UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS**

SISTEMAS DE INFORMAÇÃO

ANTONIO AUGUSTO CASTANHEIRA

ANÁLISE DE DESEMPENHO DE WEB SITES RESPONSIVOS

Bandeirantes
2015

ANTONIO AUGUSTO CASTANHEIRA

ANÁLISE DE DESEMPENHO DE WEB SITES RESPONSIVOS

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Estevan Braz Brandt Costa

Bandeirantes
2015

ANTONIO AUGUSTO CASTANHEIRA

ANÁLISE DE DESEMPENHO DE WEB SITES RESPONSIVOS

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____
Prof.Me. Estevan Braz Brandt Costa
UENP –*Campus* Luiz Meneghel

_____
Prof. Me. Ricardo Gonçalves Coelho
UENP – *Campus* Luiz Meneghel

_____
Prof. Esp. Wellington Della Mura
UENP – *Campus* Luiz Meneghel

Bandeirantes, 19 de agosto de 2015

RESUMO

A crescente evolução das tecnologias e a redução dos preços dos dispositivos móveis tem ocasionado o crescimento do número de páginas *Web* voltadas para estes dispositivos. Essas páginas recebem um tratamento diferente do que internet estava habituada. Este tratamento é denominado *Web* design responsivo. O presente trabalho avalia a performance de um site quando aplicadas as técnicas de *Web* design responsivo. Para tanto, o trabalho, realizou uma analise de desempenho entre páginas que foram tratadas com técnicas de *Web* design responsivo e páginas não tratadas. A coleta de informação do tempo de carregamento das páginas é realizada pelo uso da *API Navigation Timing*, embasado nas comparações entre os dados obtidos nos testes realizados. Conclui-se quais são os reais impactos da utilização do *Web* design responsivo em uma página.

Palavras-chave: *Web* Design Responsivo, *Benchmarking,* Desenvolvimento *Web*.

ABSTRACT

The evolution of technology and the reduction in the prices of mobile devices has caused the increment in the number of Web pages focused on these devices. These pages receive a different treatment than the internet was used to. This treatment is called responsive web design. This work evaluates the performance of a site when applied to the responsive web design techniques. To this end, the work, conducted a performance analysis between pages that have been treated with responsive web design techniques and untreated pages. The collection page loading time the information is accomplished by using the Navigation Timing API, based on comparisons between data obtained in the tests. We conclude what the real impacts of using responsive web design on a page.

Keywords*: Responsive Web Design, Benchmarking, Web Development.*

LISTA DE FIGURAS

Figura 1: Gráfico da performance de dez sites relevantes. Fonte: Adaptado de SOUDERS, 2012. 13
Figura 2: Exemplo de código HTML. Fonte: W3C, 2014. 16
Figura 3: Exemplo de DOM. Fonte: W3C, 2004. 17
Figura 4: Funcionamento do DOM. Fonte: Adaptado de GRIGORIK, 2014. 18
Figura 5: Funcionamento do CSSOM. Fonte: Adaptado de GRIGORIK, 2014. 19
Figura 6: Construção da árvore de renderização. Fonte: Adaptado de GRIGORIK, 2014. 20
Figura 7: Visualização de *Web* sites. Fonte: ZEMEL, 2012. 21
Figura 8: Layout flexível Proporção dos Elementos. Fonte: LOPES, 2013. 22
Figura 9: *Viewport* utilizada em um *iPhone*. Fonte: ZEMEL, 2012. 25
Figura 10: Dispositivos e resoluções. Fonte: NATDA, 2013. 26
Figura 11: Suporte da API *Navigation Timing*. Fonte: CANIUSE, 2015 27
Figura 12: Modelo de processo da API *Navigation Timing*. Fonte: W3C, 2012. 28
Figura 13: Funcionamento da ferramenta. Fonte: Autoria própria. 36
Figura 14: Tela inicial do sistema. Fonte: Autoria própria 37
Figura 15: Formula do carregamento crítico do DOM. Fonte: Adaptado de GRIGORIK, 2014. 38

LISTA DE TABELAS

Tabela 1 - Descrição dos atributos da tag viewport. 24
Tabela 2 - Atributos da Interface Performance Timing 28
Tabela 3 - Comparativo entre protótipos de página estático e responsivo. 35
Tabela 4 - Resultados obtidos por meio do navegador Google Chrome. 40
Tabela 5 - Resultados obtidos por meio do navegador Internet Explorer. 43

LISTA DE GRÁFICOS

Gráfico 1: Google Chrome executando protótipo página na simulação de *desktop.* Fonte: Autoria própria. 41
Gráfico 2: Google Chrome executando o protótipo página na simulação *tablet.* Fonte: Autoria própria 41
Gráfico 3: Google Chrome executando o protótipo página na simulação mobile. Fonte: Autoria própria. 42
Gráfico 4: - Internet Explorer executando o protótipo página na simulação desktop. Fonte: Autoria própria. 43
Gráfico 5: Internet Explorer executando o protótipo página na simulação tablete. Fonte: Autoria própria. 44
Gráfico 6: Internet Explorer executando o protótipo página na simulação mobile. Fonte: Autoria própria. 45

LISTA DE SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
CSSOM	*Cascading Style Sheets Object Model*
CSV	*Comma Separeted Values*
DOM	*Document Object Model*
HTML	*Hyper Text Markup Language*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
PHP	*PHP Hyper Text Preprocessor*
TCP	*Transmission Control Protocol*
TXT	*Text*
W3C	*World Wide Web Consortium*

SUMÁRIO

1. INTRODUÇÃO 11
1.1. CONTEXTO E DELIMITAÇÃO DO TRABALHO 11
1.2. FORMULAÇÃO DO PROBLEMA 12
1.3. OBJETIVOS 12
1.3.1. OBJETIVO GERAL 12
1.3.2. OBJETIVOS ESPECÍFICOS 12
1.4. JUSTIFICATIVA 13
1.5. METODOLOGIA 14
1.6. ORGANIZAÇÃO DO TRABALHO 14
2. FUNDAMENTAÇÃO TEÓRICA 15
2.1. WEB 15
2.2. HTML 15
2.3. *CSS* 16
2.4. DOM 16
2.5. CSSOM 18
2.6. ÁRVORE DE RENDERIZAÇÃO 19
2.7. *WEB* DESIGN RESPONSIVO 20
2.7.1. RESOLUÇÕES DE TELA 21
2.7.2. ADAPTAÇÃO DE INTERFACES 21
2.7.3. *LAYOUT* FLUIDO 22
2.7.4. IMAGENS E RECURSOS FLEXÍVEIS 23
2.7.5. MEDIA QUERIES 23
2.7.6. TIPOS E MEDIDAS EM CSS 23
2.7.7. META TAG VIEWPORT 24
2.7.8. BREAKPOINT 25
2.8. BENCHMARKING ..... **Error! Bookmark not defined.**
2.9. API NAVIGATION TIMING 26
3. DESENVOLVIMENTO 32
3.1. PROTÓTIPOS 32
3.1.1. LAYOUT 33
3.1.2. FONTES 33
3.1.3. IMAGEM 34
3.1.4. VÍDEO 34
3.1.5. PÁGINA 34
3.2. FERRAMENTA 35
3.3. USO DOS PROTÓTIPOS NA FERRAMENTA 37
3.4. ANÁLISE DOS DADOS 38
4. RESULTADOS OBTIDOS 40
4.1. GOOGLE CHROME 40
4.2. INTERNET EXPLORER 42
4.3. ANÁLISE DOS RESULTADOS 45
5. CONCLUSÂO 47
REFERÊNCIAS 48

1. INTRODUÇÃO

A interface gráfica é responsável pela interação com o usuário, sem ela não seria possível estabelecer a troca de informações entre homem e máquina. De acordo com Lopes (2013) no surgimento da internet, as interfaces dos sites eram rudimentares, não amigáveis e sem muita interatividade. Com o passar dos anos surgiram novas tecnologias e técnicas que possibilitaram a evolução das interfaces gráficas. Atualmente os novos desafios das interfaces gráficas são os dispositivos móveis.

Segundo Zemel (2012) a internet estava habituada com interfaces estáticas e de tamanho fixo, o que funcionou perfeitamente durante muitos anos, o problema é que os dispositivos móveis não se adaptam muito bem a estes tipos de interfaces. De acordo com Lopes (2013) um site com interfaces estáticas não proporcionam uma boa navegação ao usuário de dispositivos móveis. O problema a ser resolvido é a capacidade de os sites adaptarem suas interfaces gráficas para os dispositivos móveis.

De acordo com Marcotte (2010), as técnicas de design responsivo têm com o objetivo fornecer páginas capazes de suportar mudanças de *layouts* para diferentes resoluções, a fim de atender a qualquer tipo de dispositivo com acesso a internet. O usuário não tem que se preocupar com qual dispositivo irá acessar determinado site, e poderá utilizar todos os recursos disponíveis pela página (LOPES, 2013).

Marcotte (2010) ressalta que, o design responsivo consegue solucionar o problema de fornecer páginas a qualquer dispositivo, porém se tratando de dispositivos com baixa capacidade de processamento, a performance é um fator que deve ser levado em consideração. A área de estudos voltados para performance de interfaces de dispositivos móveis ainda é muito nova. O presente trabalho pretende realizar um estudo comparativo entre o design responsivo e o desenvolvimento tradicional de sites.

1.1. CONTEXTO E DELIMITAÇÃO DO TRABALHO

Segundo Meenan (2013), o processo de carregamento de uma página o se inicia após o navegador encontrar o endereço IP (*Internet Protocol*) da página, em seguida estabelece uma conexão utilizando o TCP (*Transmission Internet Protocol*) com o servidor, e envia requisições do código HTML, após o navegador baixar todo

o código HTML, ele processa e executa esse código, como resultado final a página é apresentada ao usuário.

Este processo pode ser dividido em três grupos: rede, servidor e navegador (MEENAM, 2013). Ao realizar esta divisão, facilita o processo de verificação de funcionamento de cada grupo em especifico. O trabalho foi realizado utilizando a uma conexão local, descartando quaisquer aspectos ligados a redes e servidores.

O estudo tem como foco medir a performance dos sites no navegador, onde segundo Grigorik (2014) são realizadas as interpretações do HTML e CSS, mais especificamente nos modelos de objeto, DOM (*Document Object Model*) e o CSSOM (*Cascading Style Sheets Object Model*).

1.2. FORMULAÇÃO DO PROBLEMA

De acordo com Zemel (2012), o uso das técnicas de design responsivo afeta o comportamento de uma página, elas passam a oferecer diferentes interfaces gráficas dependendo do dispositivo que efetuou a requisição.

A questão levantada no trabalho é a relação entre performance e design responsivo, o uso de medições do tempo de renderização de uma página são indispensáveis para a *Web* atual.

Neste contexto, a seguinte questão é levantada: O desempenho de uma página *Web* é afetado quando aplicado às técnicas de design responsivo?

1.3. OBJETIVOS

Neste tópico são descritos as metas e objetivos deste trabalho.

1.3.1. OBJETIVO GERAL

O objetivo geral dessa pesquisa é realizar uma análise comparativa entre sites convencionais e sites que utilizam técnicas de design responsivo.

1.3.2. OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral deste trabalho a pesquisa atuará sobre os seguintes objetivos específicos:

- Elencar as técnicas de design responsivo;
- Desenvolver protótipos utilizando conceitos de páginas responsivas;
- Desenvolver uma ferramenta de teste;

- Avaliar os protótipos desenvolvidos utilizando a ferramenta;
- Analisar os dados obtidos.

1.4. JUSTIFICATIVA

O número de pessoas que acessam a internet cresce a cada dia, mais de 1.2 bilhões de pessoas, e quase metade destes usuários, esperam que a página seja carregada em 2 segundos ou menos (SHOPFY, 2015).

A velocidade de carregamento da página influencia diretamente na decisão do usuário, por meio dela o usuário define se vai ou não, retornar ao site. Uma mudança realizada pela Google, aumentou de 10 para 30 resultados mostrados em uma busca, a mudança resultou em um salto no tempo de carregamento da página de 0.4 segundos para 0.9 segundos, após o aumento ouve uma queda de 20% nas buscas do site (MAYER, 2006).

Na Figura 1 pode-se observar que a pesquisa realizada por Souders (2012), constatou que entre dez site relevantes atualmente, apenas 24% representam o tempo de processamento do servidor, os outros 76% são referentes a interface gráfica.

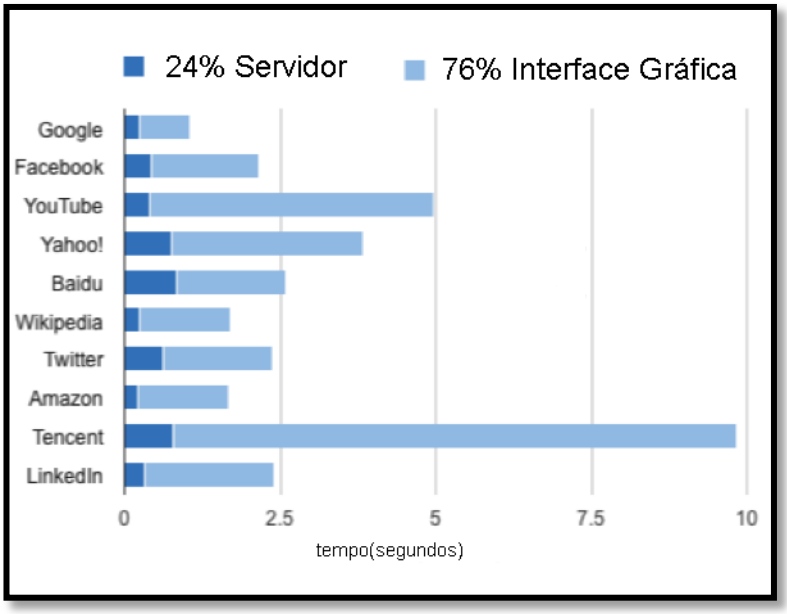

Figura 1: Gráfico da performance de dez sites relevantes. Fonte: Adaptado de SOUDERS, 2012.

Ao final deste trabalho espera-se obter uma posição em relação a incerteza se a performance de um site é afetada, ou não, quando se aplica técnicas de design responsivo.

1.5. METODOLOGIA

Este estudo caracteriza-se como uma pesquisa de caráter experimental, sendo um estudo comparativo entre páginas desenvolvidas com design responsivo e páginas estáticas.

O presente trabalho foi elaborado da seguinte maneira:

–**Revisão de Literatura:** para a realização do trabalho foram estudados alguns temas relacionados com a criação de páginas *Web* como HTML, CSS, performance de páginas *Web,* DOM, CSSOM.

–**Estudo das técnicas de design responsivo**: foram analisadas as técnicas para a criação de páginas responsivas como *layout* flexível, *media queries*, *meta tag viewport*, resoluções de tela*,* tipos de medida em CSS, *breakpoints.*

–**Estudo da API *Navigation Timing*:** para criação da ferramenta foi necessário um estudo da API *Navigation Timing* para entender seu funcionamento e o que é possível realizar com o seu uso.

–**Desenvolvimento dos protótipos:** foram desenvolvidos dez protótipos utilizando técnicas de design responsivo e técnicas tradicionais de desenvolvimento de páginas *Web*.

–**Desenvolvimento da ferramenta:** a ferramenta denominada *RWD Benchmark* foi desenvolvida para possibilitar o teste do tipo benchmarking e também simular diferentes tipos tamanhos de tela.

–**Coleta de dados:** com a ferramenta *RWD Benchmark* desenvolvida, foi possível testar cada protótipo utilizando as simulações de tamanhos de tela fornecidas pela ferramenta.

–**Análise dos dados obtidos:** após coletados de dados dos protótipos a partir do uso da ferramenta proposta, foi realizada as análises dos resultados, afim de concluir o problema proposto.

1.6. ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte maneira: a seção 2 apresenta o referencial teórico, com embasamento da pesquisa; a seção 3 contém o desenvolvimento do trabalho; a seção 4 exibe os resultados obtidos e análise dos resultados; a seção 5 mostra as conclusões.

2. FUNDAMENTAÇÃO TEÓRICA

Neste trabalho, serão abordados conceitos relacionados com a *Web* como *HTML, CSS*, funcionamento do DOM e CSSOM e a árvore de renderização. Também são abordados temas como *Web* design responsivo, resoluções de tela, adaptação de interfaces, layout fluido, imagens e recursos flexíveis, tipos e medidas em CSS, *meta tag viewpot* e *breakpoint*. E por fim temas como *benchmarking* e a API *Navigation Timing.*

2.1. WEB

A *Web* pode ser definida como um conjunto de documentos acessíveis por meio da Internet. Esses documentos, conhecidos por páginas *Web*, contêm uma tecnologia chamada hipertexto, através da qual é possível percorrer partes do documento e alcançar outros documentos através de ligações entre eles, conhecidos como *links* (REIS, 2007).

Para a criação de uma página *Web* usam-se textos e comandos especiais que fazem parte da linguagem HTML. A finalidade do HTML é formatar o texto exibido para criar ligações entre as páginas *Web*, gerando assim documentos com o conceito de hipertexto. Para que o conteúdo HTML possa ser formatado e exibido na Internet, existem os navegadores, programas que leem o conteúdo do arquivo, interpretam os comandos e exibem sua página *Web* (REIS, 2007).

Atualmente existem navegadores em diversos dispositivos como *notebooks*, *tablets*, *smartphones* e televisores (ZEMEL, 2012). Segundo Marcotte (2012) os sites precisam estar disponíveis para os dispositivos móveis, e os recursos como HTML e CSS dão forma as páginas que são apresentadas ao usuário.

2.2. HTML

Em 1989, nascia a linguagem de marcação de hipertexto, o HTML, com o objetivo de facilitar a comunicação e a disseminação de pesquisas e trabalhos científicos. Com o crescimento da Internet pública e com as padronizações da linguagem, o HTML tornou-se o principal meio de transmissão e visualização das informações via páginas *Web* no mundo (W3C, 2012).

A Figura 2 demonstra um exemplo simples utilizando HTML, o código imprime na tela do usuário um título e um *link.* (W3C, 2014).

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample page</title>
 </head>
 <body>
  <h1>Sample page</h1>
  <p>This is a <a href="demo.html">simple</a> sample.</p>
  <!-- this is a comment -->
 </body>
</html>
```

Figura 2: Exemplo de código HTML. Fonte: W3C, 2014.

De acordo com a W3C (2012) o navegador *Web* ao ler um documento HTML, interpreta as *tags* que contém no documento, e processa como serão as informações serão exibidas para o usuário.

2.3. *CSS*

Em 1996, a W3C lançava oficialmente a recomendação de separação de folha de estilo e formatação do código HTML padrão, o CSS, no intuito de separar a formatação da informação do resto do código HTML, evitando problemas de interpretação dos navegadores e facilitando a compreensão do código por parte dos desenvolvedores (MAZZA, 2012).

O CSS formata a informação entregue pelo HTML, não importando o formato de mídia que ela se encontre, seja em imagem, texto, vídeo, áudio ou qualquer outro elemento criado. Essa formatação, na maioria das vezes, é visual, mas não necessariamente, e prepara essa informação para que ela seja consumida da melhor maneira possível (W3C, 2013).

Em sua terceira versão, está presente uma nova característica, que se baseia na modularização da linguagem, na qual cada módulo tem atualização independente dos outros, tornando a melhora do código contínua e mais rápida. Outra vantagem da modularização é a capacidade de dispositivos específicos terem a possibilidade de escolher quais módulos suportam, assim focando recursos em funções mais importantes ao sistema (MAZZA, 2012).

2.4. DOM

Segundo a W3C (2004), o DOM é uma plataforma e linguagem de interface

neutra que permite programas e scripts acessarem e atualizarem dinamicamente o conteúdo, estrutura e estilos de páginas *Web*. As páginas podem ser processadas e os resultados do processamento podem ser incorporados de volta à página.

Com o DOM, os desenvolvedores podem criar e construir documentos, navegar em sua estrutura, e adicionar, modificar ou excluir elementos e conteúdos (MAZZA, 2012).

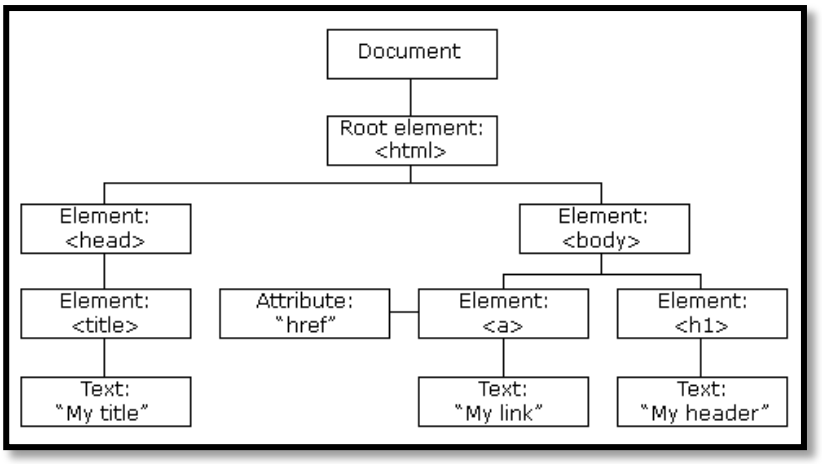

Figura 3: Exemplo de DOM. Fonte: W3C, 2004.

De acordo com a W3C (2004) o DOM é utilizado no auxílio de renderização páginas HTML nos navegadores, na Figura 3 é demonstrado como é a interpretação do HTML após ser transformado no DOM.

No DOM, os documentos têm uma estrutura lógica semelhante com uma árvore, como no HTML, os elementos são filhos de outros elementos. Esta estrutura possibilita uma organização eficiente na hora de organizar os dados e aplicar estilos (W3C, 2004).

Outra característica desta estrutura é a forma como é processado, permitindo que qualquer navegador atual consiga processar os elementos. Existem *tags* e estilos que alguns navegadores, como o Internet Explorer 8.0 não dá suporte a *tag* vídeo, lançada na versão 5 do HTML. Quando o navegador não fornece suporte à alguma funcionalidade, ele simplesmente ignora o comando (MAZZA, 2012).

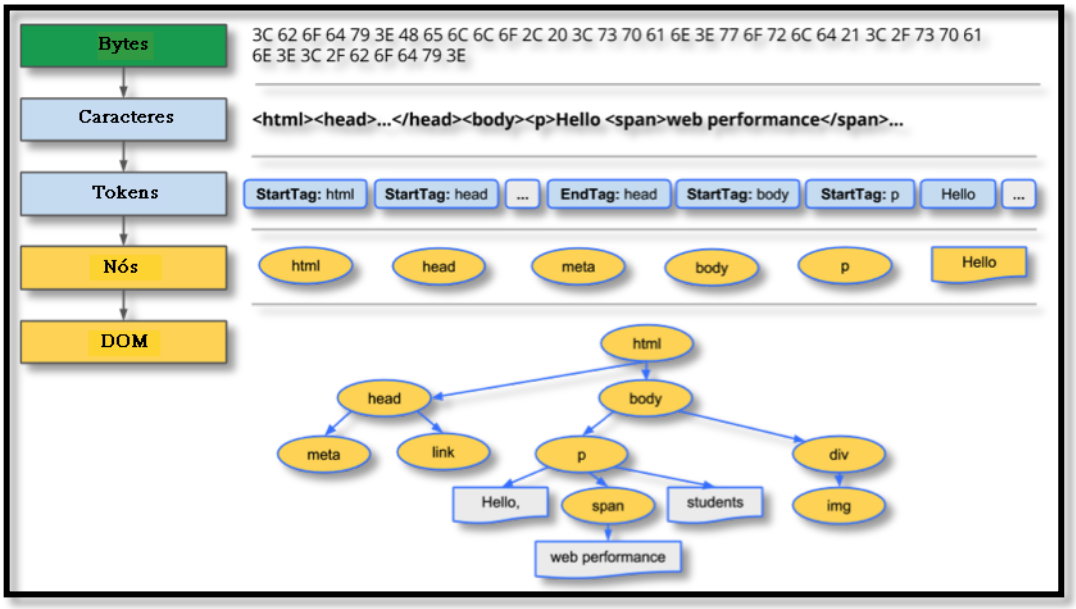

Figura 4: Funcionamento do DOM. Fonte: Adaptado de GRIGORIK, 2014.

Segundo Grigorik, (2014) o processo de interpretação de um HTML até que ele se transforme em um DOM é dividido em cinco etapas como mostra a Figura 4.

Conversão: o navegador lê os bytes brutos do HTML do disco ou da rede e os traduz em caracteres individuais baseados na codificação determinada do arquivo (por exemplo, UTF-8).

Criação de tokens: o navegador converte as *strings* de caracteres em tokens distintos especificados pelo padrão W3C HTML5, por exemplo, <html>, <body> e outras *strings* nos colchetes angulares. Cada *token* tem um significado e conjunto de regras especiais.

Lexicalização: Os *tokens* emitidos são convertidos em objetos semelhantes aos nós, onde nó, define suas propriedades e regras.

Criação do DOM: por último, como a marcação de HTML define as relações entre diferentes *tags*, os objetos criados são vinculados em uma estrutura de dados em forma de árvore que também estabelece as relações pai-filho definidas na marcação original: o objeto HTML é pai do objeto *body, body* é pai do objeto parágrafo e assim por diante.

2.5. CSSOM

O CSSOM é um objeto de modelo muito semelhante ao DOM, porém é referente ao CSS. As principais funções do CSSOM são no sentido de fornecer recursos para a manipulação de estilos de uma página HTML (W3C, 2004).

Tanto o CSSOM quanto o DOM bloqueiam a renderização da página, ou seja, enquanto o navegador não processar ambos, a página não é exibida para o usuário (LEWIS, 2015).

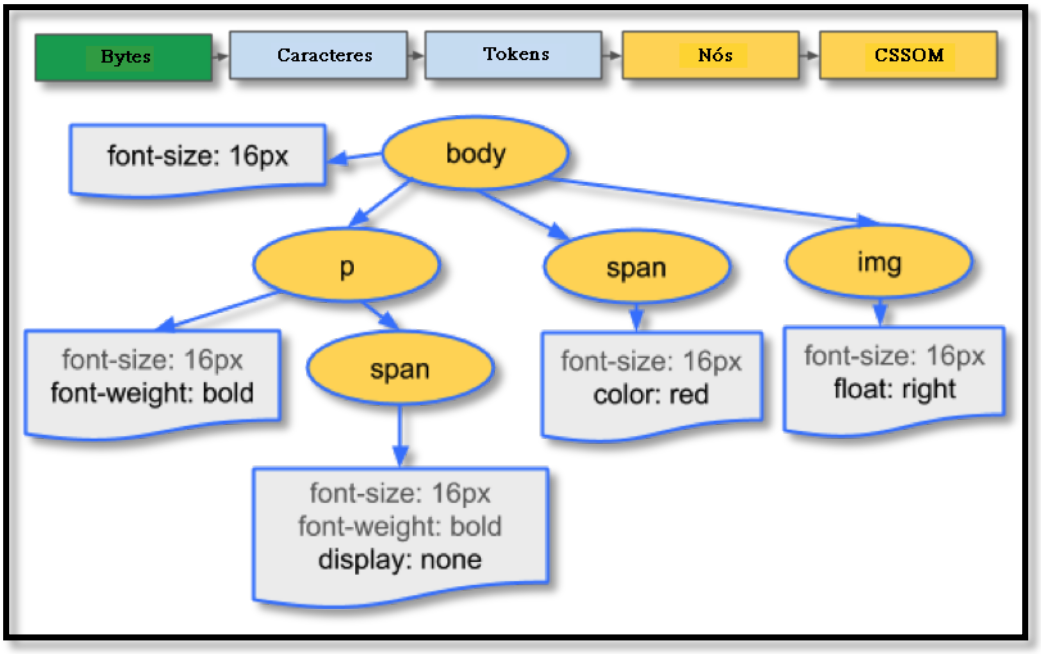

Figura 5: Funcionamento do CSSOM. Fonte: Adaptado de GRIGORIK, 2014.

De acordo com Grigorik (2014) o processo de interpretação do CSSOM também é dividido em cinco etapas como no DOM, a diferença se encontra na última etapa, onde os nós são interpretados pelo navegador e é criada uma estrutura em árvore conhecida como CSSOM, que contém as instruções de estilos prontas para serem aplicadas aos elementos *HTML*.

2.6. ÁRVORE DE RENDERIZAÇÃO

Nas sessões anteriores foi apresentado como o navegador interpreta os modelos de objetos DOM e CSSOM. Nesta sessão será apresentada como funciona o processo de renderização depois que os modelos de objetos estão previamente processados.

Segundo Lewis (2015), a Árvore de Renderização ou *Render Tree*, só pode ser montada depois que o DOM e CSSOM estiverem prontos pois são modelos de objetos independentes, o navegador então tem como função unir as duas árvores. A Figura 6 representa o funcionamento da árvore de renderização, o processo começa na raiz da árvore do DOM, o navegador analisa todos os nós visíveis. A Figura 6 descreve o funcionamento na árvore de renderização.

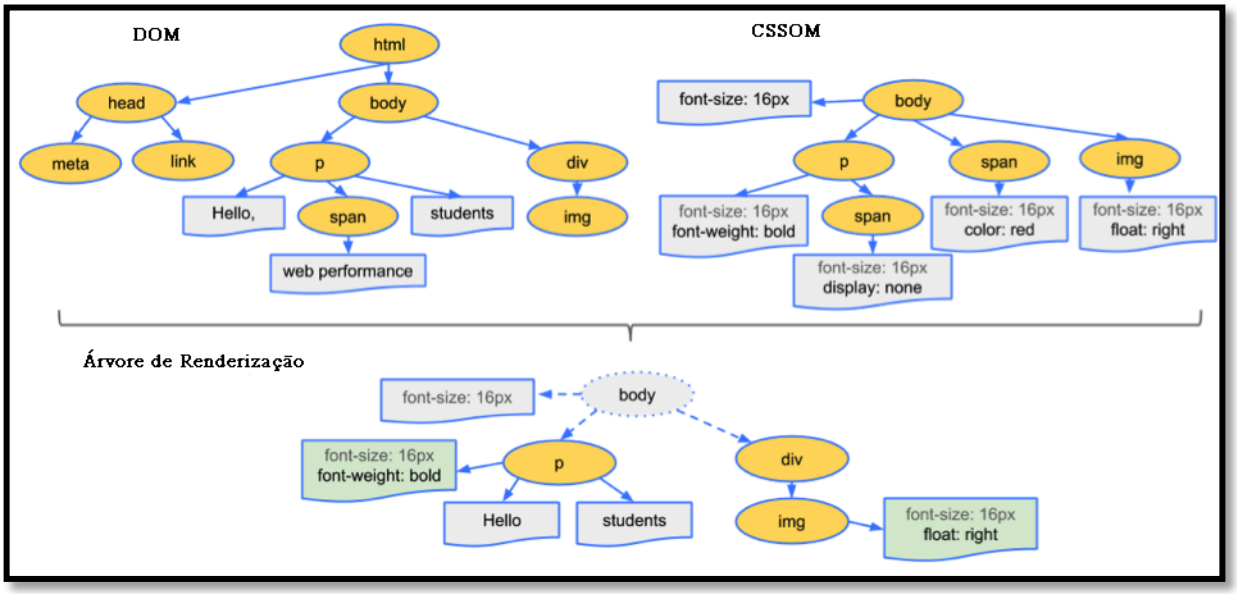

Figura 6: Construção da árvore de renderização. Fonte: Adaptado de GRIGORIK, 2014.

Alguns nós como *tags* de script ou *meta tags* não são necessárias na construção da árvore e são omitidos pois não são refletidos no resultado da renderização. (GRIGORIK, 2014).

Por fim depois de processar todos os elementos de ambas as árvores, a árvore de renderização está pronta para ser processada pelo navegador e para realizar um processo chamado *Paint*, que nada mais é do que o processo de pintar os elementos na tela do usuário (LEWIS, 2015).

2.7. *WEB* DESIGN RESPONSIVO

No ano de 2010 um desenvolvedor chamado Ethan Marcotte, publicou um artigo intitulado "*Responsive Web Design* no site *A List Apart*" que mencionava conceitos e sugestões usando a tecnologia da época, para que as páginas fossem responsivas, ou seja, responde a quaisquer dispositivos ou resolução (ZEMEL, 2012).

O controle que os designers têm no meio impresso, muitas vezes, desejam ter no meio *Web*, é simplesmente um reflexo da limitação da página impressa. Devemos aceitar o fato de que a *Web* não tem as mesmas restrições e projetar o *Web* design para essa flexibilidade (MARCOTTE, 2010).

2.7.1. RESOLUÇÕES DE TELA

Hoje em dia, dispositivos com telas de diferentes tamanhos, desde celulares a desktops com grandes telas, são utilizados para acessar a web. (MARCOTTE, 2010).

De acordo com Lopes (2013), não é viável dar suporte específico a todos os dispositivos que estão disponíveis e que ainda vão ser desenvolvidos. Tendo isto em mente, o que importa não é o tamanho físico ou o dispositivo em si, mas sua resolução.

É com base na resolução que o design responsivo trabalha e a partir dele é possível desenvolver para uma gama de dispositivos sem se preocupar com o tamanho da tela (LOPES, 2013).

Na Figura 7, são apresentadas as resoluções de telas mais comuns.

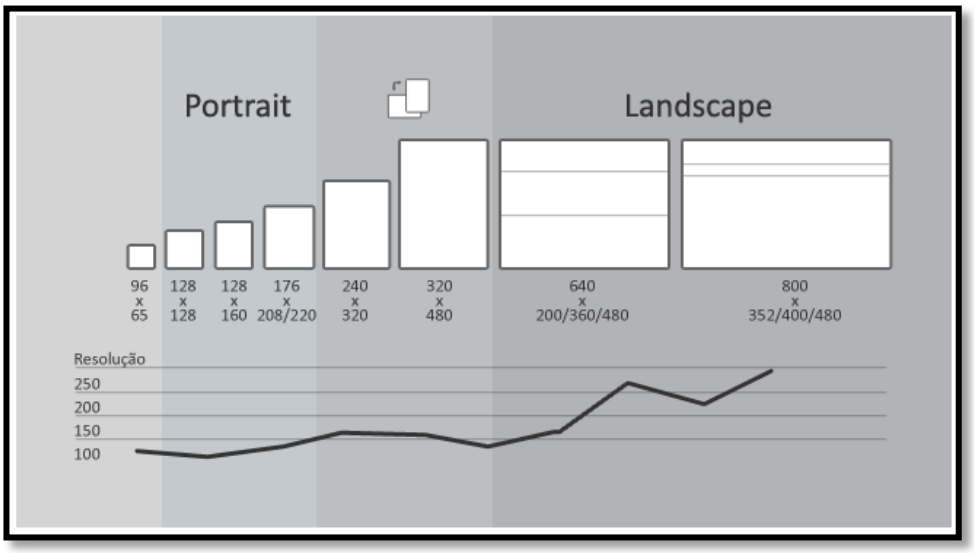

Figura 7: Visualização de *Web* sites. Fonte: ZEMEL, 2012.

Segundo Lopes, (2013) existem dois tipos de visualização de *Web sites* que podem ser observados na Figura 7, o *portrait* que é um retrato que possui a largura menor que a altura, é utilizado em celulares e *tablets,* o outro tipo de visualização é o *landscape* também conhecido como paisagem que possui a largura maior que altura, utilizado em *notebooks* e desktops e em celulares e *tablets* quando utilizados na horizontal.

2.7.2. ADAPTAÇÃO DE INTERFACES

Há três tecnologias básicas por trás do *design* responsivo, são elas: *HTML*, *CSS* e *JavaScript*. As técnicas fundamentais para design responsivo como *layouts*

flexíveis; *media queries,* que auxiliam a adaptação do conteúdo para os tamanhos de tela específicos; e as imagens e recursos flexíveis, que respondem a mudanças em tamanhos de tela (NATDA, 2013).

De acordo uma aplicação com *layout* responsivo pode ser bem visualizada em qualquer um dos dispositivos, por exemplo, uma TV, *tablet*, *smartphone*, *desktop* ou até mesmo em novas geladeiras que possuem tela com conexão à internet (ZEMEL, 2012).

2.7.3. *LAYOUT* FLUIDO

Um *layout* fluido ou como originalmente intitulado, *grid* flexível, é o primeiro passo para começar a pensar no *Web design* responsivo. Para se conseguir um layout fluido num projeto *Web*, a principal medida a ser tomada é: não usar medidas absolutas no CSS (MARCOTTE, 2010).

A técnica de *layout* fluido é baseada valores percentuais ao invés de absolutos, as porcentagens são usadas para especificar medidas de tamanho com relação ao tamanho do elemento pai (LOPES, 2013).

Ao especificar tamanhos, espaçamentos, margens, *paddings* ou qualquer medida fixa no site, gera uma incompatibilidade de adaptação do mesmo a outros tipos de tela (ZEMEL, 2012). Sendo assim, a melhor forma de especificar esses valores é a utilização das medidas porcentagem e EM, que serão explicadas na Sessão 2.76.

Os elementos de uma página que utiliza as técnicas de *layout* fluido devem se ajustar quando estão em diferentes resoluções, a Figura 8 demonstra como um *layout* deve se comportar em um celular.

Figura 8: Layout flexível Proporção dos Elementos. Fonte: LOPES, 2013.

2.7.4. IMAGENS E RECURSOS FLEXÍVEIS

Uma parte complexa da técnica de design responsivo é resolver como lidar com as imagens, normalmente elas são criadas a partir de um número fixo de pixels, não favorecendo o *layout* fluido.

Para a utilização de imagens e recursos flexíveis no desenvolvimento de *sites* responsivos é necessário à utilização do CSS para determinar efeitos sobre os recursos utilizados em um site (ZEMEL, 2012). Normalmente os recursos como imagens e vídeos são configurados para ocupar 100% do contexto do layout em que se ocupa (LOPES, 2013).

Colocar porcentagens fará com que a imagem aumente ou diminua de acordo com o tamanho da tela. Uma imagem de 300px, por exemplo, sendo esticada a 100% de uma tela de 1280px, ficaria com uma aparência bem ruim. De acordo com LOPES (2013) as imagens devem ser tratadas na especificação do código CSS, evitando com que os recursos como imagens aumentem seu tamanho de forma inadequada.

2.7.5. MEDIA QUERIES

Na versão CSS2 se utilizava uma função denominada *media type* usada para reconhecer um determinado dispositivo, que possuía alguns tipos definidos, como *Speech*, *Print*, *Projection*, *TV* entre outros (W3C, 2012).

Para resolver essas alternâncias de tipos de dispositivos, a versão do *CSS3* utiliza *media queries* (ZEMEL, 2012).

A técnica *medias queries* é a principal aliada do layout fluido, é uma técnica disponível no CSS que permite trabalhar com design condicional, para que os elementos possam comportar-se de maneiras diferentes à medida que resolução do navegador mude (LOPES, 2013).

O conceito de *breakpoint*, define os pontos onde o *layout* vai ser ajustado por causa de uma resolução diferente (LOPES, 2013).

2.7.6. TIPOS E MEDIDAS EM CSS

Segundo Lopes (2013) existem outros tipos de medidas em *CSS*, os 4 principais são: *pixels*, pontos, porcentagens e *sem* que serão descritos abaixo:

• ***Pixel*** **(px):** É a unidade de medida fixa mais usada nas *CSS*, um *pixel* é um ponto indivisível na tela de exibição de um dispositivo. Muitos *Web designers*

preferem usar este tipo de medida para fazer uma estrutura *HTML*/*CSS*, modelando um site funcional, mas que muitas vezes não atender as demandas de dispositivos.

• **Ponto (point):** Pontos são tradicionalmente utilizados no *CSS*. Um ponto é igual a 1/72 polegadas. Assim como *pixels*, pontos são unidades de tamanho fixo.

• **Ems (em):** O "*em*" é uma unidade escalável. Quando se trata do tamanho da fonte, 1em é igual ao tamanho atual da fonte do elemento- pai. Por exemplo, se o tamanho da fonte do elemento é 12pt, 1em é igual a 12pt. As medidas *em* são escaláveis, 2em seria igual a 24pt, 0.5 seria 6pt e assim por diante.

• **Porcentagem (%):** A unidade por cento é muito parecida com a unidade *em* mas possui algumas diferenças fundamentais, principalmente o fato de que o atual tamanho da fonte é igual a 100% (ou seja, 12pt = 100%). Durante o uso da unidade por cento, o texto permanece totalmente escalável para dispositivos móveis.

2.7.7. META TAG VIEWPORT

Os navegadores *mobile* tentam exibir páginas *Web* feitas somente para *desktop*, ajustando automaticamente o *zoom* do *display*, isso pode ser problemático para os sites que já foram planejados para telas pequenas (ROSA; SILVA, 2014).

Existe uma meta *tag* para contornar essa característica padrão dos navegadores. A *meta tag viewport*, como qualquer outra *metatag*, ela possui o formato: *<meta name="viewport" content="">*

Sendo que, em *content*, é possível especificar uma diversidade de parâmetros e valores conforme o tipo de visualização que se configurar às páginas. Com a *meta tag viewport*, é possível apresentar resoluções personalizadas aos visitantes para determinados dispositivos (ROSA; SILVA, 2014). Na Tabela 1 contém os principais e mais usados parâmetros da *tag viewport.*

Tabela 1 - Descrição dos atributos da *tag viewport*.

Valor	Descrição
width	Define uma largura para o *viewport*. Os valores podem ser em PX ou "*device-width*", que determina automaticamente um valor igual à largura da tela do dispositivo.
height	Define uma altura para o viewport. Os valores podem ser em PX ou "*device-height*", que determina automaticamente um valor igual a altura da tela do dispositivo.

Valor	Descrição
initial-scale	Define a escala inicial do *viewport*.
user-scalable	Define a possibilidade de o usuário fazer “zoom” em um determinado lugar da tela. É ativado quando o usuário bate duas vezes com o dedo em um lugar da tela.

Fonte: EIS, 2011

Segundo Zemel (2012) a Apple que originalmente propôs e colocou em uso a *meta tag viewport*, a Figura 9 apresenta o uso de uma *tag meta viewport* exposta em um iPhone, utilizando as configurações padrão de *viewport*.

Figura 9: *Viewport* utilizada em um *iPhone*. Fonte**:** ZEMEL, 2012.

2.7.8. *BREAKPOINT*

Um *breakpoint* indica o momento em que o *layout* muda de um esquema para outro, e é geralmente gerado pela largura de uma tela (LOPES, 2013).

De acordo com Natda (2013), as mudanças de *layout* são baseados nos tamanhos típicos de dispositivos, como celulares, *tablets notebooks* e *desktops*

como ser visto na Figura 10. Isso significa que uma das primeiras coisas a se fazer ao planejar um site responsivo, é definir os limites de resolução para o seu *layout*.

Figura 10: Dispositivos e resoluções. Fonte: NATDA, 2013.

Como exemplificado na Figura 10 os estilos móveis terão como alvo qualquer tela menor do que 479 *pixels* de largura, enquanto estilos *tablet* irá geralmente têm como alvo as telas entre 480 e 767 *pixels*, tela *tablet (wide)* entre 768 e 1023 *pixels*. Estilos de *desktop*, terão como alvo qualquer tela mais larga do que 1024 *pixels* (NATDA, 2013).

2.8. API NAVIGATION TIMING

Em 2010, os fabricantes de navegadores se reuniram em conjunto com W3C e formaram o *Web Performance Working Group*, com o objetivo de padronizar e melhorar as métricas de desempenho nos navegadores. No final de 2010 o grupo lançou uma especificação denominada API *(Application Programming Interface) Navigation Timing*, que consistia em coleta de dados do tempo de carregamento da página do ponto de vista do navegador (MEENAN, 2013).

Realizar testes é fundamental para a qualidade de um site, na maioria dos casos, os testes são realizados nos servidores e nas redes. Um ponto que muitas vezes é deixado de lado é a experiência do usuário.

De acordo com Baker (2012) a API *Navigation Timing* possibilita coleta de informações do navegador do usuário.

Segundo Google *Analytics* (2012), o sistema de análise de dados em tempo real do Google, faz uso dos recursos da API para realizar suas medições. E de acordo com Dutton (2013), o Google também possui um *plug-in* chamado *Page*

speed test que permite que medições com a API em tempo real.

Um fator determinante para a utilização da API é o suporte fornecido, de acordo com o site CANIUSE (2015) pode ser observado na Figura 11, que a API oferece suporte para a maioria dos navegadores, com exceção do Opera Mini.

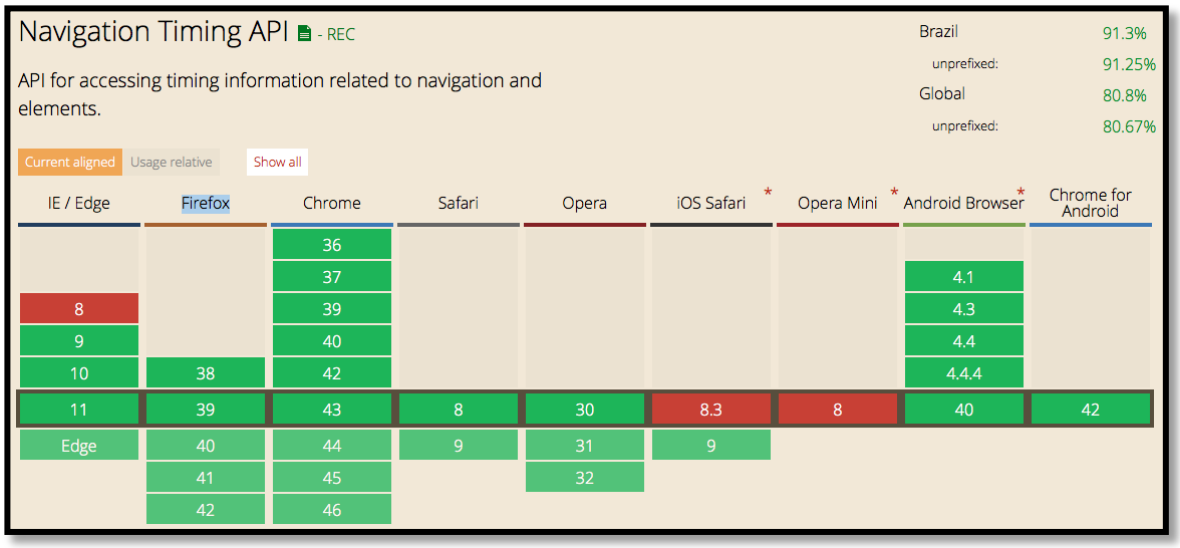

Figura 11: Suporte da API *Navigation Timing*. Fonte: CANIUSE, 2015

A interface *Performance Timing* encontrada na API é responsável por gravar horários, que representa um dado significativo no carregamento da página. Possui 21 atributos do tipo *long* para armazenar os tempos, que somente podem ser lidos, não escritos. (W3C, 2012).

Os atributos podem ser divididos em três grupos: rede, servidor e navegador. Um dos benefícios do uso da API, é fato que ele expõe uma série de dados que levam até o carregamento da página.

Segundo Meenan (2013), a cada etapa importante do carregamento a interface *Performance Timing* registra com precisão informações como: redirecionamentos, os tempos de pesquisa de DNS, tempo para se conectar ao servidor, quanto tempo leva para o servidor *Web* responder ao pedido de cada usuário e para cada página que o usuário visita.

O processo de carregamento de página, de acordo com a W3C (2012) API Navigation Timing, está descrito na Figura 12.

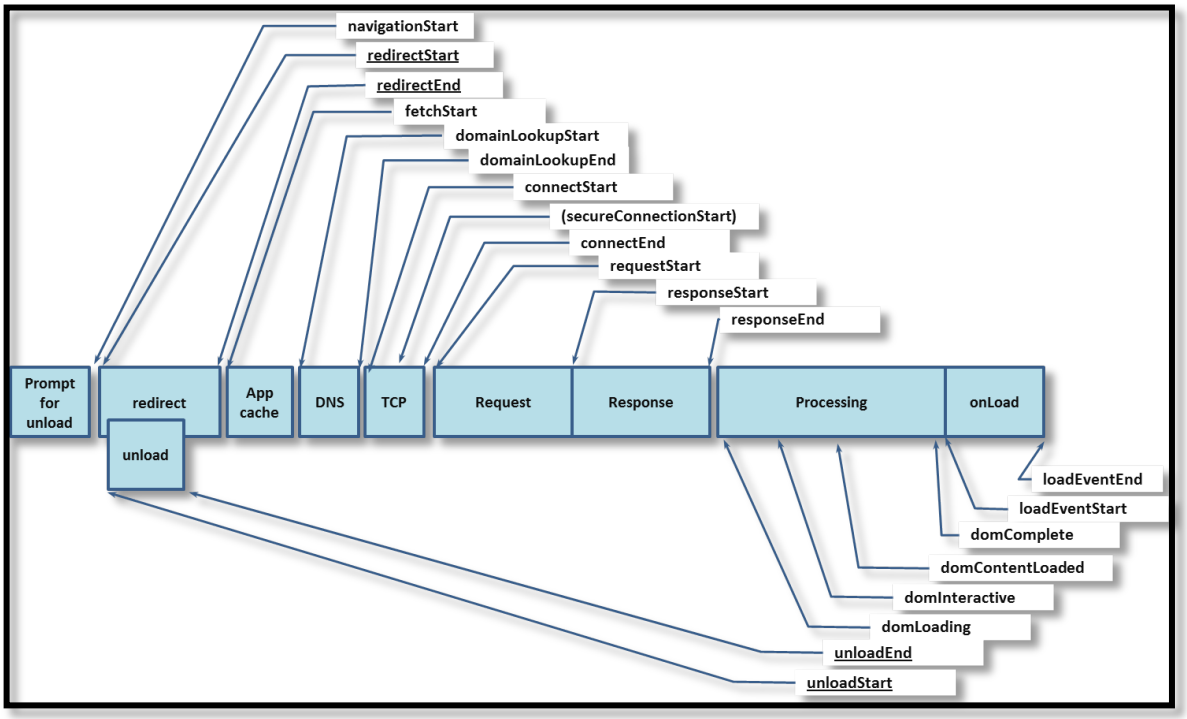

Figura 12: Modelo de processo da API *Navigation Timing*. Fonte: W3C, 2012.

O processo é constituído de atributos que a cada etapa do carregamento, gravam a data e hora exata que o evento ocorreu. De acordo com a W3C (2012) cada marca possui uma função especifica que será descrita abaixo.

Tabela 2 - Atributos da Interface Performance Timing

Atributo	Descrição
navigationStart	Retorna o tempo imediatamente após o navegador terminar de carregar o evento *unload* do documento anterior. Se não houver nenhum documento anterior, este atributo deve retornar que o documento atual foi criado.
unloadEventStart	Marca o tempo imediatamente antes do evento *unload* ser disparado. Se não houver nenhum documento anterior ou do documento anterior tem uma origem diferente do que o documento atual, esse atributo deve retornar zero.

Atributo	Descrição
unloadEventEnd	Se o documento anterior e o documento atual têm o mesmo mesma origem, este atributo deve retornar o tempo imediatamente após o navegador do usuário termina o evento de descarregamento do documento anterior. Se não houver nenhum documento anterior ou do documento anterior tem uma origem diferente do que o documento atual ou o descarregamento ainda não está concluído, este atributo deve retornar zero.
redirectStart	Se há redirecionamentos HTTP este atributo deve retornar a hora de início da busca que inicia o redirecionamento. Caso contrário, deve retornar zero.
redirectEnd	Se há redirecionamentos HTTP este atributo deve retornar a hora de início da busca que termina o redirecionamento. Caso contrário, deve retornar zero.
fetchStart	Se o novo recurso deve ser buscado através de HTTP GET ou equivalente, o atributo marca o momento imediatamente antes que o navegador comece a verificar quaisquer caches de aplicação relevantes. Caso contrário, ele deve retornar o tempo em que o navegador inicia a buscar do recurso.
domainLookupStart	Deve retornar o tempo imediatamente antes que o navegador inicie a resolução do nome do domínio.
domainLookupEnd	Deve retornar o tempo imediatamente após o navegador terminar a pesquisa de resolução do nome do domínio.

Atributo	Descrição
connectStart	Retorna o tempo imediatamente antes que o navegador estabeleça conexão com o servidor para receber o documento.
connectEnd	Retorna o tempo imediatamente após o navegador estabelecera conexão com o servidor para recuperar o documento atual.
secureConnectionStart	Esse atributo é opcional. Os navegadores que não possuem esse atributo disponível devem defini-lo como indefinido. Quando este atributo está disponível, se o protocolo da página atual é HTTPS, este atributo deve retornar o tempo imediatamente antes que o navegador inicie o processo. Proteger a conexão atual. Se esse atributo não estiver disponível este atributo deve retornar zero.
requestStart	Deve retornar o tempo imediatamente antes que o navegador inicie uma solicitação ao servidor, ou a partir de *caches* de aplicativos relevantes ou a partir de recursos locais.
responseStart	Retorna o tempo imediatamente após o navegador receber o primeiro *byte* da resposta do servidor, ou a partir de *caches* de aplicativos relevantes ou a partir de recursos locais.
responseEnd	Retorna o tempo imediatamente após o navegador receber o último *byte* do documento atual ou imediatamente antes da conexão de transporte ser fechada, o que ocorrer primeiro. O documento pode receber as informações a partir do servidor, *caches* de

Atributo	Descrição
	aplicativos relevantes ou de recursos locais.
domLoading	Primeira marca de data e hora de todo o processo do DOM, quando o navegador está prestes a começar a analisar os primeiros bytes recebidos do documento HTML.
domInteractive	Marca o ponto em que o navegador termina de analisar todo o HTML e a construção do DOM é concluída.
domContentLoadedEventStar	Marca o ponto em que o DOM está começando a carregar arquivos externos como estilos e *scripts*.
domContentLoadedEventEnd	Marca o ponto em que o DOM está pronto e não há folhas de estilo bloqueando a execução de *scripts*.
domComplete	Marca o ponto em que todo o processamento do DOM foi concluído e todos os recursos da página como imagens, estilo e *scripts* foram transferidos.
loadEventStart	Retorna o tempo imediatamente antes do evento l*oad* do documento atual é acionado. Ele deve retornar zero quando o evento não foi iniciado.
loadEventEnd	Retorna o momento em que o evento de carregamento do documento foi concluído. Ele deve retornar zero quando o evento não foi iniciado ou a página não foi carregada.

Fonte: Adaptado de (W3C, 2012).

3. DESENVOLVIMENTO

O desenvolvimento do presente trabalho será dividido em quatro etapas. A primeira etapa foi o desenvolvimento dos protótipos. Em seguida a criação da ferramenta proposta. A terceira consiste na utilização dos protótipos na ferramenta. A última etapa diz respeito a análise dos dados coletado.

3.1. PROTÓTIPOS

Tendo em vista que, para o desenvolvimento deste trabalho, seria preciso encontrar um meio para avaliar o design responsivo. Chegou-se à conclusão que um modo de realizar testes comparativos práticos, seria a criação de duas páginas *Web* similares, uma contendo as técnicas de design responsivo, a outra não.

A primeira etapa do desenvolvimento foi a construção dos protótipos, que foram divididos em dois grupos: responsivos e estáticos. Os responsivos foram aplicados técnicas de design responsivo, tais como: medidas em porcentagem e tamanho de fontes na medida EM. Os estáticos foram aplicados as técnicas tradicionais, tamanho do *layout* fixo e as fontes são medidas em *pixel*.

Foram criados dez protótipos, cinco responsivos e cinco estáticos. Aparentemente o responsivo e o estático são idênticos quando visualizados de uma tela grande como a de um computador *desktop.* A diferença ocorre quando são visualizados em telas menores. O responsivo se adapta a tela e o estático não.

Em relação ao tamanho dos arquivos, foi determinado que os arquivos HTML deveriam conter um número significante de linhas pois, o intuito do trabalho era causar algum tipo de estresse ao navegador.

Os *layouts* podem ser divididos em pares, ou seja, cada par exerce uma mesma função, e seus arquivos tem tamanhos semelhantes ou iguais.

Segundo Macotte (2010), as técnicas de design responsivo podem ser aplicadas em diversos recursos de uma página, como layout da página, fontes, imagens, vídeos e por este motivo foi definido avaliar cada recurso separadamente e também analisar os elementos de uma página trabalhando em.

A seguir serão apresentados os protótipos, eles foram divididos em funcionalidades, sendo elas: *layout*, fontes, imagem, vídeo e página. Com esta

divisão foi possível analisar cada componente separadamente e também analisar o funcionamento dos componentes trabalhando em conjunto.

3.1.1. *LAYOUT*

O *layout* de uma página é responsável por organizar e determinar a posição de cada elemento em uma página HTML. O *layout* proposto tem duas colunas: um cabeçalho e um rodapé. Foram utilizados recursos do HTML5 baseando-se nas novas *tags: aside* e *footer.*

O protótipo de *layout*, contém apenas elementos do tipo *div* e algumas fontes dentro das *divs*.

Na estrutura do CSS do protótipo responsivo os elementos são estruturados usando porcentagem, por este motivo quando o navegador altera o tamanho da tela, a estrutura da página também o acompanha.

No caso dos protótipos de *layout*, não ouve diferenças significativas entre os tamanhos arquivos dos protótipos estático e responsivo.

3.1.2. FONTES

As fontes exercem um papel fundamental no desenvolvimento *Web,* são por meio delas que o usuário consegue visualizar textos. O protótipo de fontes, trabalha com diferentes tamanhos, sendo eles: h1, h2, h3, h4, h5 e h6.

O arquivo CSS do protótipo de fontes estáticas define o tamanho das fontes como h1 com 38px, h2 com 32px, h3 com 18px, h4 com 16px, h5 com 12px e h6 com 8px. O arquivo CSS do responsivo define o tamanho das fontes como h1 com 3.9em, h2 com 3.3em, h3 com 1.8em, h4 com 1.6em, h5 com 1.2em e h6 com 0.8em.

Todas as *tags* de ambos os protótipos estavam dentro de uma *div* de tamanho 960px.

Espera-se que com essas configurações nos arquivos CSS, o protótipo responsivo, aumentem ou diminuam de tamanho conforme o dispositivo que foi acessado.

Os arquivos dos protótipos de fontes também não apresentaram diferenças entre o estático e o responsivo.

3.1.3. IMAGEM

As imagens são recursos visuais muito utilizados no desenvolvido *Web,* e por esse motivo, foi escolhido testar este tipo de recurso separadamente.

O protótipo imagem é relativamente simples, são 4834 *tags img* chamando a mesma imagem na página. A imagem de extensão JPG tem 2560px de largura por 1600px de altura.

O responsivo tem em seu arquivo CSS uma propriedade dizendo que ele sempre ocupará 100% da largura tela, e sua altura seja ajustada automaticamente. O protótipo estático utiliza a mesma imagem, porém em seu arquivo CSS seu tamanho é definido com 1024px de largura e 800px de altura.

Os protótipos de imagem também não apresentaram diferenças significativas nos tamanhos de seus arquivos, sendo que a diferença dos tamanhos não ultrapassaram 1kb.

3.1.4. VÍDEO

Outro tipo de recurso utilizado no desenvolvimento de páginas *Web,* são os vídeos. A utilização desse recurso proporciona uma maior interatividade com o usuário e por esse motivo foi testado separadamente.

O tratamento de vídeos em sites responsivo é bem semelhante ao tratamento de imagens, ou seja, eles devem ocupar 100% da área do *layout* em que está posicionado. Foram utilizadas 21 *tags* do tipo *vídeo* no protótipo, e o vídeo escolhido para o teste se encontra no formato MP4 e possui 9 MB.

Para o estático foi definido um tamanho fixo de 1024px de largura e 800px de altura, foi definido este tamanho para ocupar o tamanho inteiro da tela da máquina testada. O responsivo possui uma especificação que sua largura ocupará 100% do *layout* em que estiver, e sua altura será configurada automaticamente.

Os protótipos do tipo vídeo, não apresentaram diferenças nos tamanhos de seus arquivos.

3.1.5. PÁGINA

O protótipo página pode ser considerado o mais importante, pois o mesmo engloba todos os outros recursos apresentados anteriormente.

Foi proposto um modelo de página simples, com uma imagem de cabeçalho, um *layout* de duas colunas, sendo elas: uma coluna contendo o conteúdo principal, como imagens e artigos; e outra com o menu, vídeos e por final o rodapé.

A diferença entre os outros protótipos, se dá na utilização de recursos de *media queries* para tratar as páginas fornecidas aos dispositivos móveis.

O responsivo e o estático se visualizados a partir de um dispositivo com tela grande, vão ter a mesma aparência, para isso, foram definidas duas *medias queries* para o protótipo. Uma para resoluções com 768px de largura e outra para 320px de largura.

Quando o protótipo responsivo é acessado de um dispositivo com 768px de largura, os vídeos contidos no protótipo se adaptam a tela toda e o menu é exibido na forma vertical, os demais conteúdos são proporcionalmente diminuídos. Já quando o protótipo é acessado de um dispositivo com no mínimo 320px de largura, o menu assume toda a largura da tela, os artigos são mostrados em apenas uma coluna e os vídeos ocupam a largura toda do *layout*.

A Tabela 3 mostra o comparativo de tamanho dos protótipos do tipo página.

Tabela 3 - Comparativo entre protótipos de página estático e responsivo.

Nome	**Extensão**	**Tamanho Estático**	**Tamanho Responsivo**
style	CSS	1kb	2kb
exemplo	HTML	227kb	227kb
img	JPG	307kb	307kb
uenp-logo	PNG	219kb	219kb
responsive	MP4	9mb	9mb

Fonte: Autoria própria

Neste caso em específico, o arquivo CSS do protótipo responsivo ficou 1kb maior, pois foram definidas características específicas para dois tamanhos de tela, utilizando o recurso *media queries*.

Foram criadas essas *medias queries,* para se adaptarem a ferramenta descrita na próxima sessão.

3.2. FERRAMENTA

A criação da ferramenta denominada *RWDBenchmark*, foi primordial para o decorrer da pesquisa. O fator que foi levado em consideração na decisão da criação da ferramenta, foi o escopo a ser avaliado, no caso o foco foi o carregamento do DOM.

A primeira etapa foi a realização de um estudo a respeito da API *Navigation Timing*, pois a mesma tem o papel principal no desenvolvimento da ferramenta.

A segunda etapa foi definir quais tamanhos de tela a ferramenta iria simular, para tanto foram definidos três tamanhos diferentes: *mobile*, *tablet* e *desktop*.

Os tamanhos definidos das telas simuladas, foram escolhidos com base na pesquisa realizada pela *Mobify*, segundo Abasov (2012) as medidas 320px de largura e 480px de altura representam 11.4% dos usuários que acessam a internet, por meio desta medida foi definido como tamanho *mobile*, o tamanho *tablet* foi definido como 768px de largura e 1024px de altura por representar 7.3% dos usuários, e o tamanho do *desktop* foi definido como 1280px de largura por 800px de altura por representar 19.5% dos usuários da internet.

Na Figura 13 será apresentado o funcionamento da ferramenta.

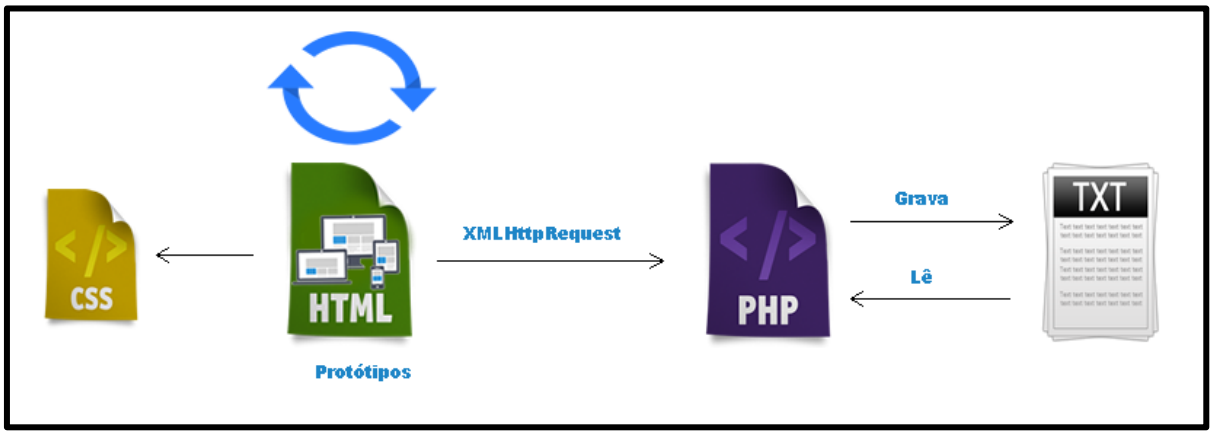

Figura 13: Funcionamento da ferramenta. Fonte: Autoria própria.

Na Figura 13 podemos observar como é o funcionamento da ferramenta proposta. O processo se inicia quando o protótipo realiza uma requisição de seu estilo, em seguida ele carrega o HTML junto ao CSS, é nesta hora que a ferramenta efetua a coleta de dados.

Após a coleta, a ferramenta envia os dados por meio de uma chamada *XMLHttpRequest*, onde estão contidas todas informações da API *Navigaton Timing* e informações como nome do protótipo e se ele é responsivo ou estático.

A última etapa é persistir os dados, para tanto foi definido que os dados seriam gravados em arquivos do tipo texto, utilizando recursos da linguagem PHP.

A realização dos testes de desempenho exige uma certa quantidade de execuções, esta quantidade é definida nas linhas de código do programa. Foi utilizado o recurso de sessão do PHP para controlar o número de execuções, onde, a cada execução a variável de sessão realiza um incremento, até que o valor seja

igual ao definido no sistema. A cada nova bateria de testes a sessão no navegador deve ser limpa.

O resultado final da ferramenta desenvolvida, pode ser observado na Figura 14, onde usuário pode escolher entre os três tamanhos de tela pré-definidos, colocar o endereço do site a ser testado e em seguida efetuar o teste.

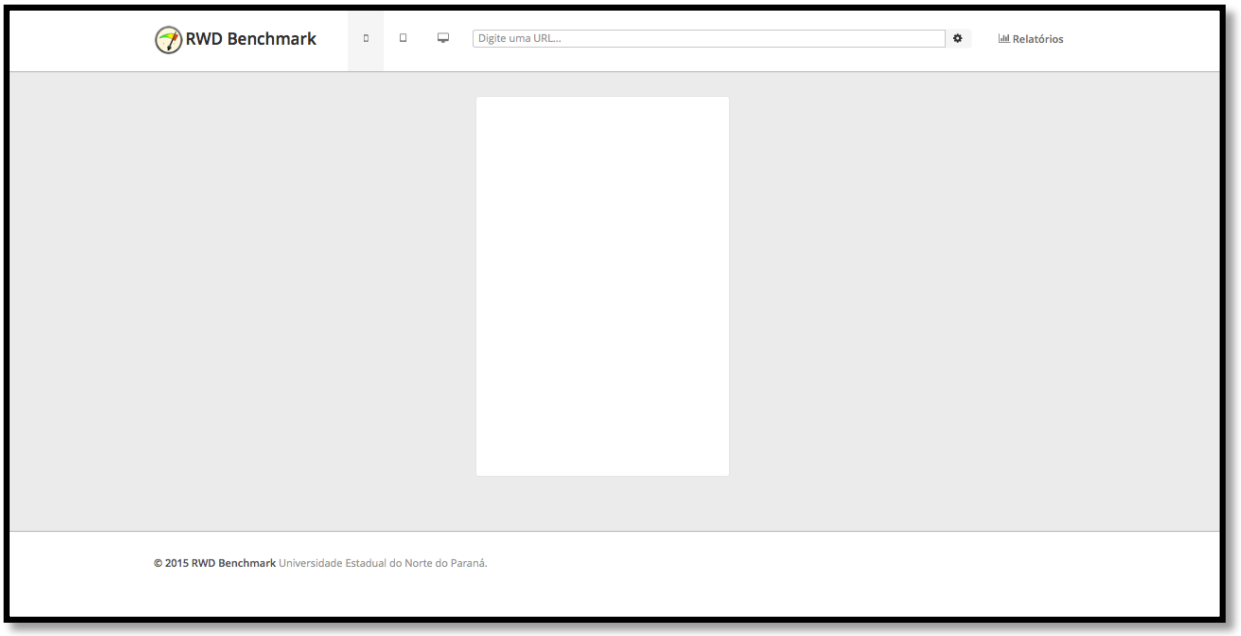

Figura 14: Tela inicial do sistema. Fonte: Autoria própria

A próxima sessão é voltada para explicar como a ferramenta foi utilizada para avaliar os protótipos.

3.3. USO DOS PROTÓTIPOS NA FERRAMENTA

Após o desenvolvimento dos protótipos e da ferramenta foi possível a realização dos testes.

Foram definidas cem execuções de rotinas de cada protótipo e em cada simulação de tamanho de tela.

Os testes foram realizados em *localhost*. Foram usados os navegadores Microsoft Internet Explorer 11 e Google Chrome versão 44. Também foram desativadas configurações que dizem respeito a *cache* dos navegadores, que se encontram no menu de opções.

A máquina utilizada possui as seguintes configurações: processador Core i3-330M, 4 Gb de memória RAM e HDD de 500 Gb. O sistema operacional definido foi o Windows 7 Ultimate 64 bits .

Os dados foram registrados pela ferramenta desenvolvida, gerando uma grande quantidade de informações referente ao tempo de carregamento da página. Nas coletas efetuadas foram gravados vinte e um atributos fornecidos pela API *Navigation Timing,* e o horário em que o teste foi feito.

Ao todo foram gerados sessenta arquivos TXT(*Text*), uma vez que os dados foram gravados, foi possível realizar diversas analises com os tempos, que serão descritos na próxima sessão.

3.4. ANÁLISE DOS DADOS

Após a coleta de dados efetuada por meio da ferramenta desenvolvida para testar os protótipos, foi possível definir quais seriam as métricas de avaliação.

De acordo com a (W3C, 2012), existem diversos atributos que podem ser medidos quando se utiliza a API *Navigation Timing*. Foi definido como métrica para este trabalho, os tempos que dizem respeito ao carregamento do DOM.

Durante o carregamento do DOM, o navegador precisa carregar imagens, vídeos, elementos HTML, estilos CSS, JavaScript e todos recursos que a página possui. Segundo Grigorik (2014) este processo é chamado de caminho crítico da renderização do DOM.

Como pode ser observado na figura 15, o carregamento crítico do DOM pode ser calculado por meio da subtração do atributo *domComplete* menos o *domLoading* (GRIGORIK, 2014). Diante disso presente trabalho teve como o foco somente estas métricas.

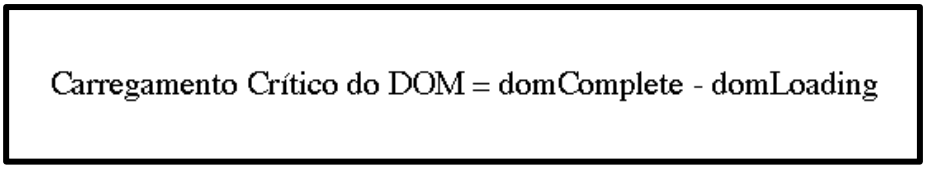

Figura 15: Formula do carregamento crítico do DOM. Fonte: Adaptado de GRIGORIK, 2014.

Os arquivos TXT que contém as execuções, foram transformados em arquivos do tipo CSV, facilitando o modo como os dados seriam trabalhados. Depois de aplicar a métrica proposta, foi possível analisar os dados.

Para poder realizar as comparações entre os protótipos responsivos e estáticos, nas três simulações de resolução, optamos por trabalhar com a média dos tempos obtidos do caminho crítico; analisando as diferenças dos tempos das execuções e os desvios padrão das execuções.

Não foram considerados os dados que se mostraram instáveis, estes apresentaram tempos de execuções não condizentes com os demais, porque não contextualizam o objetivo deste trabalho. Os dados foram ordenados em ordem crescente, e em seguida foram descartados 10% dos resultados em todos os testes.

4. RESULTADOS OBTIDOS

Esta sessão é destinada a apresentar os resultados obtidos no o decorrer do trabalho. Os dados coletados dos foram divididos em duas tabelas, onde cada tabela apresenta os resultados de seu respectivo navegador.

4.1. GOOGLE CHROME

Na Tabela 4 serão apresentados os dados coletados a partir do navegador Google Chrome.

Tabela 4 - Resultados obtidos por meio do navegador Google Chrome.

		Estático			Responsivo		
Protótipo		**Desktop**	**Tablet**	**Mobile**	**Desktop**	**Tablet**	**Mobile**
Página	Média (ms)	3280.02	3283.20	3281.09	3285.11	3293.81	3264.58
	Desvio Padrão (ms)	25.73	25.49	25.14	26.42	27.52	23.94
Layout	Média (ms)	25.30	24.34	24.84	25.26	25.07	24.34
	Desvio Padrão	6.36	5.05	3.67	4.95	4.24	3.30
Fontes	Média (ms)	154.03	143.16	137.44	153.32	143.29	138.83
	Desvio Padrão (ms)	17.66	8.51	7.30	17.79	9.70	10.18
Imagens	Média (ms)	588.51	627.49	570.22	609.33	597.32	611.53
	Desvio Padrão (ms)	3027.58	2988.69	2994.35	3004.80	3066.49	3023.34
Vídeos	Média (ms)	3169.44	3172.86	3171.26	3170.11	3172.68	3172.03
	Desvio Padrão (ms)	2.33	2.10	3.30	2.65	4.59	4.70

Fonte: Autoria própria

A seguir serão apresentados os gráficos utilizando a média do tempo de execução e o desvio padrão dos protótipos de página, no navegador Google Chrome.

No Gráfico1 serão mostrados os resultados das execuções simulando o tamanho de tela *desktop* no navegador Google Chrome.

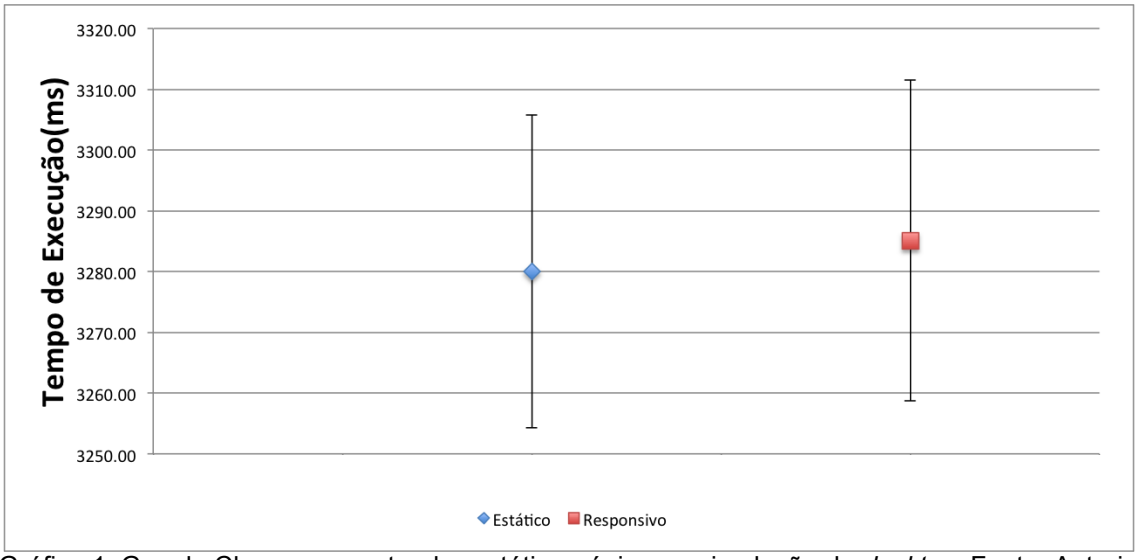

Gráfico 1: Google Chrome executando protótipo página na simulação de *desktop.* Fonte: Autoria própria.

Pode ser observado que o protótipo responsivo obteve uma performance inferior ao do estático, demorando cerca de cinco *ms* a mais para ser processado. O desvio padrão do estático ficou em 25.75 ms e o responsivo em 26.42 ms.

No Gráfico 2, observa-se a simulação de tela do tipo *tablet* no protótipo página, por meio do navegador Google Chrome.

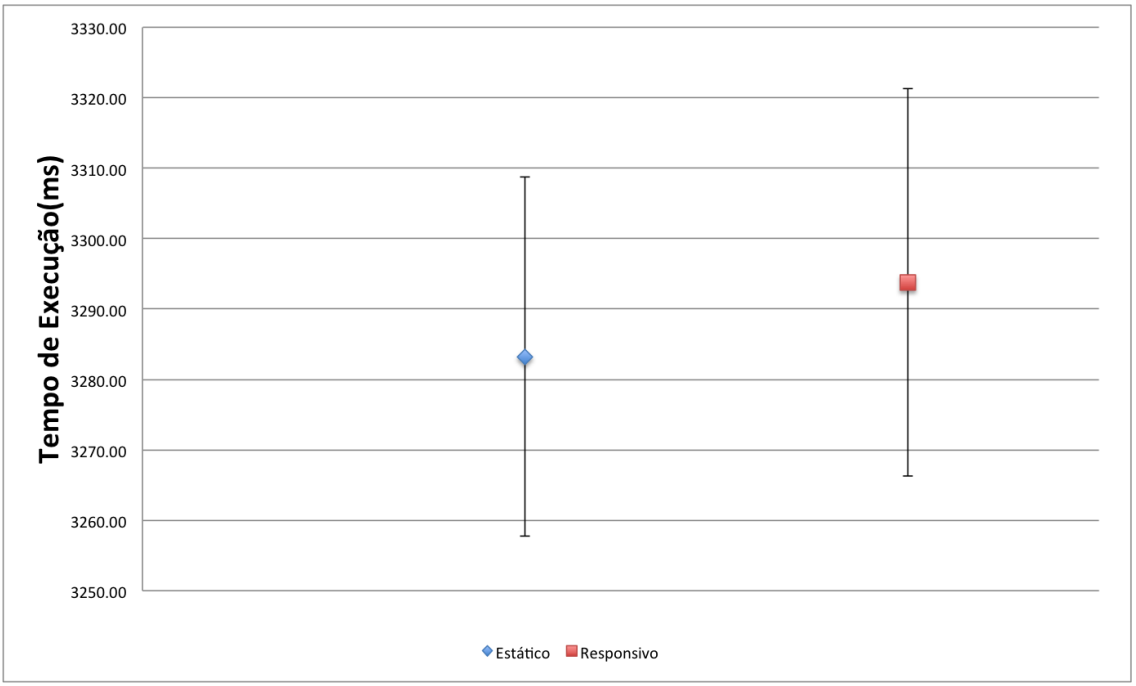

Gráfico 2: Google Chrome executando o protótipo página na simulação *tablet.* Fonte: Autoria própria

O teste mostrou que o desempenho do estático é superior ao do responsivo em cerca de 10 ms de diferença. O desvio padrão do estático ficou em 25.49ms, já o do responsivo ficou em 27.52ms.

Os testes realizados utilizando a simulação de tela *mobile,* demostram o protótipo estático com um desempenho inferior ao responsivo como pode ser observado no Gráfico 3.

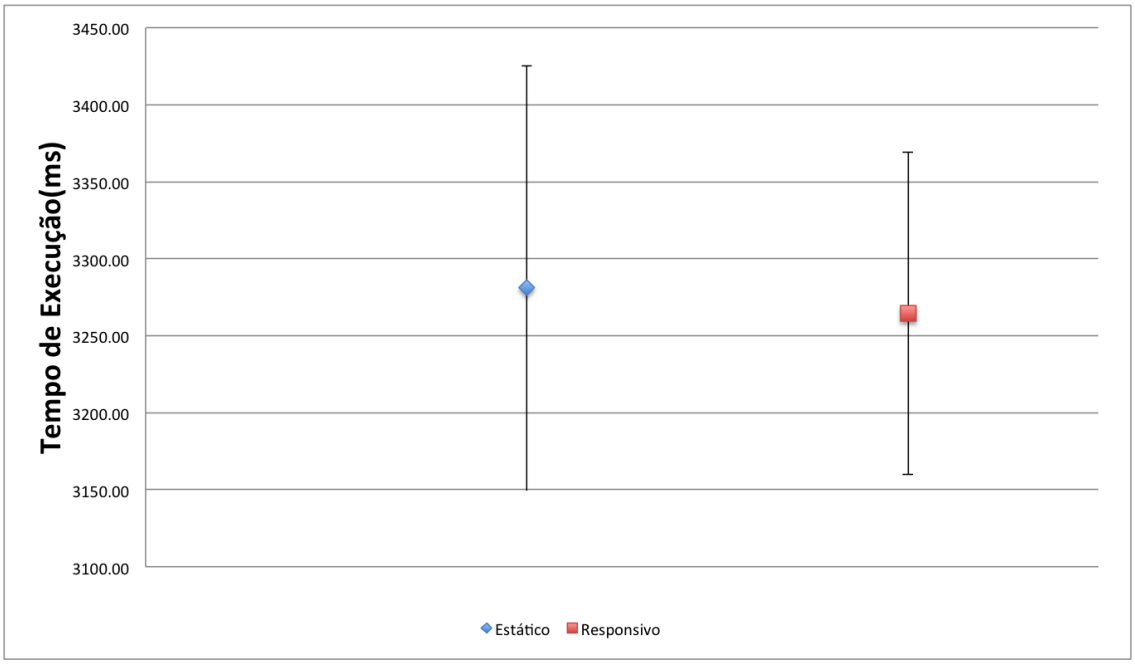

Gráfico 3: Google Chrome executando o protótipo página na simulação mobile. Fonte: Autoria própria.

O protótipo estático foi cerca de 17 ms mais lento que o responsivo. O desvio do responsivo ficou em 23.94 e o estático em 25.14.

4.2. INTERNET EXPLORER

Os resultados das médias e desvios padrão dos protótipos responsivos e estáticos utilizando o navegador Internet Explorer, em cada simulação de tamanho de tela, pode ser observado na Tabela 5.

Tabela 5 - Resultados obtidos por meio do navegador Internet Explorer.

		Estático			Responsivo		
Protótipo		**Desktop**	**Tablet**	**Mobile**	**Desktop**	**Tablet**	**Mobile**
Página	Média (ms)	738.72	733.20	667.91	826.32	868.99	732.11
	Desvio Padrão (ms)	140.72	137.58	143.97	104.12	113.81	104.79
Layout	Média (ms)	34.58	36.12	34.58	35.56	35.92	36.01
	Desvio Padrão (ms)	2.02	5.27	1.45	1.87	2.45	1.45
Fontes	Média (ms)	967.79	864.20	886.90	839.28	750.57	949.86
	Desvio Padrão (ms)	136.34	265.23	232.54	285.07	317.47	150.11
Imagens	Média (ms)	588.51	627.49	570.22	609.33	597.32	611.53
	Desvio Padrão (ms)	483.25	518.21	492.68	495.63	479.17	476.41
Vídeos	Média (ms)	335.93	358.78	358.47	372.2	357.44	366.96
	Desvio Padrão (ms)	104.57	83.99	119.75	104.57	72.82	90.50

Fonte: Autoria própria

Os resultados do desempenho dos protótipos de páginas responsivos e estáticos serão apresentados a seguir em forma de gráficos. O Gráfico 4, representa o protótipo de página na simulação de *desktop*, utilizando o navegador Internet Explorer.

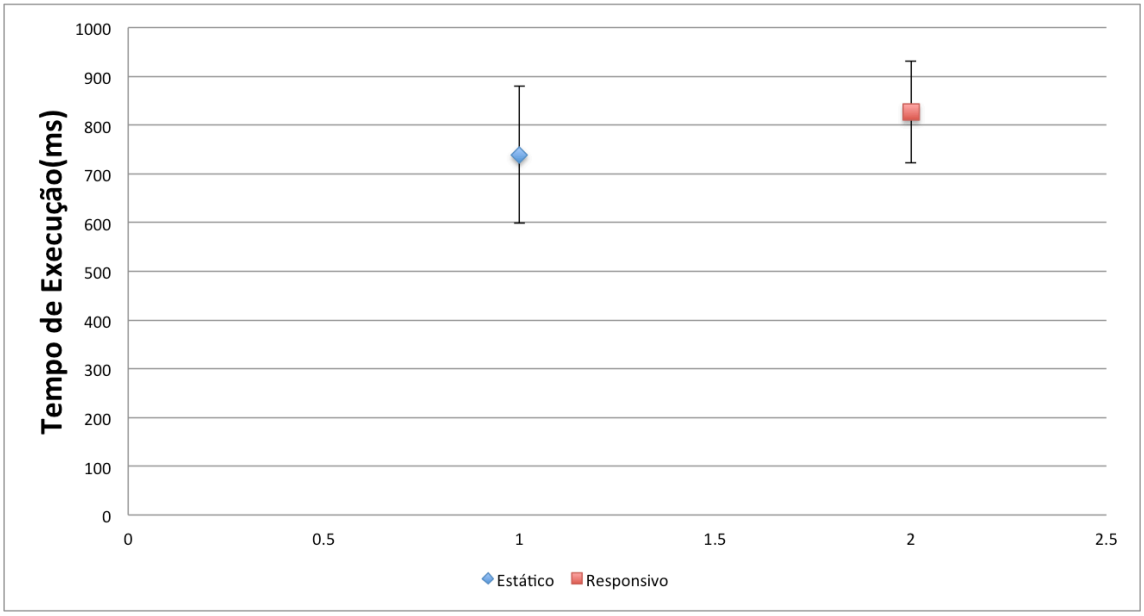

Gráfico 4: - Internet Explorer executando o protótipo página na simulação desktop. Fonte: Autoria própria.

Neste caso o desempenho do protótipo estático foi superior ao responsivo que precisou de cerca de 8 ms a mais para ser processado na simulação de tela *desktop*. Os desvios padrão ficaram em 140.44ms para o estático e 104.12 ms para o responsivo.

Os resultados do protótipo página em simulação de tamanho tela do tipo *tablet,* utilizando o navegador Internet Explorer é apresentado no Gráfico 5.

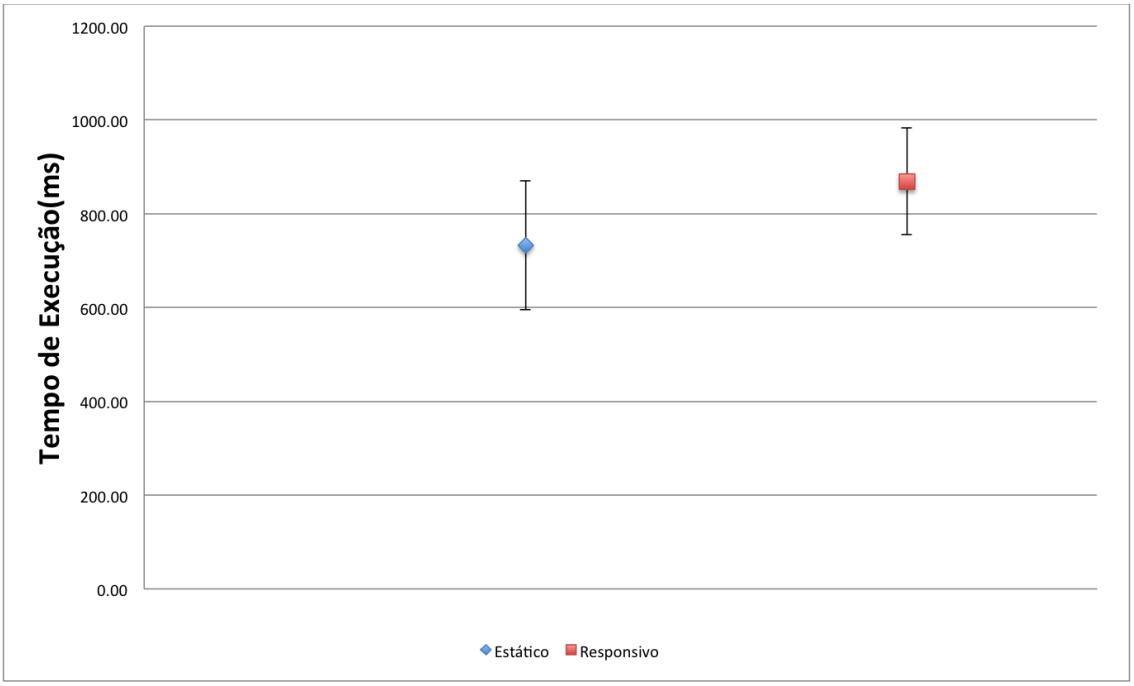

Gráfico 5: Internet Explorer executando o protótipo página na simulação tablete. Fonte: Autoria própria.

Foi observado uma diferença entre os protótipos estático e responsivo, sendo que estático mostrou o melhor desempenho. É possível notar uma diferença de cerca de 135 ms entre o processamento do estático e do responsivo. O desvio padrão do estático ficou em 137.58 ms e o do estático ficou em 113.81 ms.

No Gráfico 6, estão apresentados os testes realizados utilizando o navegador Internet Explorer, se mostrando melhor no protótipo estático.

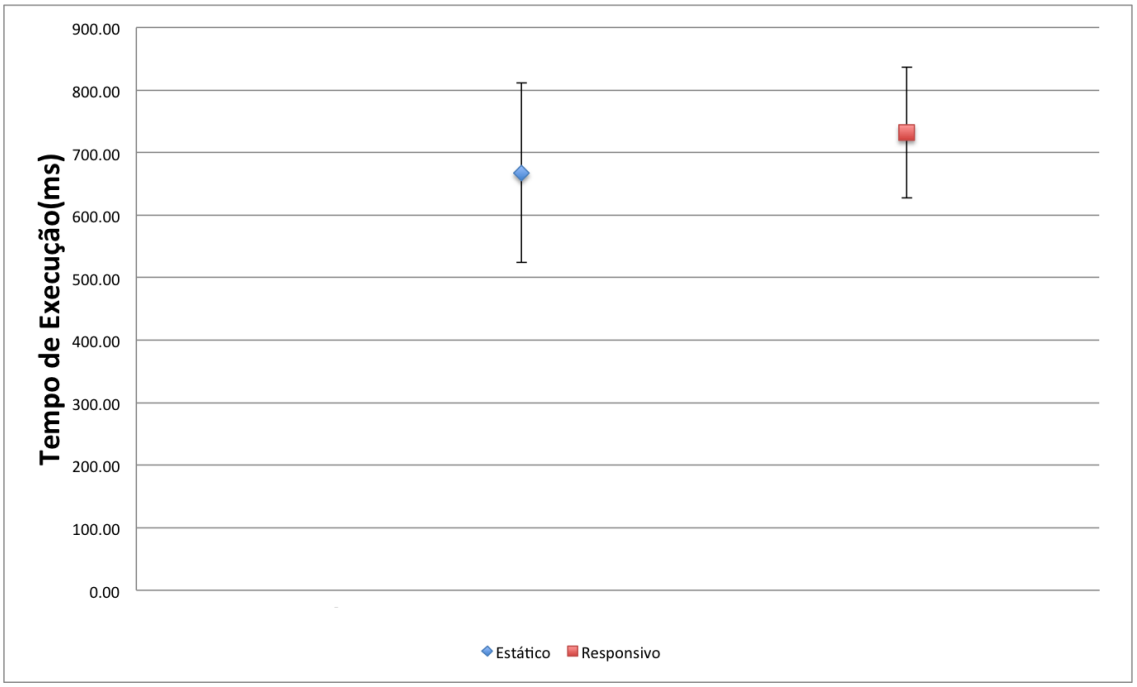

Gráfico 6: Internet Explorer executando o protótipo página na simulação mobile. Fonte: Autoria própria.

O responsivo utilizou cerca de 65 ms a mais que o estático para processar o protótipo de página na simulação de tela *mobile.* O desvio padrão do estático ficou em 143.97 ms e o responsivo ficou em 104.79ms.

A próxima sessão é destinada a analisar mais afundo os dados, e realizar algumas conclusões baseadas nos dados apresentados.

4.3. ANÁLISE DOS RESULTADOS

De acordo com as Tabelas 4 e 5, apresentadas nas seções anteriores, pode-se obter resultados e partir deles, foram realizadas algumas análises com os dados.

O navegador Internet Explorer obteve um melhor desempenho na maioria dos testes em relação ao navegador Google Chrome. Porém o Chrome apresentou um desvio padrão menor na maior parte dos testes.

Em um comparativo entre os dez protótipos desenvolvidos considerando os desvios padrão, pode-se constatar que os protótipos estáticos foram 60% mais rápidos que os responsivos no navegador Google Chrome, e 53% no navegador Internet Explorer.

A menor diferença encontrada entre os protótipos no Google Chrome foi de 0.58ms, do protótipo de tipo fontes estáticas na simulação de *desktop* e a maior foi de 47.63ms, encontrada no protótipo de imagens responsivas na simulação de *tablet*.

No Internet Explorer a menor diferença foi de 0.83ms encontrado no protótipo do tipo *layout* responsivo na simulação de *desktop* e a maior foi de 112.02ms localizado no protótipo de página responsivo na simulação de *tablet*.

Considerando os desvios padrão, o protótipo de vídeo se mostrou mais estável entre os protótipos testados no Google Chrome. Já no Internet Explorer, o que se apresentou mais estável foi o protótipo de *layout.*

Observando os desvios padrão, os únicos casos que apresentaram diferenças significativas nas simulações de tamanho de tela, foram os testes realizados, no protótipo de página. Os testes em ambos os navegadores, na simulação *mobile* apresentaram o responsivo como mais rápido. Nos outros protótipos não foram encontradas diferenças significativas entre as simulações de tela.

Os protótipos de imagens se demonstraram instáveis em ambos os navegadores, porém os desvios padrão apresentados no Google Chrome foi maior que no Internet Explorer.

Já os protótipos de vídeos foram estáveis no Chrome e instáveis no o Internet Explorer. Porém o desempenho do Explorer foi melhor.

O pior desempenho no navegador Internet Explorer foi nos protótipos de fontes, sendo o caso onde o Chrome foi mais rápido.

Com os conhecimentos obtidos através dos dados coletados, foi preciso realizar uma análise e reflexão sobre os resultados, a fim de se chegar a uma conclusão relevante.

5. CONCLUSÃO

O objetivo deste trabalho consiste em analisar o desempenho de páginas responsivas em relação as páginas não responsivas. Por meio do uso das técnicas de design responsivo, foi possível a criação dos protótipos, responsivos e estáticos. O desenvolvimento da ferramenta *RWDBenchmark,* permitiu simular diferentes tamanhos de tela e tornou possível a coleta dos dados. Com os dados coletados, foi possível realizar uma analise e obter uma resposta para pergunta que ficou em aberto na seção problema do capitulo 1. O desempenho de uma página *Web* é afetado quando aplicado às técnicas de design responsivo?

Pode se concluir, por meio dos dados apresentados, que as técnicas de design responsivo aumentam o tempo de carregamento do DOM, e dependendo do navegador utilizado, o tempo pode se agravar. Contudo é valido ressaltar que na maioria dos casos a diferença do tempo de carregamento não foi significativa.

Um dos desafios encontrados na execução do trabalho, foi a dificuldade em encontrar formas de analisar a performance do carregamento do DOM. Outro desafio foi o estudo das técnicas de design responsivo, pois existem diferentes formas de desenvolver páginas responsivas, principalmente quando o desenvolvimento é realizado conjunto com a linguagem JavaScript.

Como trabalhos futuros, sugere-se melhorias na ferramenta desenvolvida como a implementação relatórios em tempo real e a possibilidade de utilização online. Além disso o trabalho proposto analisou somente uma das diversas métricas de medições fornecidas pela API *Navigation Timing*, outros fatores ainda podem ser exploradas, tais como servidor e rede.

REFERÊNCIAS

ABASOV, M. **Global Screen Size Diversity Infographic: Sizing Up Man's New Best Friend**. Disponível em: <http://www.mobify.com/blog/global-screen-size-diversity/>

BAKER, T. **JavaScript Performance Monitoring and Visualization**.1ed. Apress, 2012.

CAMPOS, V; VASCONCELOS B. A. **Teste de desempenho em Desktops Virtuais**. Centro Universitário Ritter dos Reis, 2010.

CANIUSE. **Navigation Timing API**.CANIUSE 2015. Disponível em: < http://caniuse.com/#feat=nav-timing/> - Acesso em: 5 jul.2015.

DUTTON, S. **Measuring Page LoadSpeedwith Navigation Timing.** Disponível em: <http://www.html5rocks.com/en/tutorials/webperformance/basics/ /> - Acesso em: 1 jul.2015.

EIS.D, **Manipulando a meta tag Viewport.** Tabless 2011. Disponível em http://tableless.com.br/manipulando-metatag-viewport/> - Acesso em 3 dez.2012

GRIGORIK, I. **Measuring the Critical Rendering Path with NavigationTiming.** Disponível em: <https://developers.google.com/web/fundamentals/performance/critical-rendering-path/measure-crp?hl=en> Acesso em 10 jul.2015

GOOGLE ANALYTICS. 2012. **Measure your Web site's performance with improved Site Speed reports**; Disponível em: <http://analytics.blogspot.com/2012/03/measure-your-websites-performance-with.html.> Acesso em: 1 jul.2015.

LEWIS, P. **Avoidlarge, complex layouts and layout trashing**. Disponível em: <https://developers.google.com/web/fundamentals/performance/rendering/avoid-large-complex-layouts-and-layout-thrashing?hl=en> Acesso em 10 jul.2015

LOPES,S. **A Web Mobile: Programe para um mundo de muitos dispositivos.** 1 ed. São Paulo: Casa do Código, 2013.

MARCOTTE, E. **Responsive web design**. A List Apart Magazine, 2010. Disponível em:< http://alistapart.com/article/responsive-web-design/ >. Acesso em: 20 jun.2014

MAYER, M. **In Search of a better, faster, stronger Web**. Disponível em: <http://assets.en.oreilly.com/1/event/29/Keynote%20Presentation%202.pdf>

MAZZA, L. **HTML5 e CSS3: Domine a web do futuro.** 1 ed. São Paulo: Casa do Código, 2013.

MEENAN, P. **How Fast is Your Website.** Disponível em: <http://queue.acm.org/detail.cfm?id=2446236> Acesso em: 1 jul.2015.

NATDA, V. **Responsive Web Design**. Natda 2013. Disponível em: <http://ejournals.saurashtrauniversity.edu/index.php/eduvantage/article/view/18/pdf/> - Acesso em: 3 nov.2014.

REIS, R. T. **Desenvolvimento Web com o Uso de Padrões: Tecnologias e Tendências.** UFJF, Juiz De Fora – MG. 2007.

ROSA,D.; SILVA,T.L . **Adaptação de interfaces para dispositivos móveis com HTML5.** Disponível em: < http://www.eati.info/eati/2013/assets/anais/artigo249.pdf> . Acesso em: 22 jun.2014

SOUDERS, S. **The Performance Golden Rule.** Disponível em: <http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule> Acesso em: 10 ago.2015.

SOUZA, S.C.N.; IGARASHI, W . **Web Design Responsivo no desenvolvimento de aplicações multi - dispositivos.** Disponível 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> . Acesso em: 22 jun.2014

W3C**. Media Queries W3C Recommendation**. Disponível em: <http://www.w3.org/TR/css3-mediaqueries/> - Acesso em: 3 nov.2014.

W3C. **Navigation Timing**.W3C.ORG 2012. Disponível em: <http://www.w3.org/TR/navigation-timing//> - Acesso em: 1 jul.2015.

W3C. **A vocabulary and associated APIs for HTML and XHTML 5**. Disponível em <http://www.w3.org/TR/html5/>. Acessado em 28 jul. 2015.

W3C. **DOM Level 3**. Disponível em: <http://www.w3.org/2004/03/dom-level-3-pr> Acesso em: 1 jul.2015.

W3C. **Selector Level 3**. Disponível em: <http://www.w3.org/TR/css3-selectors> Acesso em: 10 ago.2015.

ZEMEL, T**. Web Design Responsivo**: Páginas adaptáveis para todos os dispositivos. 1 ed. São Paulo: Casa do Código, 2012.