Análise de Aplicações Web Responsivas para Múltiplos Dispositivos

Pedro Henrique Rocha Silva, Wander Antunes Gaspar Valente

Curso Bacharelado em Sistemas de Informação – Centro de Ensino Superior de Juiz de Fora (CES/JF) Caixa Postal 36.016-000 – Juiz de Fora – MG – Brasil

pedroweb18@gmail.com

Resumo. *Hoje em dia, graças ao avanço da tecnologia e a evolução da internet, é muito comum deparar-se com diversos dispositivos capazes de acessarem uma página na web. Porém, há uma quantidade de aplicações na web que, ao serem acessadas por smartphones, tablets e tvs, são de difícil visualização, navegabilidade ruim ou até mesmo ilegíveis para leitura. O design responsivo apresenta várias soluções para diminuir esse problema, fazendo com que seja desenvolvida uma única aplicação, sendo ela adaptável para diversos dispositivos, tornando a navegação para o usuário final cada vez melhor e mais agradável. O principal foco do projeto é analisar quais são as principais vantagens e desvantagens do design responsivo, qual a metodologia para desenvolver aplicações responsivas e uma comparação sobre os principais frameworks do mercado.*

Abstract. *Due to the advances in technology and the internet evolution, it is widely common nowadays to find different devices to access a website However, there is an amount of web applications that, when used in smartphones, tablets and tvs, are of difficult viewing, bad navigability or even unreadable. Responsive design has many options to solve this problem, through the development of a single application, which is adjustable to many devices, making the browsing better and more pleasant to the user. The project's main purpose is analyzing the advantages and disadvantages of responsive design, which methodology to use in the development of responsive applications and comparing the major frameworks in the market.*

1. Introdução

Em meados dos anos 90, os meios de acesso à internet eram muito limitados. Os acessos eram feitos pelo *desktop* ou *notebooks,* que possuíam resoluções baixas e muito parecidas.

Hoje em dia, torna-se cada vez mais comum acessar uma página por meio de um dispositivo móvel, um *tablet* ou uma *smart* tv. Com a ascensão da Internet, sobretudo em dispositivos móveis se deparam com diversos problemas de cunho técnico, tais como a dificuldade na navegação e o redimensionamento do website para resoluções menores, o que acaba gerando desconforto aos utilizadores e a eventual necessidade de desenvolvimento de aplicações específicas para os diversos tipos de sistema operacional dos dispositivos, o que dificulta a manutenção visto a necessidade de se haver uma aplicação para cada dispositivo [Souza, 2012].

Desenvolver mais de uma aplicação para os diversos dispositivos pode ser uma tarefa inviável, devido ao grande trabalho que daria aos desenvolvedores, para manter os websites funcionando corretamente.

Ethan Marcotte no dia 25 de maio de 2010, explicou pela primeira vez em uma matéria publicada no site AlistApart o termo *Web Design Responsivo*. O *Responsive Web Design* ou *layout* responsivo expande e contrai com a finalidade de se acomodar de maneira usável e acessível à área onde é visualizado ou, mais genericamente, ao contexto onde é renderizado, seja um smartphone, um *tablet*, um *desktop*, um leitor de tela, um mecanismo de busca etc. [Silva, 2014].

Com ele é possível adaptar o layout das páginas a qualquer tipo de dispositivo, com objetivo de garantir uma boa experiência ao usuário, possibilitando navegação e leitura confortável sem comprometer o conteúdo, independentemente do tipo de tela ou de resolução.

1.1 Metodologia

A escolha é motivada devido a grande mudança na forma de desenvolver aplicações web que o *Design Responsivo* vem trazendo nesses últimos anos. Ele trouxe uma facilidade e praticidade tanto para os desenvolvedores que não precisam mais construir várias aplicações, quanto para o usuário final, que agora pode utilizar seu *smartphone* ou *tablet* para navegar na internet tranquilamente.

Para o desenvolvimento deste artigo foi realizado um estudo sobre como surgiu o *web design responsivo*, quais são as técnicas utilizadas para que uma página *web* seja adaptável a qualquer tipo de dispositivo e um estudo sobre alguns *frameworks* do mercado atual.

2 Técnicas do Design Responsivo

O *web design responsivo* apresenta algumas técnicas para unificar o desenvolvimento e gerenciamento de conteúdo de um *website*.

2.1 Mobile First

O termo *Mobile First* surgiu pela primeira vez a partir de uma matéria publicada por Luke Wrosblewski [Silva, 2014]. Nesse artigo ele justificou a importância de se iniciar um projeto com o layout desenvolvido para dispositivos móveis.

Uma abordagem importante para se ter um bom *web design responsivo* é *Mobile First*! *Mobile First* (ou Móvel Primeiro), criada por Luke Wroblewski, é uma metodologia de desenvolvimento web que preconiza o dever de primeiro planejar para dispositivos móveis e, só depois, aumentar os possíveis dispositivos até se chegar ao *desktop* (e além) [Zemel, 2012].

Figura 1. Posição dos elementos do mobile ao desktop

A estratégia envolvida no *Mobile First* faz com que o desenvolvedor priorize o conteúdo do *website* e foque no que é mais importante naquele momento e, de acordo com a capacidade do aparelho, vão sendo inseridos novos elementos que preenchem a tela do dispositivo de acordo com a sua resolução.

2.2 Layout Fluido

O *layout* fluido é um dos principais responsáveis por levar qualquer conteúdo para os diversos tipos de aparelhos existentes no mercado atual.

Para que o *layout* adapte ao tamanho da tela do dispositivo, é necessário usar valores relativos, como o ems e porcentagem.

Os quatro principais tipos de medidas em css são: pixel, ponto, ems e porcentagem.

Pixel(px): entre todas as unidades de medidas do CSS, o pixel é o mais utilizado. Pode ser utilizado para definir a largura ou altura de um elemento, o tamanho de um texto e entre outras coisas.

Ponto(pt): ponto é a unidade utilizada pelo CSS como impressão. Assim como o pixel, a sua unidade de medida é fixa e cada ponto é igual a 1/72 polegadas.

Ems(em): é mais conhecida como uma unidade de medida tipográfica. Na unidade de medida ems o tamanho de uma fonte de 12pt é o equivalente a 1em.

Porcentagem(%): apesar de ser parecida com a unidade ems, nessa unidade o tamanho da fonte equivale a 100%, ou seja, uma fonte de 12pt é igual a 100%.

Há uma grande diferença entre os quatro tipos de medidas apresentados. As duas primeiras unidades são definidas como unidades de medidas fixas. As duas últimas unidades de medidas são definidas como escaláveis, que tem o poder de se adaptarem a qualquer resolução de tela.

Um dos segredos para a criação de um *website responsivo*, está na utilização das unidades de medidas ems e porcentagem.

2.2.1. Transformando pixel em porcentagem

Antes da publicação que criou o design responsivo, Ethan Marcotte publicou uma matéria em seu site com o nome de "Grid fluído", onde mostrou uma fórmula que transformava Pixel em Ems ou Porcentagem. Mais tarde essa fórmula seria aplicada no *design responsivo*.

Para realizar a conversão de um layout fixo para fluido é utilizada a seguinte fórmula:

Alvo / Contexto = Resultado

- Alvo: elemento-alvo com a medida atual;
- Contexto: onde o elemento-alvo está;
- Resultado: o valor relativo que se está sendo procurado.

Com a fórmula desenvolvida por Ethan Marcotte é possível realizar a conversão das medidas absolutas para medidas escaláveis. A conversão pode ser realizada para o tamanho das fontes, o tamanho dos layouts e entre outros elementos.

Com essa fórmula rápida, é possível realizar cálculos simples para saber o resultado de conversões de medidas absolutas do CSS para medidas relativas. E isso vale tanto para o cálculo de tamanho de fontes (existe uma certa convenção de que o tamanho padrão de fontes em navegadores para desktop é de 16px), quanto para medidas de *layout* [Zemel, 2014].

Aplicando à formula: 24 / 16 = 1.5

Ou seja, ao pegar o alvo (o título) de 24px e dividir pelo contexto no caso, seu elemento-pai, que é *body*, e possui tamanho da fonte de 100%, ou seja, 16px, tem-se como resultado 1,5. Como o assunto é tamanhos relativos de fontes em CSS, o resultado final é 1.5em. [Zemel, 2014].

2.3 MetaTag Viewport

A meta tag viewport foi desenvolvida pela Apple a fim de resolver alguns problemas de resolução que aconteciam em seus smartphones. A ideia deu tão certo, que outros navegadores também incorporaram a *meta tag*.

Quando o navegador de um *smartphone* ou *tablet* acessa um site feito para *desktop*, ele é obrigado a reduzir a página, para que ela caiba na tela do dispositivo. Considerando o exemplo de um site com 960 pixels de largura sendo renderizado em um *smartphone* cuja largura é de 720 pixels. Logo, o site terá sua escala diminuída para que ele possa ser renderizado em um dispositivo que possui uma resolução inferior. Essa diminuição de escala torna a navegabilidade ruim, pois o usuário é forçado a rolar a página na horizontal e dar zoom para ler algum tipo de conteúdo. Para poder solucionar esse problema, é necessário colocar a meta *tag viewport*, onde ela informará ao navegador o tamanho da tela em que a página será exibida.

O *Viewport* é, por definição, o tamanho disponível para exibição do site no navegador, ou seja, da área útil da tela menos a barra de rolagem, a barra de ferramentas etc.

Meta tags servem para descrever informações sobre a página e assim, a *meta tag viewport* informa ao navegador o tamanho de tela disponível para exibir o site, além de possibilitar o controle do zoom do dispositivo. [Silva, 2014].

A meta *tag content* é representada da seguinte forma, conforme mostra a figura 2:

```
<meta name="viewport" content="">
```

Figura 2. Utilização da tag content

Dentro de *content*, é possível inserir vários parâmetros e valores como:

- width: responsável por definir a largura da *viewport*;
- height: responsável por definir a altura da *viewport*;
- initial-scale: responsável por definir a escala inicial (zoom) da *viewport*, de 0 a 10.

- user-scalable: responsável por definir se ativa ou não o zoom, recebe os valores *yes* ou *no*;

A figura 3 mostra como a Apple utilizou a *meta tag viewport* em uma imagem com 320px x 356px renderizada em um iphone.

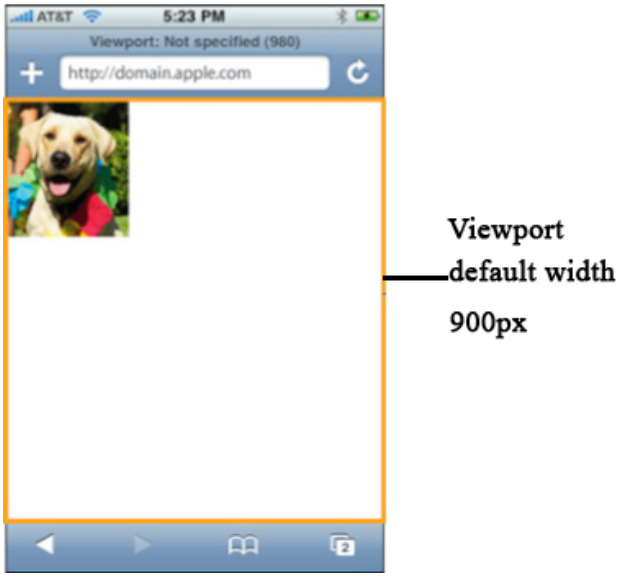


Figura 3. Viewport default

A figura 4 mostra a *viewport* com zoom de 1.0 e uma largura de 320px

```
<meta name="viewport" content="width=320,initial-scale=1">
```

Figura 4. Meta tag viewport com a utilização dos respectivos valores

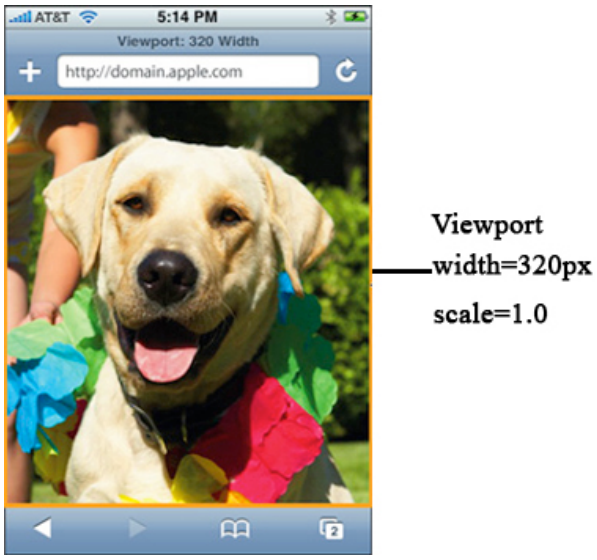


Figura 5. Viewport com zoom de 1.0 e width de 320px

2.4 Media Query

Os dispositivos móveis de hoje em dia geram um grande problema para os desenvolvedores pois, alguns modelos possuem resoluções de telas muito altas que podem se comparar a até mesmo a dos *desktops*. Isso faz com que alguns dispositivos não se encaixem na categoria de *handheld*. E nem na categoria de *desktop*.

Uma das alternativas passa a ser a *media query*.

Media query, em tradução livre, significa "consulta à mídia". O objetivo dessa funcionalidade é servir uma folha de estilo específica para uma determinada mídia mediante consulta e identificação das características da mídia para a qual aplicação está sendo servida [Silva, 2014].

```
<link rel="stylesheet" href="estilo.css" media="screen and (max-width: 480px)">
```

Figura 6. Aplicando media query na tag style

O estilo que se encontra no arquivo estilo.css, será aplicado em dispositivos que se enquadram na condição de descrita em *media* (ou seja, que tem uma tela com alta capacidade de cores) e com uma largura máxima de 480px.

2.4.1 Media Types

O HTML foi criado para ser suportado em qualquer tipo de dispositivo, porém, devido à grande variação das dimensões de tela, tecnologias e outros fatores envolvidos, os dispositivos passam a exibir o HTML de uma maneira diferente de um modelo para o outro.

Caso um *website* fosse acessado por um *desktop* e o mesmo *website* fosse acessado por um dispositivo móvel, eles teriam comportamentos diferentes devido as diferenças dos dispositivos e a forma de navegação diferente de cada dispositivo.

Um dos fatores para manter as páginas web sempre com o mesmo padrão de visualização, é a utilização dos media types.

Media types é uma recomendação da W3C desde o CSS2. Com elas, é possível apresentar o site de maneira diferente, dependendo da mídia (*media*). É possível uma apresentação diferenciada (por meio de folhas de estilo) quando a página está sendo vista de um projetor, que pode ser diferente de quando se usa uma impressora, um sintetizador de voz, uma TV, dentre outros [Zemel, 2014].

Os tipos de media *types* são:

- *all*: a folha de estilo serve para todos os dispositivos;
- *braille*: para dar feedback quando se usa dispositivo tátil;
- *embossed*: impressoras em braile paginadas;
- *handheld*: dispositivos móveis (comumente com tela pequena e largura de banda limitada);
- *print*: para material paginado e para documentos visualizados na tela no modo Visualização de impressão;

- *projection*: destinado a apresentações projetadas como, por exemplo, projetores;
- *screen*: destinado, principalmente, para telas coloridas de computador;
- *speech*: para sintetizadores de voz;
- *tty*: dispositivos de grade fixa para exibição de caracteres, como *teletypes*, terminais e alguns outros;
- tv: aparelho tipo TV (baixa resolução, cores, *scroll* limitado e som)

A ideia central do *media type* é fazer com que uma página fosse carregada de acordo com o estilo do dispositivo mais apropriado, ou seja, se o usuário estiver utilizando um dispositivo *handheld*, o estilo responsável por esse tipo de *media type* entra em ação.

A implementação de uma *media type* pode ser realizada da seguinte maneira.

```
<link rel="stylesheet" type="text/css" media="handheld" href="style.css">
```

Figura 7. Inserindo a media type handheld a tag style

Nesse caso a página será renderizada utilizando o arquivo style.css somente se o dispositivo for um *handheld*.

2.5 Imagens Flexíveis

Encontrar imagens e vídeos em uma página web, não é uma tarefa muito difícil. Em projetos responsivos, eles devem se redimensionar de acordo com a resolução da tela do dispositivo que está acessando a página web.

Para tornar uma imagem flexível, é necessário acrescentar apenas uma única linha de código de CSS. Essa linha de código informa que a imagem deve ter uma largura máxima de 100%. Isso faz com que a imagem seja redimensionada automaticamente, de acordo com a resolução do dispositivo, como exemplificado na figura 8.

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

Figura 8. Deixando uma imagem responsiva

A solução apresentada na figura 8 para o redimensionamento das imagens é de fácil utilização, porém, essa solução não possui uma boa performance, devido ao fato dela carregar a imagem em alta resolução para depois diminuir a resolução da imagem via código, deixando a página mais pesada.

Hoje em dia, existem várias técnicas que possibilitam soluções existentes para os problemas de imagens responsivas em diferentes resoluções e diferentes formatos.

Vale a pena ressaltar que as técnicas estudadas para inserção de imagens não são perfeitas. É necessário estudá-las e verificar qual é a melhor para cada tipo de projeto.

Em 28 de fevereiro de 2013, o W3C publicou o primeiro Rascunho de Trabalho para a especificação denominada "O atributo srcset - Uma extensão HTML para imagens adaptativas" [Silva, 2014].

O atributo srcset tem o intuito de adaptar uma mesma imagem a diferentes versões. O atributo é declarado dentro do elemento img.

Para informar o valor do atributo srcset, é necessário criar uma lista de *image candidate strings*, passando o nome da imagem e seus respectivos atributos, conforme a figura 9 mostra.

```
<img src="imagem.jpg"
srcset="logo-HD.jpeg 2x, logo-phone.jpeg 100w, logo.jpeg">
```

Figura 9. Utilizando o atributo srcset

3.0 Frameworks de Design Responsivo

Montar um *website* responsivo com todas as suas necessidades de uma página web, pode ser muito custoso e trabalhoso. É por isso que muitas empresas e até mesmo alguns desenvolvedores preferem utilizar os *Frameworks* de *design responsivo*. Eles têm a missão de levar mais praticidade aos desenvolvedores, devido a várias estruturas que já estão prontas para serem utilizadas.

Dependendo do nível de afinidade do desenvolvedor com o HTML e CSS ou até mesmo com outros *frameworks de design responsivo*, a curva de aprendizagem pode se tornar mais fácil.

3.1 Bootstrap

O Bootstrap é um *Framework* de código aberto que foi criado em meados de 2010 por Mark Otto e Jacob Thornton. A princípio ele foi criado para ser uma solução interna do Twitter, onde iria resolver alguns problemas de inconsistências de código dentro da equipe de desenvolvimento, problema esse que afetava toda a equipe, pois não havia nenhum tipo de padrão para os desenvolvedores seguirem, o que gerava uma grande dificuldade na junção dos códigos [Bootstrap, 2015].

A finalidade original do Bootstrap era incentivar o uso de uma única estrutura de código, nomenclatura de classes, etc, pelas equipes de engenharia da empresa. A iniciativa foi bem-sucedida, resultando em menos inconsistências e consequentemente maior rapidez nos projetos. [Arthur, 2014].

A primeira versão pública do Bootstrap foi lançada no GitHub no dia 19 de agosto de 2011. A versão inicial continuou a ser mantida por Mark Otto e Jacob Thornton, juntamente com alguns desenvolvedores.

Com menos de um ano, foi lançada a segunda versão do Bootstrap, na qual foram adicionadas algumas melhorias como: inserção de grids, alteração de componentes existentes entre outras funcionalidades.

A terceira versão trouxe algumas novidades como o *mobile first*, o conceito de *flat design*, os *plug-ins* de *Javascript* vieram com uma nova roupagem e os ícones agora utilizam o conceito de Glyphicons, ou seja, fontes em formato de ícones.

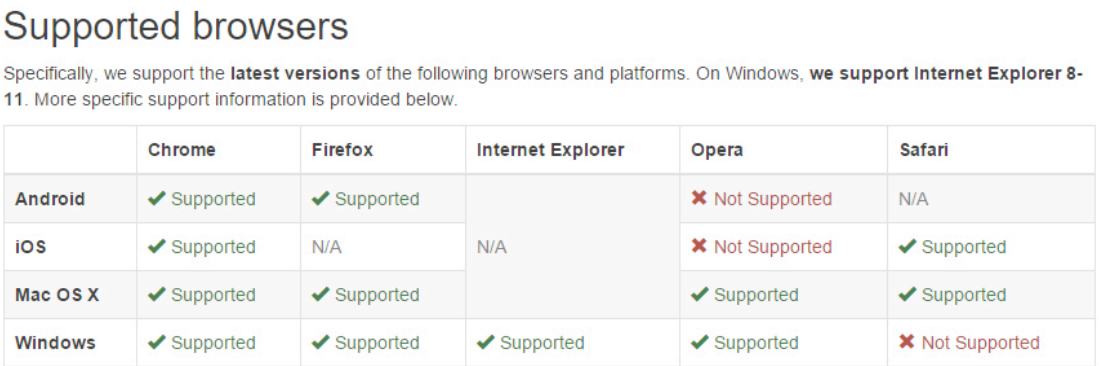
Em outubro foi lançada a versão Alpha do Bootstrap 4, contendo algumas novidades como: mais rapidez na compilação, melhoramento no sistema de *grid*, *flexbox* e alguns componentes novos entre outras novidades.

O *Framework* Bootstrap em sua versão 3.3.5 conta com três pastas em sua forma básica. São elas: CSS, JavaScript e Fonts como mostra a imagem 10.

Figura 10. Estrutura das pastas do Bootstrap

Dentro das pastas é possível encontrar os arquivos de CSS e JS compilados para uso nos projetos *web*. As imagens são comprimidas usando o ImageOptim, um aplicativo para comprimir PNGs.

O Bootstrap foi construído para rodar na versão mais recente dos navegadores mobile e *desktop*. As versões mais antigas podem renderizar alguns elementos de alguma forma diferente do habitual. Na imagem 12 é a compatibilidade do *framework* com os navegadores *web* e alguns sistemas operacionais *desktop* e *mobile*.



Supported browsers

Specifically, we support the **latest versions** of the following browsers and platforms. On Windows, **we support Internet Explorer 8-11**. More specific support information is provided below.

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	✔ Supported	✔ Supported	N/A	✖ Not Supported	N/A
iOS	✔ Supported	N/A		✖ Not Supported	✔ Supported
Mac OS X	✔ Supported	✔ Supported		✔ Supported	✔ Supported
Windows	✔ Supported	✔ Supported	✔ Supported	✔ Supported	✖ Not Supported

Figura 12. Navegadores e sistemas operacionais suportados pelo Bootstrap

3.2 Foundation

O Foundation nasceu na empresa Zurb, onde foi usado como um guia de estilo pela equipe de desenvolvedores, para aperfeiçoar a criação de websites e torná-la mais rápida. Quando foi realizada a junção do CSS, juntamente com alguns *plug-ins jquery* e o guia de estilo, deu origem ao Foundation, que foi lançado ao público no ano de 2011 [Foundation, 2015].

Para a realização do download o Foundation apresenta 4 opções: a primeira, vem com todos os arquivos, a segunda opção oferece uma versão mais básica, a terceira uma versão customizada e a quarta é uma versão que oferece suporte a versão com SASS.

Dentro das pastas é possível encontrar arquivos contendo CSS, *javascript* até imagens.

3.3 Uikit

O Uikit foi criado pela equipe do YOOtheme. Ele surgiu com o intuito de ser um *framework* responsivo modular front-end de simples utilização, fácil customização e extensível [Uikit, 2014].

Em seu interior o Uikit utiliza CSS com LESS, para aumentar a capacidade de trabalho dos componentes e as ferramentas já conhecidas como Query, FontAwesome e Normalise.

A utilização do Uikit é totalmente gratuita para download, dando autonomia ao desenvolvedor para usar ou copiar sem qualquer tipo de limitação.

Depois de feito o download do Uikit, dentro do arquivo ZIP é possível verificar sua estrutura de pastas, conforme mostra a figura 14.

```
/ css
     <! - UIKit com o estilo básico ->
    uikit.css
    uikit.min.css

    <! - UIKit com estilo Gradiente ->
    uikit.gradient.css
    uikit.gradient.min.css

    <! - UIKit com estilo quase plana ->
    uikit.almost-flat.css
    uikit.almost-flat.min.css

    <! - Componentes avançadas ->
    / componentes

/ fonts
    <! - WebFont FontAwesome ->
    fontawesome-webfont.ttf
    fontawesome-webfont.woff
    fontawesome-webfont.woff2
    FontAwesome.otf

/ js
    <! - JavaScript e versão minified ->
    uikit.js
    uikit.min.js

    <! - Componentes avançadas ->
    / componentes

    <! - Os componentes principais ->
    / core
```

Figura 14. Estruturas de pastas Uikit

Dentro da pasta é possível encontrar os arquivos de CSS, Javascript e fontes, todos prontos para serem inseridos no projeto. A estrutura do Uikit tem poucas folhas de estilo, mantendo o projeto sempre leve.

Em sua biblioteca de módulos o Uikit possui mais de 30 modelos extensíveis, que podem ser combinados entre si, dando uma maior facilidade ao desenvolvedor.

Confira os principais modelos disponíveis no Uikit.

- Grid responsivo

- Botão

- Paginação
- Formulário
- Tabela
- Ícones
- Alertas
- Animações
- Rolagem suave (*Smooth scroll*)
- Opções de navegação (*off-canvas*)
- Overlay (*lightbox*)

4. Comparação entre Frameworks

Na comparação dos *frameworks* são levadas em consideração alguns componentes importantes para o bom funcionamento de uma aplicação *web*. Nesse tópico é possível verificar as diferenças e semelhanças nos códigos dos *frameworks* estudados.

4.1 Instalação

Após realizar o *download* dos *frameworks* em seus respectivos *websites*, é necessário fazer a junção do *framework* ao projeto. A versão do *framework* utilizada durante o desenvolvimento desse projeto é a versão básica.

A estrutura básica que envolve os três *frameworks* analisados é muito parecida.

O Bootstrap utiliza um arquivo css e um javascript. Utiliza também uma biblioteca Jquery, que é acessada com um link que o próprio Google disponibiliza. Dentro do *head* é utilizada a *metatag viewport*, conforme mostra a figura 15.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

Figura 15. Código de instalação do Bootstrap ao projeto

O *Foundation* necessita de um arquivo css e um *javascript*. Ele utiliza também a biblioteca *modernize*.js, que já vem incorporada em sua estrutura de pastas, sendo ela a responsável por manter a compatibilidade com os navegadores mais antigos.

A figura 16 mostra o código de instalação do Foundation ao projeto.

```
<!DOCTYPE html>
<html class="no-js" lang="en">
  <head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width-device-width, initial-scale=1.0" />
    <title>Foundation</title>
    <link rel="stylesheet" type="text/css" href="css/foundation.css">
    <script type="text/javascript" src="js/vendor/modernizr.js"></script>
    <script type="text/javascript" src="js/foundation.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  </head>
  <body>

  </body>
</html>
```

Figura 16. Código de instalação do Foundation ao projeto

O processo de junção do *framework* ao projeto, não é diferente com o *Uikit*. Ele utiliza os mesmos três arquivos, css, *javascript* e a biblioteca *javascript*, conforme mostra a figura 17.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Uikit</title>
        <link rel="stylesheet" href="css/uikit.min.css" />
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
        <script src="js/uikit.min.js"></script>
    </head>
    <body>

    </body>
</html>
```

Figura 17. Código de instalação do Foundation ao projeto

O processo de instalação dos três frameworks ao projeto é bem parecido. Não havendo nenhum tipo de dificuldade durante o processo. A documentação dos *frameworks* também disponibiliza várias informações muito bem explicadas e com vários exemplos.

4.2 Componentes

4.2.1 Grid

No *design responsivo* as grids são as responsáveis por facilitar a inserção dos elementos nas páginas web. E por isso, os três *frameworks* estudados possuem um funcionamento em cima de um sistema de grid.

Entre eles o *Bootstrap* e o *Foundation* utilizam no máximo 12 colunas flexíveis enquanto o *Uikit* utiliza no máximo 10 colunas com classes pré-definidas dentro de seus arquivos css, que tem a capacidade de definirem a largura da coluna.

Para deixar uma grid fluida no *Bootstrap*, basta somente inserir a classe ".row-fluid". Para informar o tamanho da coluna desejada, basta inserir a classe "col-md-valor". No lugar de valor é necessário inserir um valor de 1 a 12, conforme mostra a figura 18.

```
<body>
  <h1>Grid</h1>
  <div class="container-fluid">
    <div class="row">
      <div class="col-xs-12 col-md-8 border">grid 1</div>
      <div class="col-xs-12 col-md-4 border">grid 2</div>
    </div>
  </div>
</body>
```

Figura 18. Criação de grid fluida no Bootstrap

A figura 19 mostra qual é o resultado da utilização do código mostrado na figura 18.

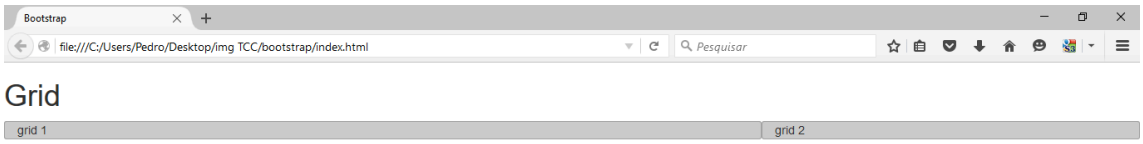


Figura 19. Exibição do sistema de grid Bootstrap

No *Foundation* para adicionar uma *grid* fluida basta adicionar a classe "row" e nas classes referentes as colunas basta inserir as seguintes classes: "large-valor" é a responsável por informar qual será o valor da coluna. A classe "*columns"* é somente para informar que se trata de uma coluna. As classes *small*, servem para indicar em quais tipos de tela a coluna vai se manter posicionada. A figura 20 mostra o código de criação de uma grid no *Foundation*.

```
<body>
<h1>Grid</h1>
  <div class="row">
    <div class="small-2 large-4 columns border">4</div>
    <div class="small-4 large-4 columns border">4</div>
    <div class="small-6 large-4 columns border">4</div>
  </div>
  <div class="row">
    <div class="large-3 columns border">3</div>
    <div class="large-6 columns border">6</div>
    <div class="large-3 columns border">3</div>
  </div>
</body>
```

Figura 20. Código de criação da grid no Foundation

A figura 21 mostra qual é o resultado da utilização do código mostrado na figura 20.

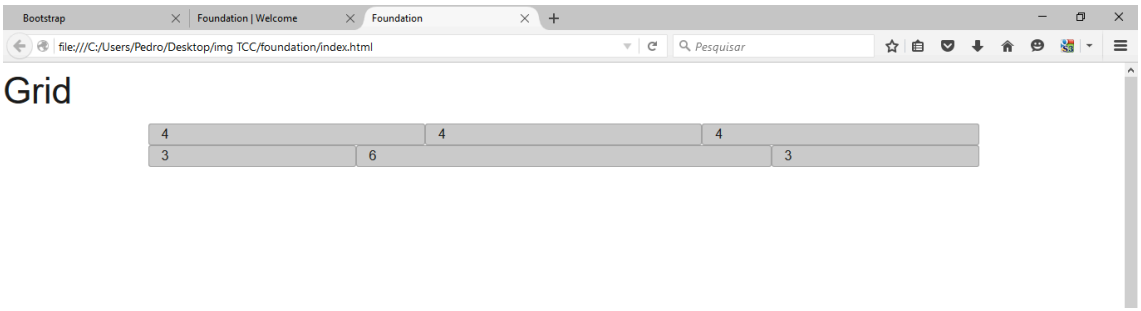


Figura 21. Exibição do sistema de grid Foundation

Para a criação de uma *grid* fluida no *Uikit* é necessário utilizar a classe "uk-grid". Logo em seguida inserir nas colunas uma outra div informando qual será o tamanho coluna na classe ".uk-width-1-valor". O valor pode variar de 1 a 10.

A figura 22 mostra a criação de uma grid no Uikit.

```
<body>
    <h1>Grid</h1>
    <div class="uk-grid">
        <div class="uk-width-1-3 border">Grid 1</div>
        <div class="uk-width-1-3 border">Grid 2</div>
        <div class="uk-width-1-3 border">Grid 3</div>
    </div>
</body>
```

Figura 22. Código de criação da grid no Uikit

A figura 23 mostra qual é o resultado da utilização do código mostrado na figura 22.

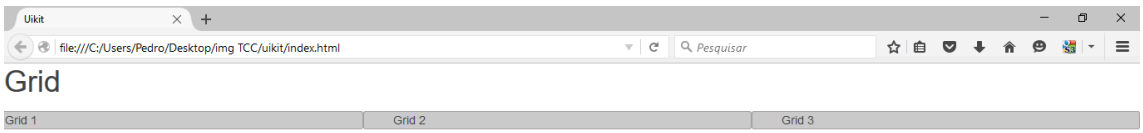


Figura 23. Exibição do sistema de grid Uikit

Os sistemas de *grid* são muito parecidos. Tendo apenas algumas diferenças na maneira conforme se cria as colunas. O *Bootstrap* utiliza uma nomenclatura confusa, o que pode confundir seus usuários iniciais, diferentemente do *Foundation* e *Uikit*.

4.2.2 Imagens Responsivas

A utilização de imagens responsivas em um layout todo responsivo é de suma importância.

Para isso o *Bootstrap* simplificou com a maneira como se torna uma imagem responsiva, bastando apenas adiciona a classe ".img-responsive" no elemento img, conforme mostra a figura 24.

```
<body>
  <img src="ces-jf.jpg" class="img-responsive">
</body>
```

Figura 24. Deixando uma imagem responsiva com Bootstrap

A figura 25 mostra qual é o resultado da utilização do código mostrado na figura 24.

Figura 25. Exibição de uma imagem responsiva Bootstrap

No *Uikit* o processo é bem semelhante bastando apenas adicionar a classe ".uk-responsive-width" ao elemento img. Conforme mostra a figura 26.

```
<body>
    <img class="uk-responsive-width" src="ces-jf.jpg">
</body>
```

Figura 26. Deixando uma imagem responsiva com Uikit

A figura 27 mostra qual é o resultado da utilização do código mostrado na figura 26.

Figura 26. Exibição de uma imagem responsiva Uikit

As classes do *Bootstrap* e *Uikit* utilizam um max-width com um valor de 100%, o que tornas as imagens responsivas.

O *Foundation* trata as imagens de uma maneira bem diferente. Ele visa deixar mostrar a imagem de acordo com o dispositivo que está acessando o *website* naquele determinado momento.

Dentro da estrutura do elemento img coloca-se o atributo *data-interchange* onde como primeiro parâmetro é passado o lugar onde a imagem se encontra e no segundo parâmetro é a consulta de mídia. Ele verifica se a página está sendo acessada por um *desktop* e mostra a devida imagem para aquele tipo de dispositivo. A figura 27 mostra como o *Foundation* exibe uma imagem.

```
<body>
    <img data-interchange="[ces-jf.jpg, (small)], [ces-jf-2.jpg, (large)]">
</body>
```

Figura 27. Deixando uma imagem responsiva com Foundation

A figura 28 e 29 mostra como é o comportamento de uma imagem diante da utilização do atributo data-interchange do framework Foundation.

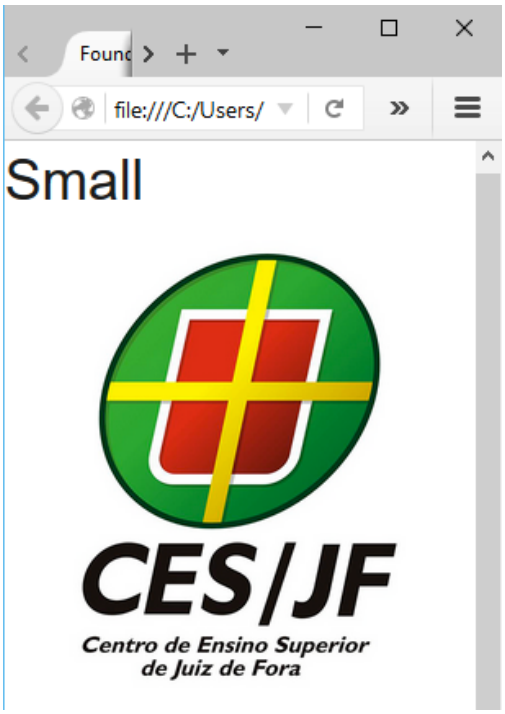


Figura 28. Exibição imagem responsiva small

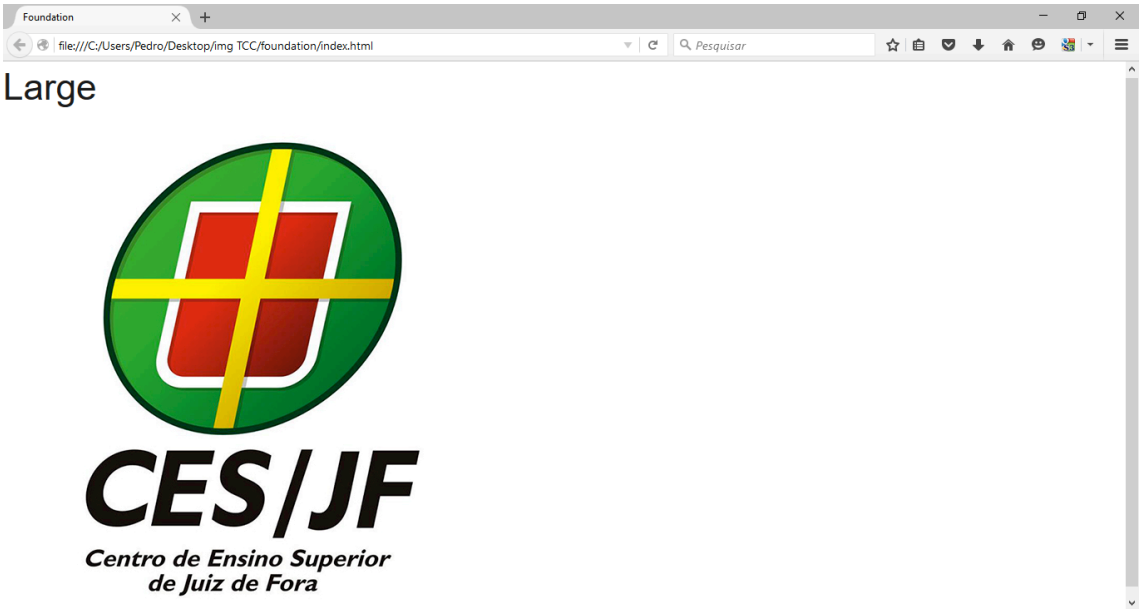


Figura 29. Exibição imagem responsiva large

O processo de redimensionamento de imagens no *Bootstrap* e no *Uikit* são bem simples. Porém, dependendo do tamanho da imagem original, a página web pode perder performance devido ao fato de carregar a imagem original e somente depois fazer o redimensionamento da mesma para o tamanho necessário ao dispositivo.

O *Foundation* trabalha de uma forma mais inteligente. Nele podem ser inseridas uma imagem para vários tamanhos de telas, e assim de acordo com o dispositivo que acessar a página ele exibe uma imagem correspondente para aquele tamanho de tela. O que torna a performance muito melhor.

5 Conclusão

O objetivo do trabalho foi realizar um estudo sobre as técnicas utilizadas pelo *web design responsivo* e uma breve comparação entre alguns frameworks do mercado, *Bootstrap*, Foundation e Uikit.

O *web design responsivo* é uma tecnologia que veio para ficar e evoluir, pois a cada dia que passa o número de dispositivos móveis só aumenta e o surgimento de novas tecnologias com acesso à páginas *web* também é cada vez maior.

Com relação aos *frameworks* é possível afirmar que as três plataformas são ótimas para o desenvolvimento de aplicações web, porém o *Bootstrap* e o *Uikit* possuem uma variedade maior de componentes css e *javascript* muito maior do que o *Foundation*. Facilitando muito a vida do desenvolvedor. O *Bootstrap* é muito rígido com relação aos seus temas o que deixa os *layouts* com uma aparência muito similar.

O *Uikit* é bem parecido com o *Bootstrap*, mas possui algumas vantagens como leveza e documentação muito bem estrutura. Ele é indicado para desenvolvedores que estiverem iniciando o processo de criação de websites utilizando *frameworks*.

O *Foundation* dá um poder de personalização maior que os seus concorrentes. Por isso ele é indicado para desenvolvedores que já possuem uma certa familiaridade com e framework e é indicado para projetos maiores.

Referências

Bootstrap. Disponível em <http://globocom.github.io/bootstrap/> Acessado em 19 de setembro de 2015.

Foundation. Disponível em <http://foundation.zurb.com/> Acessado em 19 de setembro de 2015.

Prostt, M. E. **Interface Web Utilizando Design Responsivo: um estudo de caso aplicado a smartphones, tablets, computadores e televisores.** Disponível em

<http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2513/1/CT_TECJAVMOV_I_2012_12.pdf> Acessado em 5 de setembro de 2015.

Silva, A. A. P. **Design Responsivo: técnicas, frameworks e ferramentas.** Disponível em <http://bsi.uniriotec.br/tcc/201412Almeida.pdf> Acessado em 5 de setembro de 2015.

Silva, M. S. **Web Design Responsivo:** aprenda a criar sites que se adaptam automaticamente a qualquer dispositivo, desde desktops até telefones celulares. São Paulo: Novatec, 2014.

Souza, S. C. N.; Igarashi, W. **Web Design Responsivo no desenvolvimento de aplicações multi-dispositivos**. Disponivel em

<http://www.espweb.uem.br/site/files/tcc/2012/Saulo%20Campos%20Nunes%20de%20Souza%20%20Web%20design%20responsivo%20no%20desenvolvimento%20de%20aplicacoes%20multi-dispositivos.pdf >Acessado em 7 de setembro de 2015.

Uikit. Disponível em <http://getuikit.com/> Acessado em 19 de setembro de 2015.

Zemel, T. **Web Design Responsivo:** páginas adaptáveis para todos os dispositivos. São Paulo: Casa do Código, 2012