) com largura máxima de 767px, como por exemplo um *smartphone*.

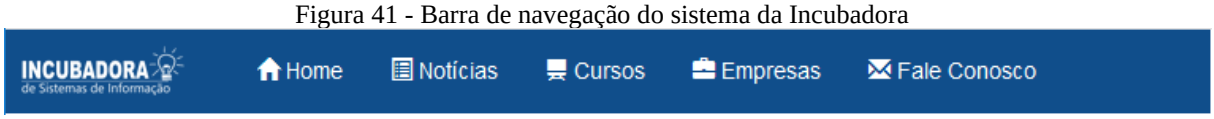
### *4.2.3 Imagens Adaptáveis*

O guia contém diversas recomendações para a utilização de imagens adaptáveis. Dentre elas, citam-se: a utilização de ferramentas para a vetorização de imagens, a adoção de fontes-ícones, aplicação de largura máxima (100%) para as imagens através de um estilo CSS e, por fim, a utilização de *media queries* para fornecer uma imagem ideal para cada contexto, ou seja, versões diferentes do mesmo arquivo de acordo com as características do dispositivo.

Na aplicação, utilizou-se imagens adaptáveis para representar de forma gráfica as opções presentes na barra de navegação. Essas formas gráficas recebem o nome de ícones e são utilizados para tornar o menu intuitivo, como mostra a Figura 41.

Brazilian Journal of Development

Figura 41 - Barra de navegação do sistema da Incubadora




Fonte: INCUBADORA (2020).

### *4.2.4 Meta Tag Viewport*

A *meta tag viewport* informa ao navegador o tamanho da janela disponível para exibição de uma página *web*, possibilitando o controle de *zoom* do dispositivo.

Para o desenvolvimento do sistema da Incubadora utilizou-se a *meta tag viewport* como disposta no guia. Dessa forma, a aplicação se ajusta conforme a janela do dispositivo do qual for acessada. A Figura 42 apresenta a aplicação sendo visualizada em um *notebook* (a) e em um *smartphone* (b). A *meta tag viewport* ajuda a manter uma boa visibilidade e legibilidade das informações da página.

Figura 22 - Aplicação da *Meta Tag Viewport*


a) Página em *notebook* b) Página em *smartphone*

Fonte: INCUBADORA (2020).

## 5 CONSIDERAÇÕES FINAIS

Este artigo apresenta uma análise sobre o desenvolvimento para *web*, abordando principalmente conceitos e características de responsividade e usabilidade, afim de contribuir para o desenvolvimento de aplicações *web* que possam ser acessadas dos mais diversos dispositivos existentes no mercado, sendo ainda de fácil interação e agradável ao usuário.

Para isso, adotou-se a norma da ISO/IEC 9126 (2003) onde estão presentes as características de Usabilidade de sistemas, sendo essas Inteligibilidade, Apreensibilidade,

Operacionalidade e Atratividade. E também, buscou-se na literatura as principais tecnologias para a criação de um *Design* Responsivo, sendo as identificadas e adotadas por profissionais da área: Fundação Flexível, *Media Queries*, Imagens Adaptáveis e *Meta tag Viewport*.

Adicionalmente, criou-se um guia de recomendações, onde encontram-se padrões de usabilidade e responsividade, implementadas através do *framework Bootstrap* para serem adotadas no processo de criação desse tipo de aplicação.

O guia foi disponibilizado em um documento de texto e na forma de uma página *web*, disponível *online* em *http://lafoca.com.br/guiaweb/*. Dessa forma, espera-se que o guia se torne mais acessível.

Em seguida, adotaram-se as recomendações presentes no guia no processo de desenvolvimento de um sistema *web* para a Incubadora de Sistemas de Informação da UFPA - Cametá. Atender a necessidade do usuário de forma fácil e sendo agradável, além, de poder ser acessada de qualquer dispositivo, foram alguns dos requisitos solicitados para serem contemplados pela aplicação. Assim, o sistema é composto de um site onde são disponibilizados os trabalhos da Incubadora, estando disponível *online* em *http://lafoca.com.br/incubadora/*, e por um gerenciador de conteúdo para alimentar o site. Nesse sentido, o guia contribuiu bastante no processo de desenvolvimento da aplicação.

## REFERÊNCIAS

ALURA. **O que são imagens vetoriais e por que utilizá-las?** 2018. Disponível em: <http://blog.alura.com.br/o-que-sao-imagens-vetoriais-e-por-que-utiliza-las>. Acesso em 04 de Abril de 2020.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. **The semantic web**. Scientific american, v. 284, n. 5, p. 34-43, 2001.

BOOTSTRAP. **Bootstrap**. Disponível em: <http://getbootstrap.com/>. Acesso em 04 de Abril de 2020.

EIS, Diego. **Manipulando a Metatag Viewport**. 2011. Disponível em: <https://tableless.com.br/manipulando-metatag-viewport>. Acesso em 04 de Abril de 2020.

FARROCO, Leonardo de Oliveira; STEFANO, Ercilia de; EBECKEN, Nelson Francisco Fávilla; PAULO, Fernanda Santana de; NOSOLINE, Sumaya Mário. **O paradigma funcional no desenvolvimento Front-End: oportunidades e desafios**. Brazilian Journal of Development, v. 6, n. 2,p.6464-6475, fev. 2020.

FROONT. **9 Basic Principles of Responsive Web Design**. 2014. Disponível em: <http://blog.froont.com/9-basic-principles-of-responsive-web-design>. Acesso em 09 de Março de 2020.

INCUBADORA. **Incubadora de Sistemas de Informação**. 2020. Disponível em <http://lafoca.com.br/incubadora/>. Acesso em 11 de Abril de 2020.

ISO/IEC 9126: **Engenharia de software - Qualidade de produto**. 2003. Disponível em: <https://www.iso.org/standard/22891.html>. Acesso em 23 de Março de 2020.

SILVA, Arthur de Almeida Pereira da. Design Responsivo: Técnicas, Frameworks e Ferramentas. Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2014.

SILVA, Maurício Samy. **Web Design Responsivo**. São Paulo: Novatec, 2014.

VALENTIM, Natasha; OLIVEIRA, Káthia; CONTE, Tayana. **Definindo uma Abordagem para Inspeção de Usabilidade em Modelos de Projeto por meio de Experimentação**. Anais do Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais (IHC), Cuiabá, pp. 165-174, 2012.

W3SCHOOLS. **Responsive Web Design - The Viewport**. Disponível em: <https://www.w3schools.com/Css/css_rwd_viewport.asp>. Acesso em 09 de Março de 2020.